



**STATE UNIVERSITY
OF BANGLADESH**
join the trendsetter

Department of Computer Science and Engineering

Project Report for
Diagnostic Lab Reporting System

AUTHORS

Md. Sohanur Rahman

UG02-40-15-012

Md. Abdullah Al Galib

UG02-40-15-029

SUPERVISOR

Muhammad Masud Tarek

Associate Professor

Department of Computer Science and Engineering
State University of Bangladesh

Certificate

This project has been done by **Md. Sohanur Rahman** and **Md. Abdullah Al Galib** in supervision of **Muhammad Masud Tarek**. This is an academic project for Bachelor of Science in Computer Science and Engineering degree from State University of Bangladesh. Resources of this project are not copied from anywhere. It is also declared that neither this project nor any part thereof has been submitted or is being currently submitted anywhere else for the award of any degree or diploma.

Acknowledgement

I would like to thank **Allah** for wisdom and perseverance given to us for the completion of this project.

This dissertation concludes our Bachelor of Science degree in Computer Science and Engineering at the State University of Bangladesh. Working on this project was both interesting and challenging for us. The successful completion of this task would be incomplete without the mention of people whose ceaseless cooperation made it possible and constant guidance and encouragement crown all efforts with success.

We gratefully acknowledge **Muhammad Masud Tarek** (Assoc. Prof. and Head, Dept. of CSE, SUB) for his guidance, certification, kind cooperation and encouragement that allowed us to progress and broaden our knowledge. His cooperation and guidance were continuous throughout this project.

Also, we would like to pay our gratitude to all the **teachers** of our Department of Computer Science and Engineering for their help and support.

Special thanks to our **friends** for their help and support.

Finally, we pay our respect and sole gratitude to our **parents** who kept their faith in our ability.

September, 2019

Md. Sohanur Rahman

Md. Abdullah Al Galib

Department of Computer Science and Engineering
State University of Bangladesh

Abstract

Diagnostic Lab Reporting System (DLRS) is a web application which will help a patient to book diagnostic tests online. After completing tests, the downloadable report will be available in his/her profile. Users can choose a diagnostic center according to their budget and requirements. Diagnostic admin and staffs will manage all the order processing steps using their custom dashboards.

Using this software, a particular diagnostic center can manage their lab test booking and reports in a very efficient and effective way. Aim of this software is to build an easy diagnostic test booking process for the customers using the internet.

There will be four types of privileged users in the system. A system admin will be able to manage diagnostic centers including admins, diagnostic admins will be able to manage diagnostic staffs, diagnostic staffs will be able to manage customers and each customer will only be able to add and modify his/her information and also can place orders as well.

TABLE OF CONTENTS

CHAPTER 1:	7
1.1 BACKGROUND	7
1.2 PURPOSE	7
1.3 QUICK OVERVIEW	8
CHAPTER 2: REQUIREMENT ENGINEERING	9
2.1 USER REQUIREMENTS	9
2.2 SOFTWARE REQUIREMENT SPECIFICATION	10
2.3 PURPOSE	10
2.4 SCOPE OF PROJECT	10
2.5 DEFINITIONS	10
2.6 OVERALL DESCRIPTION	11
2.7 PROJECT PERSPECTIVE	11
2.8 SOFTWARE REQUIRED	12
2.8.1 Software Required	12
2.8.2 Hardware Required	12
2.9 USERS OF THE SYSTEM	12
2.10 USER CHARACTERISTICS	12
2.11 USER FUNCTIONALITIES AND THEIR SCOPE	12
2.12 CONSTRAINTS	14
2.13 EXTERNAL INTERFACE REQUIREMENTS	15
2.13.1 User Interfaces	15
2.13.2 Hardware Interfaces	15
2.13.3 Software Interfaces	15
2.13.4 Communications Interfaces	15
2.14 FUNCTIONAL REQUIREMENT	16
2.14.1 User Class 1 : System Admin	16
2.14.2 User Class 2 : Diagnostic Admin	18
2.14.3 User Class 3 : Diagnostic Staff	19
2.14.4 User Class 2 : Customer	21
CHAPTER 3: SYSTEM DESIGN & IMPLEMENTATION	23
3.1 SOFTWARE DEVELOPMENT METHOD	23
3.2 WORK PROCESS	24
3.3 STATE TRANSITION DIAGRAM	25
3.4 DATA FLOW DIAGRAM	26
3.5 ENTITY RELATIONSHIP DIAGRAM	27
3.6 DATABASE DESIGN	28
3.6.1 List of Tables	28
CHAPTER 4: IMPLEMENTATION	33
4.1 DEVELOPMENT ENVIRONMENT	33
4.1.1 Common Issues	33

4.2 WEB INTERFACE	33
CHAPTER 5: SOFTWARE TESTING	49
5.1 UNIT TESTING	49
5.1.1 White-Box Testing	49
5.1.2 Black-Box Testing	50
CHAPTER 6: CONCLUSION AND FUTURE WORK.....	50
CHAPTER 7: IMPORTANT SOURCE CODES	51
CHAPTER 8: REFERENCES	64

Chapter 1:

1.1 Background

Diagnostic Lab Reporting System explicitly intended to help the customers to the improvement of efficient test booking procedures including payment gateways. Each customer/patient books a diagnostic test by going to the center, do some paper works, fix a schedule, and make the payment manually, which is difficult and time-consuming. Using this system, a customer can select centers/tests and make orders online just filling up a simple order-form including payment methods as well.

Nowadays, the diagnostic test booking system is running manually and it is getting difficult sometimes to book a test. So we need better system to manage the booking system efficiently.

1.2 Purpose

Purpose of this project is to provide a simple, easy, convenient and efficient system to manage the diagnostic test booking and lab reporting process. This System helps individuals to book their test online. A diagnostic admin can create, edit, delete tests and categories. Diagnostic staffs can approve, reject, verify, confirm, and complete order operation. Customers can able to view and download their reports from the profile and also can re-payment if the payment type was the half.

1.3 Quick Overview

Chapter 2

Includes analysis part of the application. This chapter includes system requirements specification.

Chapter 3

Includes the design principles and structure of the application. Data Flow Diagram (DFD), State-Transition Diagram (STD), and Entity-Relationship Diagram (ERD) are discussed. Database tables are also discussed in this chapter.

Chapter 4

Consists of the discussion about implementation. This chapter includes problems encountered and details of how they were overcome. Also, creating interfaces, developing backend are also added with screenshots in this chapter.

Chapter 5

Includes testing and result of this project.

Chapter 6

Includes the concluding part.

Chapter 7

Contains screenshots of the important source codes.

Chapter 8

Includes references.

The appendix contains all of the source code of this project.

Chapter 2: Requirement Engineering

2.1 User Requirements

- I. Diagnostic admin should be able to assign a staff for certain tasks.
- II. Diagnostic admin should be able to manage test, order, staff and customer operations.
- III. Diagnostic admin should be able to create, delete new tests and categories.
- IV. Diagnostic admin should be able to view new added tests, new added categories, completed order details, and staff status.
- V. Diagnostic staff should be able to approve, reject and verify the order.
- VI. Diagnostic staff should be able to confirm and complete the order.
- VII. Diagnostic staff should be able to upload, send messages and test reports to the customer.
- VIII. Diagnostic staff should be able to view newly added tests and categories, completed and not completed order details and reporting admin.
- IX. Customers should be able to register, login, logout and edit their profile information.
- X. Customers should be able to order tests and view the order processing status, order details, and report list from their profile.
- XI. Customers should be able to view and download their reports from the profile and also can re-payment if the payment type was the half.
- XII. System Admin should be able to manage own profile.
- XIII. System admin should be able to add new diagnostic centers including admin and staff's username and password.
- XIV. System admin should be able to perform various maintainability.
- XV. The overall system should be clean, efficient and convenient to use.
- XVI. The system should be secured as much as possible.

2.2 Software Requirement Specification

The detailed section of Software Requirement Specification (SRS) document for the project Diagnostic Lab Reporting System is as follows. It gives a scope description and overview of everything included in this SRS document.

2.3 Purpose

The purpose of this section is to give a detailed description of the requirements for the Diagnostic Lab Reporting System. It will illustrate the purpose and complete declaration for the development of this system. It will also explain system constraints, interface and interactions with other external objects or applications. This chapter is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system or the development team.

2.4 Scope of project

This project is aimed at developing a web-based system which can manage the activity of Diagnostic Lab Reporting System and could be used by any Diagnostic Center and Ecommerce organizations. Scopes of the project is given below.

- I. Will be used to insert and maintain customer's order and personal information in a secure system.
- II. Will be used to create diagnostic centers, admins, staffs and maintain their information efficiently.
- III. The system will be able to show and update the information of diagnostic centers, admins, staffs and customers.
- IV. This system will help the diagnostic admins and staffs to maintain the order process efficiently.
- V. Activities like inserting, updating, modification, deletion should be efficient.
- VI. On type of user shouldn't be able to access other types of user's page.

2.5 Definitions

Term	Definition
------	------------

User	Someone who interacts with the system
Database	Collection of user provided information about users and other records
Order	Arrangement or disposition of people or things in relation to each other according to a particular sequence, pattern, or method.
Website	Web portal of the system
SRS	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate.

2.6 Overall Description

This segment will give an outline of the entire system from the user's point view according to the system environment.

2.7 Project Perspective

This project is a web based system implementing client-server model. This Diagnostic Lab Reporting System provides simple mechanism for booking the test and report processing. The System mainly consist of two parts the software and the database. The website will be able to communicate with the database. Diagnostic admin will be able to create tests and manage the staffs. Staffs will be able to process orders and manage customers. Customers will be able to order tests and manage their information. A top level system admin will be able to manage Diagnostic admin and also add new programs.

2.8 Software Required

2.8.1 Software Required

- **Programming Language:** Python3.
- **Frontend Tools and Technologies:** Visual Studio Code (IDE), Adobe Photoshop, Adobe XD, HTML5, CSS3, Bootstrap4 (Responsive), JavaScript.
- **Backend Tools and Technologies:** Django Web Framework, PyCharm Professional (IDE).
- **Database:** SQLite 3 (Default)
- **Others:** Git/GitHub, Virtual Environment, Cross-Platform.

2.8.2 Hardware Required

- Most current computers and laptops have high enough specifications to be used to create a website. So, we can develop our web application using any current configuration computers or laptops.

2.9 Users of the System

The system functions may differ depending on the type of user interacting with the system. Thus, we need to explain the types of users who is capable of interacting with the system.

2.10 User Characteristics

The system is designed for 4 types of user roles in the system:

1. System Admin
2. Diagnostic Admin
3. Diagnostic Staff
4. Customer

2.11 User functionalities and their scope

As already stated, not all functions of the system should be available to all type of users in the system. It depends of the type of role assigned with the account. The availability of functions for each of the user role is shown in the table below.

User Role	Available Function
System Admin	<ol style="list-style-type: none"> 1. Can login, add, and modify own profile information. 2. Can add new diagnostic centers. 3. Can add new diagnostic admin and set/reset password. 4. Can add new diagnostic staff and set/reset password. 5. Can delete and modify everything.
Diagnostic Admin	<ol style="list-style-type: none"> 1. Can login. 2. Can add new test and category. 3. Can edit and delete test and category. 4. Can view newly added test and category. 5. Can view completed order details. 6. Can view staff status.
Diagnostic Staff	<ol style="list-style-type: none"> 1. Can login. 2. Can view newly added test and category. 3. Can approve, reject and verify order. 4. Can upload, send messages and test reports to the customer. 5. Can confirm and complete the order.
Customer	<ol style="list-style-type: none"> 1. Can register, login, and edit his/her own profile information. 2. Can view his/her own profile details. 3. Can order tests. 4. Can view his/her order status and reports. 5. Can re-payment.

2.12 Constraints

This system should be built using very efficient and high-performance web programming language. This system depends on the information of Database, so decision regarding which database to use should be taken considering the fact that data being exchanged or stored is large, and the appropriate data management system will yield efficient performance.

Though this system should be built in web technology so this system should be hosted in a highly secured and very fast responding server.

Data shouldn't be intruded by unauthorized person and should perceive the data integrity.

2.13 External Interface Requirements

This section provides a detailed description of all the inputs and outputs of the Diagnostic Lab Reporting System. It also gives a description of the hardware, software and communication interfaces.

2.13.1 User Interfaces

Each level of user will have its own interface and privilege to manage and modify information.

2.13.2 Hardware Interfaces

The system is a web-based application so the device must have an internet connection in order to be able to access the system.

2.13.3 Software Interfaces

The user's browser should be HTML5 compatible for a satisfactory user experience.

2.13.4 Communications Interfaces

The HTTP protocol will be used to facilitate communication between the client and server. Client on intranet will be using TCP/IP protocol.

2.14 Functional Requirement

This section incorporates the requirements, that indicate all the fundamental actions of the system.

2.14.1 User Class 1: System Admin

2.14.1.1 Functional Requirement 1

ID : FR1
 Title : Login.
 Rational : In order to signing in to the system.
 Description : System admins will be able to login to the system with their login credential.

2.14.1.2 Functional Requirement 2

ID : FR2
 Title : Modify own profile.
 Rational : In order to editing, updating own profile information.
 Description : The System Admin will be able to change his/her own profile information in this system.

2.14.1.3 Functional Requirement 3

ID : FR3
 Title : Create new diagnostic center.
 Rational : In order to create new diagnostic center.
 Description : The System Admin will be able to create new diagnostic center.

2.14.1.4 Functional Requirement 4

ID : FR4
 Title : Create new diagnostic admin.
 Rational : In order to create new diagnostic admin.
 Description : The System Admin will be able to create new diagnostic admin.

2.14.1.5 Functional Requirement 5

ID : FR5
 Title : Reset diagnostic admin's password.
 Rational : In order to reset diagnostic admin's password.
 Description : The System Admin will be able to reset each admin's password.

2.14.1.6 Functional Requirement 6

ID : FR6
Title : Create new diagnostic staff.
Rational : In order to create new diagnostic staff.
Description : The System Admin will be able to create new diagnostic staff.

2.14.1.7 Functional Requirement 7

ID : FR7
Title : Reset diagnostic staff's password.
Rational : In order to reset diagnostic staff's password.
Description : The System Admin will be able to reset each staff's password.

2.14.1.8 Functional Requirement 8

ID : FR8
Title : Superuser status.
Rational : In order to deleting, modifying every services.
Description : The System Admin will be able to modify everything into this system.

2.14.2 User Class 2: Diagnostic Admin

Note: All operations of a diagnostic admin are constrained to his/her respective center. A diagnostic admin can't do any operation on another diagnostic's service.

2.14.2.1 Functional Requirement 1

ID : FR1
 Title : Login.
 Rational : In order to login into the system.
 Description : Each registered diagnostic admin will be able to login to the system with their login credential.

2.14.2.2 Functional Requirement 2

ID : FR2
 Title : Add new test.
 Rational : In order to add new test.
 Description : The diagnostic admin will be able to add new test into this system.

2.14.2.3 Functional Requirement 3

ID : FR3
 Title : Add new category.
 Rational : In order to add new category.
 Description : The diagnostic admin will be able to add new category into this system.

2.14.2.4 Functional Requirement 4

ID : FR4
 Title : Edit test.
 Rational : In order to edit test.
 Description : The diagnostic admin will be able to edit test.

2.14.2.5 Functional Requirement 5

ID : FR5
 Title : Edit category.
 Rational : In order to edit category.
 Description : The diagnostic admin will be able to edit category.

2.14.2.6 Functional Requirement 6

ID : FR6

Title : Delete test.
 Rational : In order to edit test.
 Description : The diagnostic admin will be able to delete test.

2.14.2.7 Functional Requirement 7

ID : FR7
 Title : Delete category.
 Rational : In order to edit category.
 Description : The diagnostic admin will be able to delete category.

2.14.2.8 Functional Requirement 8

ID : FR8
 Title : View test.
 Rational : In order to view test.
 Description : The diagnostic admin will be able to view tests.

2.14.2.9 Functional Requirement 9

ID : FR9
 Title : View category.
 Rational : In order to view category.
 Description : The diagnostic admin will be able to view categories.

2.14.2.10 Functional Requirement 10

ID : FR10
 Title : View completed order details.
 Rational : In order to view completed order details.
 Description : The diagnostic admin will be able to view completed order details.

2.14.2.11 Functional Requirement 11

ID : FR11
 Title : View staff status.
 Rational : In order to view staff status.
 Description : The diagnostic admin will be able to view staff status.

2.14.3 User Class 3: Diagnostic Staff

Note: All operations of a diagnostic staff are constrained to his/her respective center. A diagnostic staff can't do any operation on another diagnostic's service.

2.14.3.1 Functional Requirement 1

ID : FR1
Title : Login.
Rational : In order to login into the system.
Description : Each registered diagnostic staff will be able to login to the system with their login credential.

2.14.3.2 Functional Requirement 2

ID : FR2
Title : View test.
Rational : In order to view test.
Description : The diagnostic staff will be able to view tests.

2.14.3.3 Functional Requirement 3

ID : FR3
Title : View category.
Rational : In order to view category.
Description : The diagnostic staff will be able to view categories.

2.14.3.4 Functional Requirement 4

ID : FR4
Title : Approve orders.
Rational : In order to approve orders.
Description : The diagnostic staff will be able to approve orders.

2.14.3.5 Functional Requirement 5

ID : FR5
Title : Reject orders.
Rational : In order to reject orders.
Description : The diagnostic staff will be able to reject orders.

2.14.3.6 Functional Requirement 6

ID : FR6
Title : Verify orders.
Rational : In order to verify orders.
Description : The diagnostic staff will be able to verify orders.

2.14.3.7 Functional Requirement 7

ID : FR7

Title : Upload report.
 Rational : In order to upload report.
 Description : The diagnostic staff will be able to upload report.

2.14.3.8 Functional Requirement 8

ID : FR8
 Title : Send message.
 Rational : In order to send message.
 Description : The diagnostic staff will be able to send message to the customer.

2.14.4 User Class 4: Customer

2.14.4.1 Functional Requirement 1

ID : FR1
 Title : Registration.
 Rational : In order to sign up to the system.
 Description : Customer will be able to sign up to the system with their personal information.

2.14.4.2 Functional Requirement 2

ID : FR2
 Title : Login.
 Rational : In order to login into the system.
 Description : Each registered customer will be able to login to the system with their login credential.

2.14.4.3 Functional Requirement 3

ID : FR3
 Title : Edit own profile.
 Rational : In order to editing own profile information.
 Description : Each registered customer will be able to edit his/her own profile information.

2.14.4.4 Functional Requirement 4

ID : FR4
 Title : View profile details.
 Rational : In order to view profile details.

Description : Each registered customer will be able to view his/her own profile details.

2.14.4.5 Functional Requirement 5

ID : FR5
Title : Order test.
Rational : In order to make a test order.
Description : Each registered customer will be able to order tests.

2.14.4.6 Functional Requirement 6

ID : FR6
Title : View order status.
Rational : In order to view order status.
Description : Each registered customer will be able to view his/her own order status.

2.14.4.7 Functional Requirement 7

ID : FR7
Title : View reports.
Rational : In order to view reports.
Description : Each registered customer will be able to view his/her own report table.

2.14.4.8 Functional Requirement 8

ID : FR8
Title : Make re-payment.
Rational : In order to make re-payment.
Description : Each registered customer will be able to re-pay his/her due payment from report table.

3 Chapter 3: System Design & Implementation

3.1 Software Development Method

In software engineering, a software development process is the process of dividing software development work into distinct phases to improve design, product management, and project management. It is also known as a software development life cycle. The methodology may include the pre-definition of specific deliverables and artifacts that are created and completed by a project team to develop or maintain an application.

Among various software development methodologies, Agile development process was used to design and develop this system. Thus, we followed these four core values of this process while approaching through the project:

Agile methods advocate four core value

I. **Team** (*Individuals and interactions over processes and tools*):

In agile view, the team is more important than the tools (structural or control) or operating procedures. It is better to have a solid team and communicating, consisting of developers (possibly varying levels), rather than a team of experts each operating in isolation. Communication is a fundamental concept. As we worked as a team in the project, we ensured concrete communication between us (the team members).

II. **Application** (*Working software over comprehensive documentation*):

It is vital that the application works. The rest, including the technical documentation is invaluable but not an end in itself. Accurate documentation is useful as a means of communication. Documentation represents a significant workload, but can nevertheless be harmful if it is outdated. It is better to comment on the code itself extensively, especially to transfer skills within the team (it goes back to the importance of communication). Thus, we focused if the system is functioning properly, rather than focusing on the documentation.

III. **Collaboration** (*Customer collaboration over contract negotiation*):

The client should be involved in development. One cannot simply negotiate a contract at the beginning of the project, and then neglect the customer requirements. The client needs to work with the team and provide a continuous account made on the suitability of the software with expectations. Thus, we always had to keep in mind about the user requirements on any approach through designing and developing the system.

IV. **Change** (*Responding to change over following a plan*):

Initial planning and structure of the software must be flexible to allow changes in customer demand throughout the project. The first deliveries of the software will often result in change requests. The software under development was changed multiple times, added with new features throughout the development.

3.2 Work Process

First of all, we need to design the database structure, as the whole system is supposed to be depended on the database. Then the Web UI and the backend functionality development will run parallelly. Initially the database will consist only one user account: The **System Admin**, all other data will be added as necessary during development for checking if it works. With a user database, it provides the privileged users with a clean and easy-to-use but effective Web Interface to use the system efficiently.

3.3 State Transition Diagram (STD)

A state transition diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. The STD for this project is shown below:

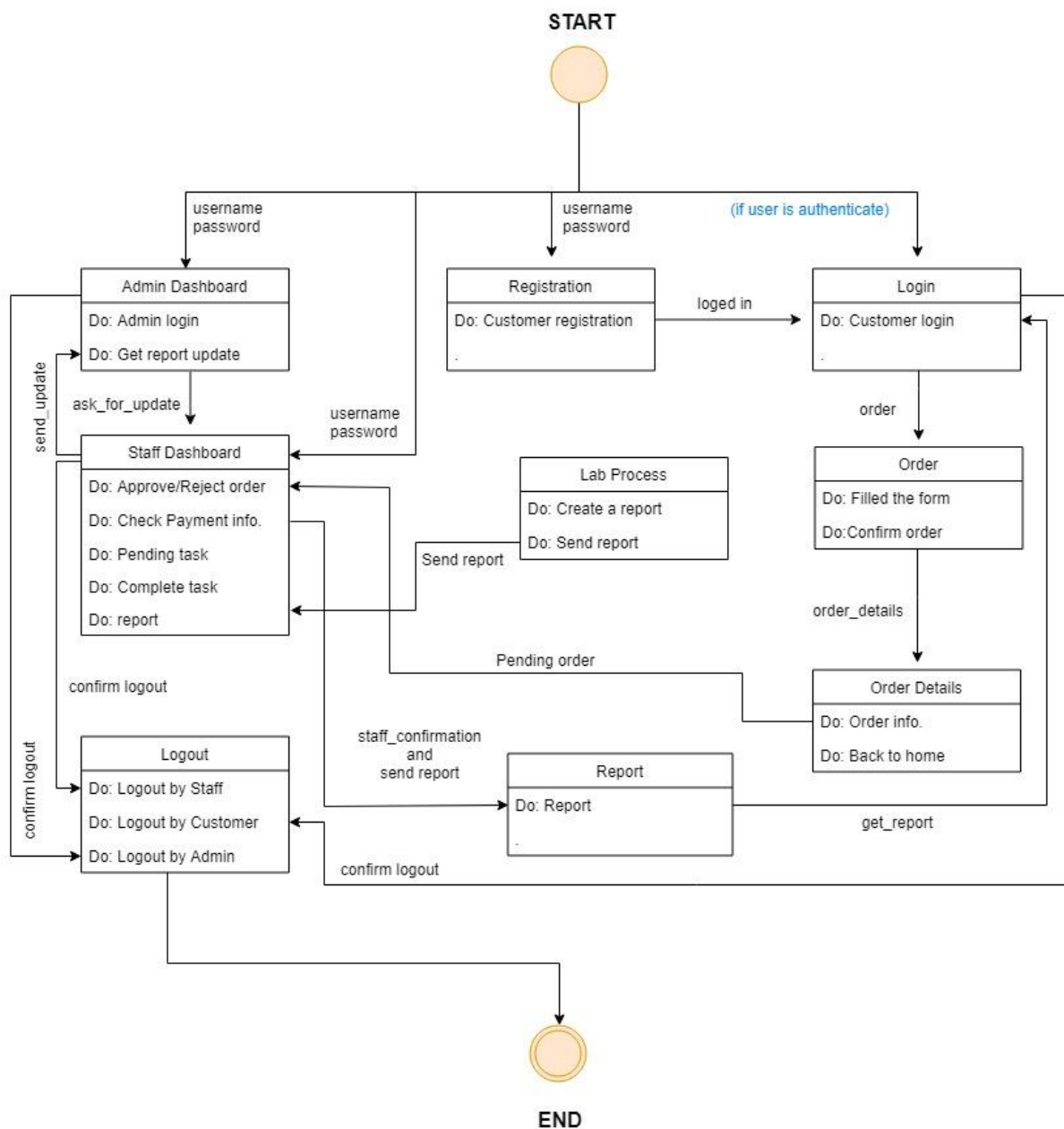


Fig: State Transition Diagram for Diagnostic Lab Reporting System

3.4 Data Flow Diagram (DFD)

A data-flow diagram is a way of representing a flow of a data of a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. The final DFD for this project is shown below:

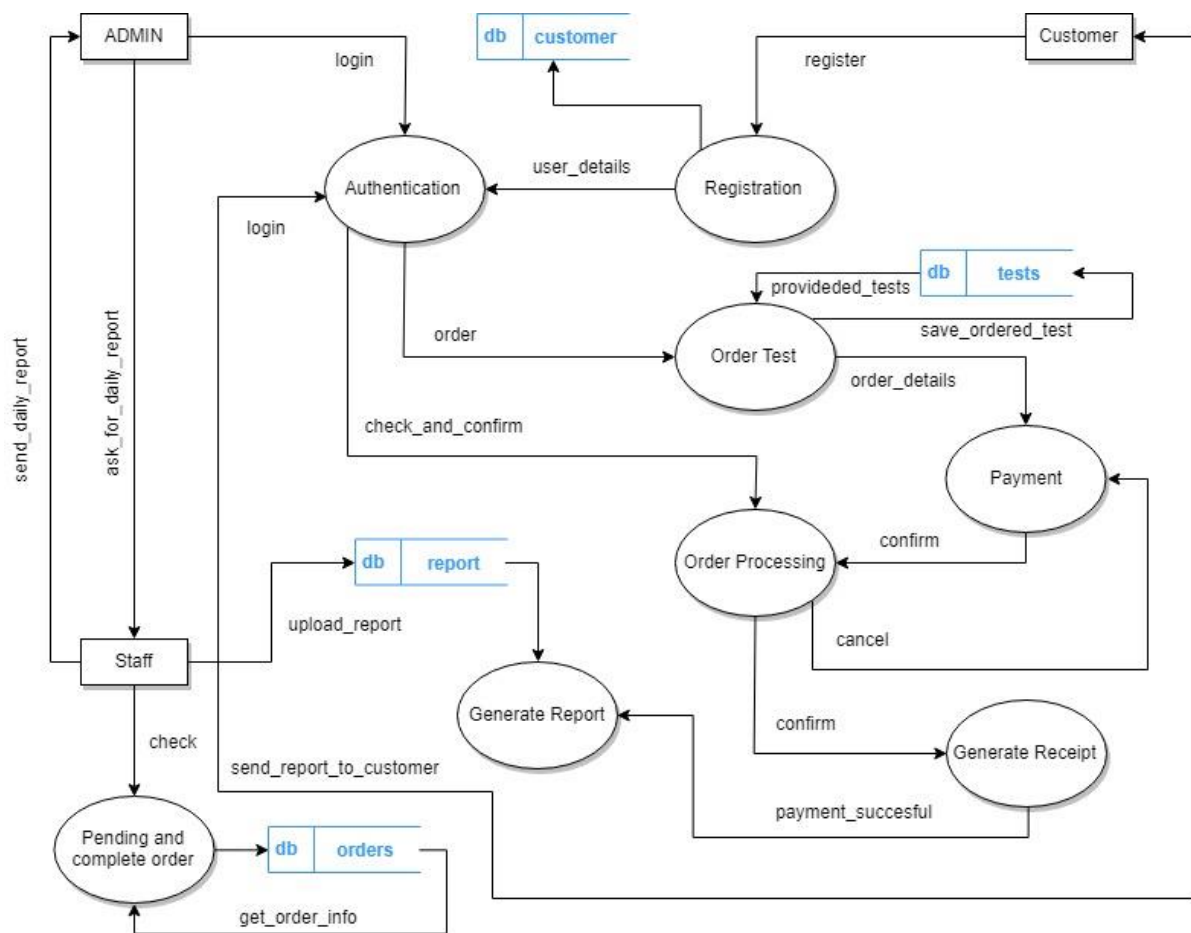


Fig: Data Flow Diagram for Diagnostic Lab Reporting System

3.5 Entity Relationship Diagram (ERD)

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how 'entities' such as people, objects or concepts relate to each other within a system. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs. The final ERD for this project is shown below:

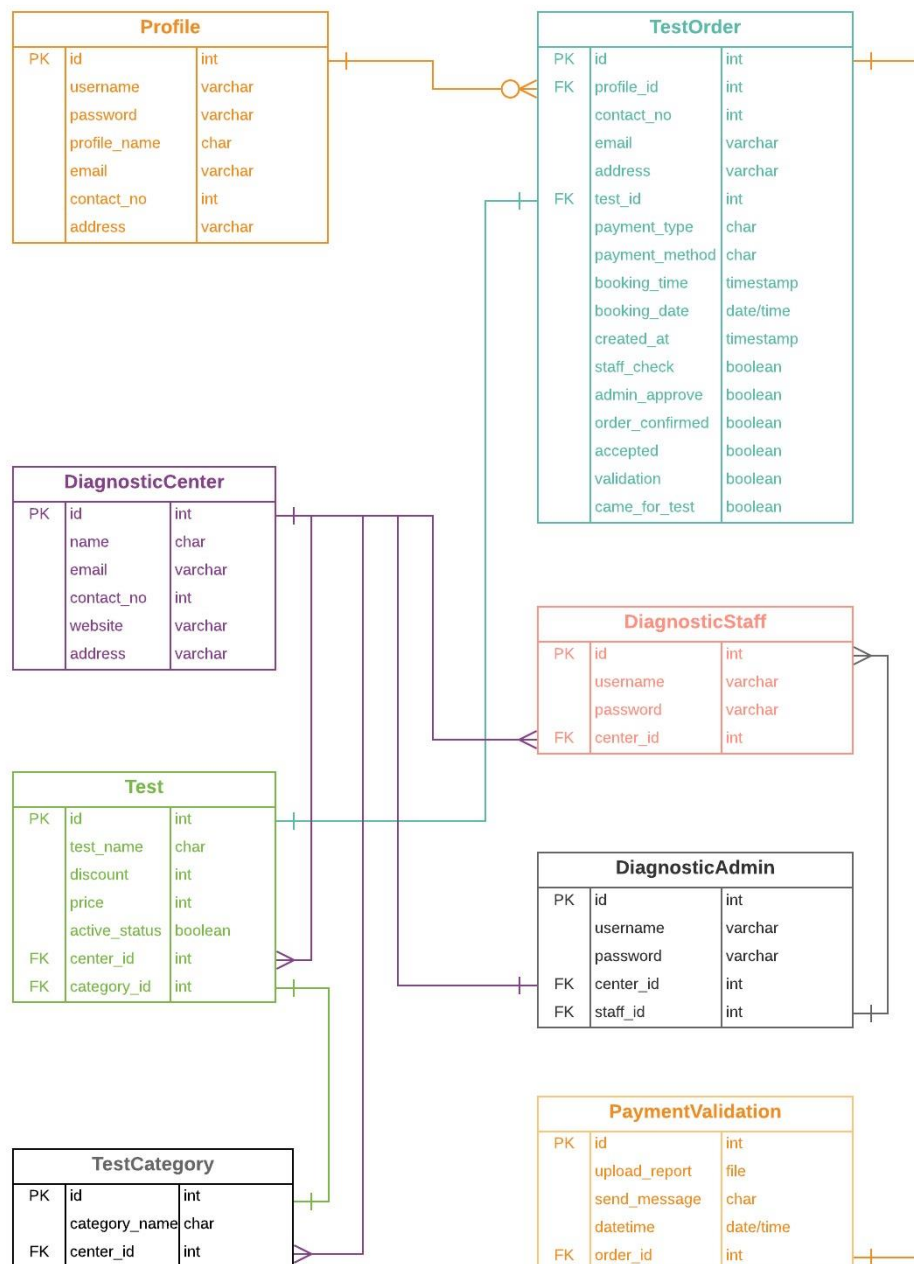


Fig:Entity Relationship Diagram for Diagnostic Lab Reporting System

3.6 Database Design

3.6.1 List of Tables

Eight tables with following names were added to the database after designing. Their fields are explained in later topics.

Table Name	Purpose/Description
Profile	Holds the information of all customers
Diagnostic Center	Holds the information of all diagnostic centers
Diagnostic Admin	Holds the information of all diagnostic admins
Diagnostic Staff	Holds the information of all diagnostic staffs
Test	Holds the information of all tests
Test Category	Holds the information of all test categories
Test Order	Holds the information of all test orders
Payment Validation	Holds the information of all completed tests

3.6.1.1 Table structure image for 'Profile'

	id	image	user_id	profile_name	address	admin	staff	contact_no
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	12	default.jpg	12	Siyam Google	Dhaka	0	0	01516113959
2	13	default.jpg	13	default	NULL	0	0	NULL
3	28	profile_pics/image.png	28	Siyam Al Galib	Mohammadpur, Dhaka	0	0	+880 1516113959
4	29	profile_pics/jh.jpg	29	Nazmul Hosssain	chandpur	0	0	NULL

Fig: Profile table structure

3.6.1.2 Table structure image for ‘Diagnostic Center’

	id	name	image	contact_no	address	website	email
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	City Hospital Ltd.	diagnostic_center...	02-8146166	1/8, Block-E, Lalmatia, Satmosjid Road, Mohammadpur Dhaka, Bangladesh	www.cityhospitalbd.com	info@cityhospitalbd.com
2	2	Popular Diagnostic Centre Ltd.	diagnostic_center...	+880 9613787801	HOUSE # 16 Rd No. 2, Dhaka 1205	www.populardiagnostic.com	info@populardiagnostic.com
3	3	Labaid Diagnostic Centre	diagnostic_center...	10606, 02-58610793-8	Plot # 01 & 03, Road # 04, Dhanmondi Dhaka 1205, Bangladesh	www.labaidgroup.com	info@labaidgroup.com
4	4	Padma Diagnostic Centre Ltd.	diagnostic_center...	+88 09617444222	245/2 New Circular Road, Malibagh Dhaka, Bangladesh	www.padma-bd.com	info@padma-bd.com
5	5	Center 1	default_center.jpg	123456789	Dhaka	www.center-bd.com	info@center-bd.com
6	6	Center 2	default_center.jpg	12456791	Dhaka	www.center-bd.com	info@center-bd.com
7	7	Center 3	default_center.jpg	01516113959	Dhaka	www.center-bd.com	info@center-bd.com
8	8	Center 4	default_center.jpg	01755223344	Dhaka	www.center-bd.com	info@center-bd.com
9	9	Center 5	default_center.jpg	12456791	Dhaka	www.center-bd.com	info@center-bd.com
10	10	Center 6	default_center.jpg	01899767789	Dhaka	www.center-bd.com	info@center-bd.com
11	11	Center 7	default_center.jpg	01715223344	Dhaka	www.center-bd.com	info@center-bd.com

Fig: Diagnostic Center table structure

3.6.1.3 Table structure image for ‘Diagnostic Admin’

	id	username	password	center_id	admin
	Filter	Filter	Filter	Filter	Filter
1	1	city_admin_1	cityad1-123	1	1
2	2	popular_admin_1	popad1-789	2	1
3	3	padma_admin_1	padad1-456	4	1
4	4	labaid_admin_1	labad1-321	3	1

Fig: Diagnostic Admin table structure

3.6.1.4 Table structure image for ‘Diagnostic Staff’

	id	username	password	center_id	admin
	Filter	Filter	Filter	Filter	Filter
1	1	city_staff_1	cityst1-123	1	0
2	2	popular_staff_1	popst1-789	2	0
3	3	padma_staff_1	padst1-456	4	0
4	4	padma_staff_2	padst2-456	4	0
5	5	popular_staff_2	popst2-789	2	0
6	6	city_staff_2	cityst2-123	1	0
7	7	labaid_staff_1	labst1-321	3	0
8	8	labaid_staff_2	labst2-321	3	0

Fig: Diagnostic Staff table structure

3.6.1.5 Table structure image for 'Test'

	id	test_name	image	discount	active_status	category_id	center_id	price
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Blood Test (B...	default_test.jpg	0	Available	1	3	200
2	2	X-Ray	default_test.jpg	0	Available	3	1	300
3	3	Sugar	default_test.jpg	50	Available	6	4	250
4	4	Diabetics	default_test.jpg	0	Available	6	1	500
5	5	Pressure	default_test.jpg	100	Available	6	2	600
6	6	New Test	default_test.jpg	0	Available	4	3	100
7	7	New Test 2	default_test.jpg	0	Available	5	3	100
8	8	New Test 3	default_test.jpg	0	Available	5	3	100
9	9	Demo	default_test.jpg	0	Available	2	3	500
10	10	Demo 1	default_test.jpg	50	Available	5	3	500
11	11	Demo 1	default_test.jpg	500	Available	5	3	500
12	12	Demo 2	default_test.jpg	100	Available	2	3	600
13	13	Demo 3	default_test.jpg	0	Available	1	3	400
14	14	Demo 4	default_test.jpg	100	Available	2	3	400
15	15	Demo 5	default_test.jpg	150	Available	3	3	650
16	16	Demo 6	default_test.jpg	80	Available	5	3	400
17	17	New Test	default_test.jpg	0	Available	1	3	100
18	18	Test 2	default_test.jpg	100	Available	1	3	600

Fig: Test table structure

3.6.1.6 Table structure image for 'Test Category'

	id	category_name	center_id
	Filter	Filter	Filter
1	1	Blood Test	3
2	2	ECG	4
3	3	X-Ray	2
4	4	ECG	1
5	5	CT-Scan	3
6	6	Others	3
7	8	Lorem Ipsum 2	3
8	9	Lorem Ipsum 3	3
9	10	Lorem Ipsum 4	3
10	11	Lorem Ipsum 5	3
11	13	joy	3
12	15	TestCat	3

Fig: Test Category table structure

3.6.1.7 Table structure image for 'Test Order'

	id	client_info_id	test_info_id	order_created_at	admin_approve	staff_check	accepted	contact_no	address	email	booked_date	booked_time_slot	payment_method	validation
		Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	52	28	16	23:07:54.156344	0	1	0	01737202004	Dhaka	client1@email.com	2019-01-12	10:00 AM - 2:00 PM	BKASH	0
2	53	28	16	23:07:54.125094	0	1	1	01737202004	Dhaka	client1@email.com	2019-07-30	10:00 AM - 2:00 PM	ON SPOT	0
3	54	28	12	03:47:42.238072	0	1	1	01737202004	Dhaka	client1@email.com	2020-10-10	3:00 PM - 7:00 PM	BKASH	1
4	55	28	11	22:06:00.225794	0	1	1	01737202004	Dhaka	client1@email.com	2025-10-10	3:00 PM - 7:00 PM	CREDIT CARD	1
5	56	28	18	23:07:54.046417	0	1	1	01737202004	Dhaka	client1@email.com	2024-10-12	3:00 PM - 7:00 PM	CREDIT CARD	0
6	57	28	15	23:07:54.015173	0	1	1	01737202004	Dhaka	client1@email.com	2020-09-09	3:00 PM - 7:00 PM	BKASH	0
7	58	28	12	23:07:53.983926	0	1	1	01737202004	Dhaka	client1@email.com	2025-12-17	3:00 PM - 7:00 PM	ON SPOT	0
8	59	28	14	23:07:53.952680	0	1	1	01737202004	Dhaka	client1@email.com	2026-10-10	10:00 AM - 2:00 PM	BKASH	1
9	60	28	16	23:07:53.921435	0	1	1	01737202004	Mohammadpur, Dhaka	client1@email.com	2019-08-13	10:00 AM - 2:00 PM	BKASH	0
10	61	28	18	23:07:53.890182	0	1	1	01737202004	Mohammadpur, Dhaka	client1@email.com	2019-09-05	3:00 PM - 7:00 PM	CREDIT CARD	0
11	62	28	17	23:07:53.858934	0	1	1	01737202004	Mohammadpur, Dhaka	client1@email.com	2027-09-11	3:00 PM - 7:00 PM	BKASH	0
12	63	28	11	23:07:53.843301	0	1	0	01737202004	Mohammadpur, Dhaka	client1@email.com	2027-11-11	3:00 PM - 7:00 PM	CREDIT CARD	0
13	64	28	19	23:07:53.812050	0	1	1	01737202004	Mohammadpur, Dhaka	client1@email.com	2027-09-11	3:00 PM - 7:00 PM	CREDIT CARD	0
14	65	28	19	23:07:53.780797	0	1	0	01737202004	Mohammadpur, Dhaka	client1@email.com	2027-12-14	3:00 PM - 7:00 PM	CREDIT CARD	0
15	68	28	19	23:07:53.687063	0	1	1	01737202004	Mohammadpur, Dhaka	client1@email.com	2019-12-12	3:00 PM - 7:00 PM	CREDIT CARD	0
16	69	28	18	00:39:15.174608	0	1	0	01737202004	Mohammadpur, Dhaka	client1@email.com	2019-10-13	3:00 PM - 7:00 PM	CREDIT CARD	0
17	70	28	20	23:07:53.624544	0	1	1	01737202004	Mohammadpur, Dhaka	client1@email.com	2025-10-11	3:00 PM - 7:00 PM	CREDIT CARD	0

payment_method	validation	order_confirmed	came_for_test	payment_type
Filter	Filter	Filter	Filter	Filter
BKASH	0	1	0	Full Payment
ON SPOT	0	1	0	Half Payment
BKASH	1	1	0	Full Payment
CREDIT CARD	1	1	0	Full Payment
CREDIT CARD	0	1	0	Half Payment
BKASH	0	1	0	Half Payment
ON SPOT	0	1	0	Half Payment
BKASH	1	1	0	Full Payment
BKASH	0	1	0	Half Payment
CREDIT CARD	0	1	0	Half Payment
BKASH	0	1	0	Half Payment
CREDIT CARD	0	1	0	Half Payment
CREDIT CARD	0	1	0	Half Payment
CREDIT CARD	0	1	0	Half Payment
CREDIT CARD	0	1	0	Half Payment
CREDIT CARD	0	1	0	Half Payment
CREDIT CARD	0	1	0	Half Payment

Fig: Test Order table structures (single image but captured in 2 parts because of size)

3.6.1.8 Table structure image for ‘Payment Validation’

	id	datetime	pproved_order_i	send_message	upload_report
	Filter	Filter	Filter	Filter	Filter
1	3	2019-07-19 13:01:50.067832	55	Complete	reports/Convocation-Registration-Form_QQv97Vy.pdf
2	2	2019-07-19 12:53:05.485372	59	Complete	reports/Convocation-Registration-Form_Q9BZPNR.pdf
3	4	2019-07-19 13:20:11.388370	72	Complete	reports/Convocation-Registration-Form_jpFqmuJ.pdf
4	5	2019-07-19 17:00:47.826715	73	Complete	reports_PDF/Convocation-Registration-Form_3Xmongu.pdf
5	39	2019-08-02 17:35:27.198297	76	Complete	reports_PDF/report_demo_6pxdn6R.pdf
6	6	2019-07-19 17:17:35.831522	77	Complete	reports_PDF/report_demo.pdf
7	25	2019-07-27 19:32:15.795563	78	Complete	reports_PDF/report_demo_vg6xAhB.pdf
8	7	2019-07-20 17:08:36.844252	79	Complete	reports_PDF/report_demo_qW8pigb.pdf
9	8	2019-07-21 18:33:31.492898	80	Complete	reports_PDF/report_demo_F80P5rK.pdf
10	23	2019-07-27 18:50:44.131988	84	Complete	reports_PDF/report_demo_mN9Tthm.pdf
11	33	2019-07-28 16:50:04.593712	88	Complete	reports_PDF/report_demo_FXjUzHK.pdf
12	30	2019-07-28 16:29:52.141392	89	Complete	reports_PDF/report_demo_i4tSH0L.pdf
13	61	2019-08-27 16:22:55.461322	96	Please Complete Your Full Payment for REPORT	
14	60	2019-08-27 16:22:46.000432	98	Please Complete Your Full Payment for REPORT	
15	59	2019-08-27 16:22:34.366745	100	Please Complete Your Full Payment for REPORT	
16	58	2019-08-27 16:22:25.068537	102	Complete	reports_PDF/report_demo_yag2qzT.pdf
17	45	2019-08-06 15:33:18.512496	103	Complete	reports_PDF/report_demo_yMGEICN.pdf
18	57	2019-08-27 16:22:18.308191	104	Please Complete Your Full Payment for REPORT	

Fig: Payment Validation table structure

4 Chapter 4: Implementation

4.1 Development Environment

This project is developed as a Web-Application. The web interface was designed using HTML, CSS, Bootstrap and JavaScript. SQLite3 was used as database. The application is developed using Python3 and Django Web Framework. IDE used was PyCharm Professional and Visual Studio Code. For development and testing browsers used were Mozilla Firefox, Google Chrome, and Opera.

4.1.1 Common Issues

We faced several issues during developing the software. The notable ones are mentioned below for future avoidance.

- 4.1.1.1 a. Database Design:** The database needed to be changed frequently during the whole development timeline to give it a perfect shape. The final database is nowhere near the first version. For example, we were developing the system in Agile method so after giving the database a shape, sometimes user requirements were changed so we had to make huge amount of change in the database. Sometimes such situation arose that we developed the database in one manner and when we tried to integrate it with the whole system we felt that efficiency is lost because of the arrangement of the database, so we had to make changes in the database.
- 4.1.1.2 b. Making a User-Friendly UX:** To make the UX as friendly as possible, we had to trim many complications. We go through a lot of testing and which UX is more efficient to use, we tested the application with some testers and from their feedback we come to the first version of the application.
- 4.1.1.3 c. Designing the Webpages:** To give it a nice and gorgeous look, we had to spend good amount of time in building the interface. We had to go through a lot of iteration to make a perfect Bluish Theme for the whole web application.

4.2 Web Interface

Some of the screenshots of different user's user interface in the system is given below:

4.2.1 System Admin Login:

Site Administration | ODLRS

Username:

Password:

4.2.2 System Admin Dashboard:

Site Administration | ODLRS

WELCOME, CLIENT1. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Site administration

ACCOUNTS

Email addresses

+ Add

Change

CUSTOM_USERS

Customer Profiles

+ Add

Change

REPORT_PROCESSING

Payment validations

+ Add

Change

SOCIAL_ACCOUNTS

Social accounts

+ Add

Change

Social application tokens

+ Add

Change

Social applications

+ Add

Change

AUTHENTICATION AND AUTHORIZATION

Users

+ Add

Change

DIAGNOSTIC_CENTERS

Diagnostic Admins

+ Add

Change

Diagnostic Centers

+ Add

Change

Diagnostic Staffs

+ Add

Change

SITES

Sites

+ Add

Change

TESTS

Test Categories

+ Add

Change

Test Orders

+ Add

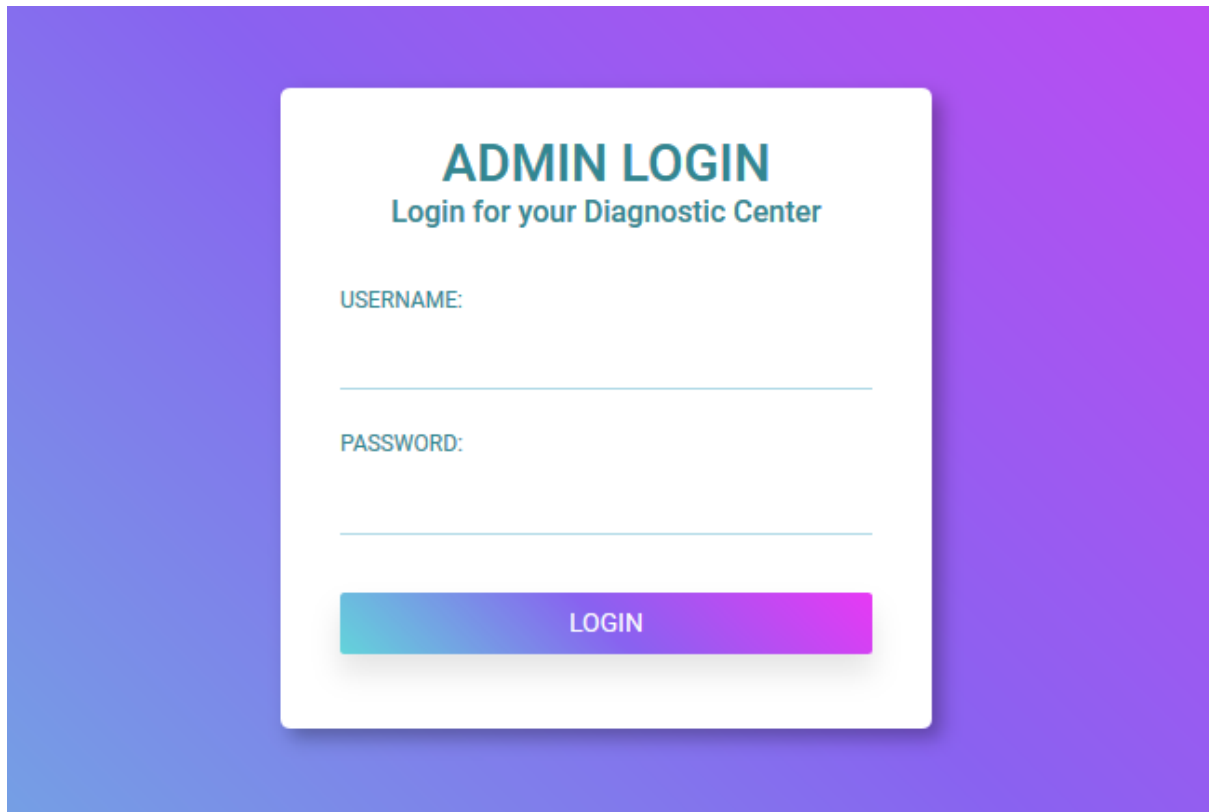
Change

Tests

+ Add

Change

4.2.3 Diagnostic Admin Login:



The image shows a login form titled "ADMIN LOGIN" with the subtitle "Login for your Diagnostic Center". It is set against a purple-to-blue gradient background. The form is a white card with a shadow. It contains two input fields: "USERNAME:" and "PASSWORD:", each with a light blue underline. Below these fields is a large, rounded rectangular button with a blue-to-purple gradient and the text "LOGIN" in white capital letters.

4.2.4 Diagnostic Admin Dashboard:

ODRLS

[Add New Category](#)[Add New Test](#)[Recent Categories](#)[Recent Tests](#)[Logout](#)

Center Name
Labaid Diagnostic Centre

Admin Name
labaid_admin_1

Staff Name:
labaid_staff_1, labaid_staff_2,

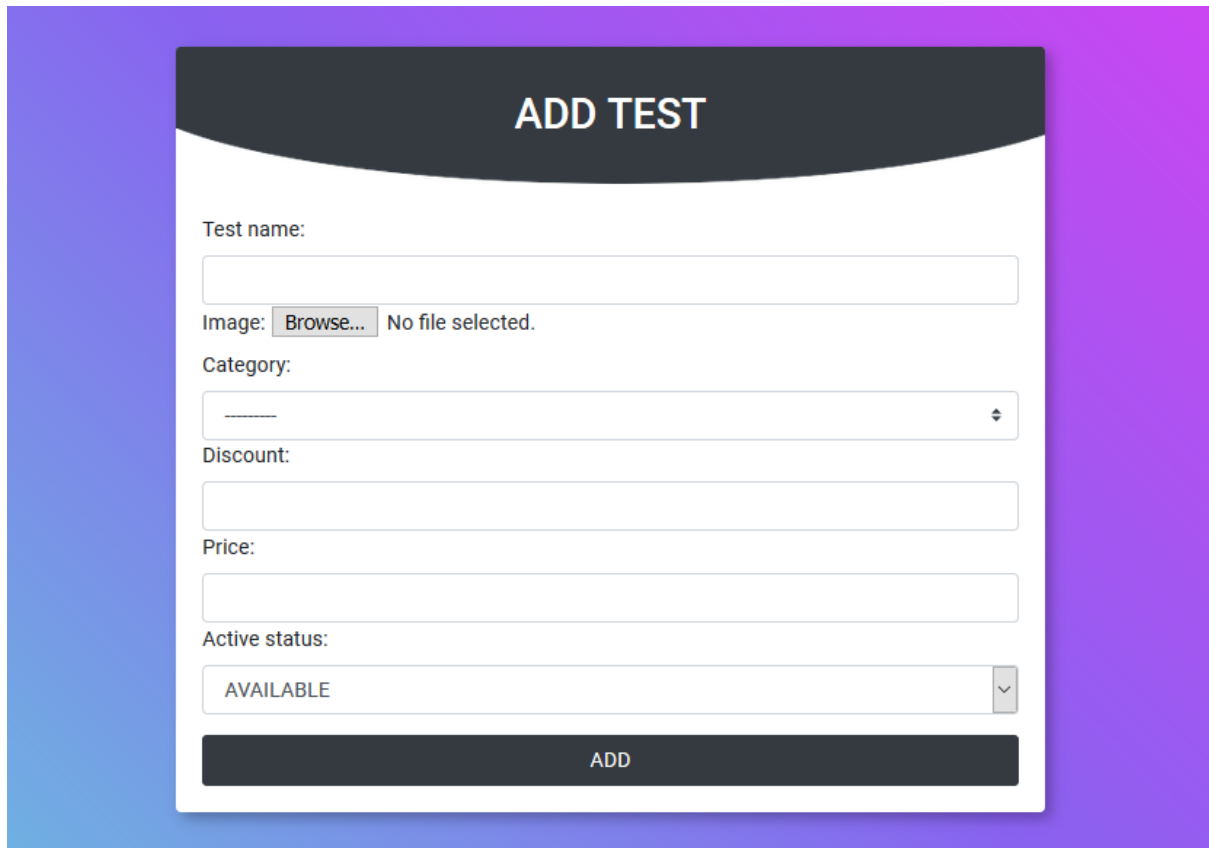
COMPLETED ORDERS

NO.	Orders	DateTime	Reports	Status
1	Final Test 4	Aug. 31, 2019, 8:04 p.m.	reports_PDF/D7XFKRXW0AAOC3_.jpg	successfully done
2	Final Test 4	Aug. 31, 2019, 8:02 p.m.		Please make a full payment for your report
3	Final Test 1	Aug. 31, 2019, 7:59 p.m.		Please make a full payment for your report
4	Final Test 1	Aug. 31, 2019, 7:58 p.m.		Please make a full payment for your report
5	joy	Aug. 31, 2019, 1 p.m.	reports_PDF/_working_SWxKYe7.pdf	Payment Done
6	joy	Aug. 31, 2019, 12:59 p.m.		Please make a full payment
7	Testing 1 Check	Aug. 31, 2019, 12:55 p.m.	reports_PDF/_working_KvLa9fu.pdf	Payment Done
8	Final Test 2	Aug. 31, 2019, 12:55 p.m.		Please make a full payment
9	Final Test 2	Aug. 31, 2019, 12:53 p.m.		Please make a full payment
10	Test Name 13 Edited	Aug. 31, 2019, 12:48 p.m.		Please make a full payment for your report

Page 1 of 5. >

Developed By Galib and Sohanur

4.2.5 Diagnostic Admin Add New Test:

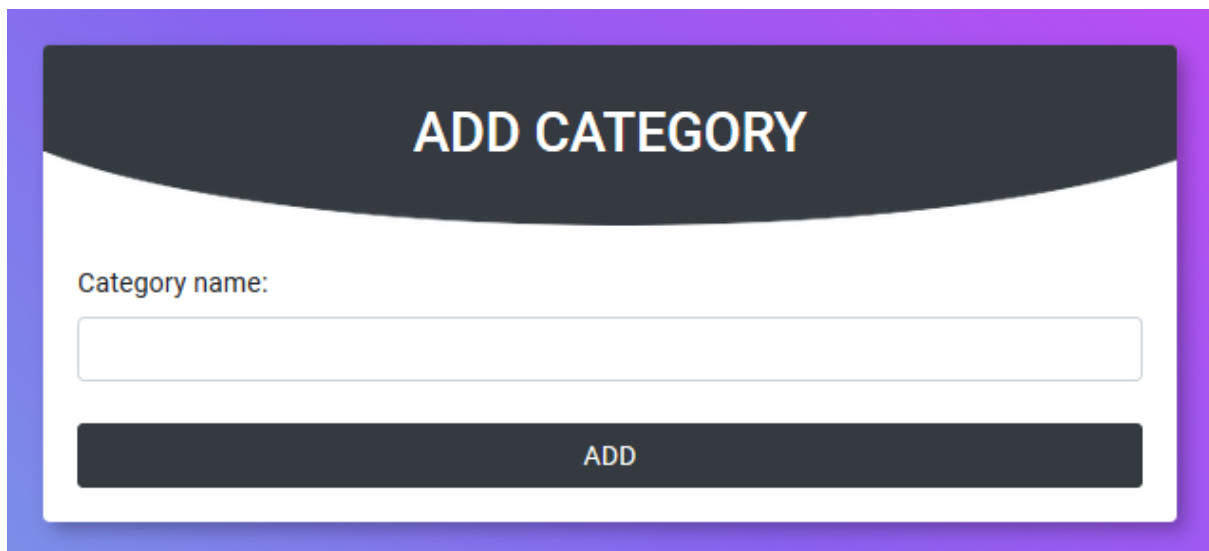


The screenshot shows a form titled "ADD TEST" with a dark header. The form fields are as follows:

- Test name:** A text input field.
- Image:** A button labeled "Browse..." followed by the text "No file selected."
- Category:** A dropdown menu with a downward arrow.
- Discount:** A text input field.
- Price:** A text input field.
- Active status:** A dropdown menu with "AVAILABLE" selected and a downward arrow.

At the bottom of the form is a dark button labeled "ADD".

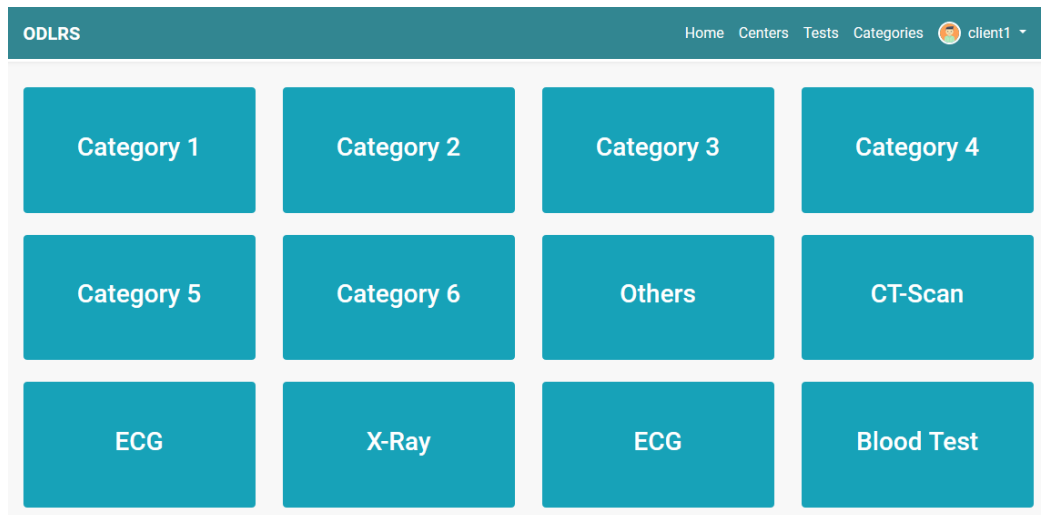
4.2.6 Diagnostic Admin Add New Category:



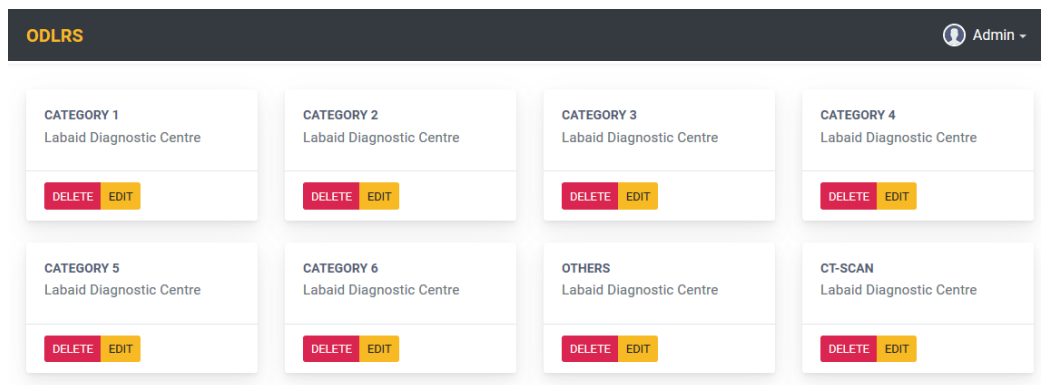
The screenshot shows a form titled "ADD CATEGORY" with a dark header. The form fields are as follows:

- Category name:** A text input field.

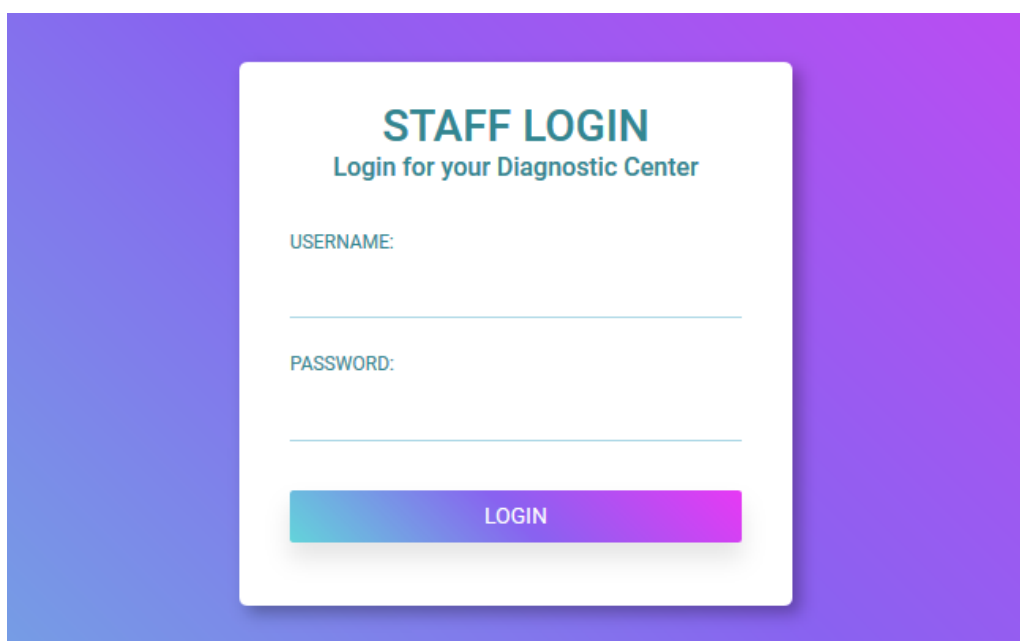
At the bottom of the form is a dark button labeled "ADD".



4.2.7 Diagnostic Admin Edit and Delete Contents:



4.2.8 Diagnostic Staff Login:



4.2.9 Diagnostic Staff Dashboard:

ODRLS

Staff Dashboard

Recent Categories

Recent Tests

Logout

Center Name
Labaid Diagnostic Centre

Staff Name
labaid_staff_1

Reported To:
labaid_admin_1

Pending Orders 4

Confirmed Orders 65

HALF Paid Orders 22

FULL Paid Orders 43

Came for Tests 65

Completed Orders 35

STEP-0: PENDING ORDER...

NO.	Orders
1	Final Test 9
2	Final Test 8
3	Final Test 7
4	Final Test 6

Page 1 of 1.

STEP-1: CONFIRMING...

NO.	Orders
1	Final Test 12 Order Confirmed
2	Final Test 11 Order Confirmed
3	Joy Order Confirmed
4	Final Test 2 Order Confirmed
Testing 1 Check	

STEP-2: CAME FOR TEST.....

NO.	Orders
1	Final Test 12 Came for Test: True
2	Final Test 11 Came for Test: True
3	Joy Came for Test: True
4	Final Test 2 Came for Test: True
Testing 1 Check	

STEP-3: VERIFYING BEFORE REPORT DELIVERY...

NO.	Orders
1	Final Test 12 STEP-2: True Payment Status: Full Payment Verified: True
2	Final Test 11 STEP-2: True Payment Status: Full Payment Verified: True
3	Joy STEP-2: True

STEP-4: FOR HALF PAYMENTS SENDING MESSAGE FOR COMPLETE PAYMENT

NO.	Orders
1	Mild Stroke STEP-3: True Payment Status: Half Payment Message Sent
2	Test Name 100 STEP-3: True Payment Status: Half Payment Message Sent
3	Test Name 12 Edited STEP-3: True

STEP-5 + 6: FOR FULL PAYMENTS REPORT DELIVERY...

NO.	Orders
1	Final Test 12 STEP-4: True Payment Status: Full Payment Report Delivered
2	Final Test 11 STEP-4: True Payment Status: Full Payment Report Delivered
3	Joy STEP-4: True

STEP-7: PAYMENT COMPLETE AND REPORT SENT COMPLETED ORDERS

NO.	Orders	DateTime	Reports	Status
1	Final Test 11	Aug. 31, 2019, 5:58 p.m.	reports_PDF/report_demo_OHQcv6Q.pdf	Complete
2	Final Test 12	Aug. 31, 2019, 5:56 p.m.	reports_PDF/report_demo_FTdlIYl.pdf	Complete
3	Final Test 11	Aug. 31, 2019, 5:55 p.m.		Please Complete Your Full Payment for REPORT
4	Joy	Aug. 31, 2019, 1 p.m.	reports_PDF/_working_SWxKYe7.pdf	Payment Done
5	Joy	Aug. 31, 2019, 12:59 p.m.		Please make a full payment
6	Testing 1 Check	Aug. 31, 2019, 12:55 p.m.	reports_PDF/_working_KvLa9fu.pdf	Payment Done
7	Final Test 2	Aug. 31, 2019, 12:55 p.m.		Please make a full payment

STEP-8: PAYMENT NOT COMPLETE AND REPORT NOT SENT PENDING ORDERS

NO.	Orders	DateTime	Reports	Status
1	Test Name 9 Edited	Aug. 27, 2019, 10:22 p.m.		Please Complete Your Full Payment for REPORT
2	Test Name 7	Aug. 27, 2019, 10:22 p.m.		Please Complete Your Full Payment for REPORT
3	Test Name 9 Edited	Aug. 27, 2019, 10:22 p.m.		Please Complete Your Full Payment for REPORT
4	Test Name 12 Edited	Aug. 27, 2019, 10:22 p.m.		Please Complete Your Full Payment for REPORT

Page 1 of 1.

Developed By Galib and Sohanur

4.2.10 Diagnostic Staff Dashboard Pending Orders:

STEP-0: PENDING ORDER...	
NO.	Orders
1	Final Test 9
2	Final Test 8
3	Final Test 7
4	Final Test 6
Page 1 of 1.	

4.2.11 Diagnostic Staff Dashboard Approve or Reject Orders:

Confirmation Voucher

Name	Test Name
Siyam Al Galib	Final Test 9
Email	Center
client1@email.com	Labaid Diagnostic Centre
Booked Time Slot	Booked Date
None	Sept. 7, 2019
Total Price	Payment By
500Tk.	BKASH
Payment Status	Contact
Half Payment	+880 1516113959
Address	
Mohammadpur, Dhaka	

Approve

Reject

4.2.12 Diagnostic Staff Dashboard Patient Came or Not:

Confirmation Voucher

Order Approved

<p>Name Siyam Al Galib</p> <hr/> <p>Email client1@email.com</p> <hr/> <p>Booked Time Slot 3:00 PM - 7:00 PM</p> <hr/> <p>Total Price 500Tk.</p> <hr/> <p>Payment Status Full Payment</p> <hr/> <p>Address Mohammadpur, Dhaka</p> <hr/>	<p>Test Name Test Name 100</p> <hr/> <p>Center Labaid Diagnostic Centre</p> <hr/> <p>Booked Date May 8, 2019</p> <hr/> <p>Payment By BKASH</p> <hr/> <p>Contact 00 123 345 567</p> <hr/>
--	---

Came For Test

4.2.13 Diagnostic Staff Dashboard Patient Came or Not Status:

STEP-1: CONFIRMING...	
NO.	Orders
1	<div>Test Name 100</div> <div>Order Confirmed</div>
2	<div>Test Name 100</div> <div>Order Confirmed</div>
3	<div>Test Name 12 Edited</div> <div>Order Confirmed</div>
4	<div>Testing 1 Check</div> <div>Order Confirmed</div>
	Test Name 100

4.2.14 Diagnostic Staff Dashboard Payment Verification:

STEP-3: VERIFYING BEFORE REPORT DELIVERY...

NO. Orders

1	Test Name 100 STEP-2: True Payment Status: Full Payment Verified: True
3	Test Name 12 Edited STEP-2: True Payment Status: Full Payment Verified: True

4.2.15 Diagnostic Staff Dashboard Upload Report and Sending Message:

STEP-4: FOR HALF PAYMENTS SENDING MESSAGE FOR COMPLETE PAYMENT

NO. Orders

1	Test 2 STEP-3: True Payment Status: Half Payment <input type="text"/> <input type="button" value="SEND MESSAGE"/>
2	Demo 6 STEP-3: True Payment Status: Half Payment <input type="text"/> <input type="button" value="SEND MESSAGE"/>

« Page 5 of 5 »

STEP-5 + 6: FOR FULL PAYMENTS REPORT DELIVERY...

Payment Status: Full Payment

Upload report:

2 No file selected.

Send message:

3	Testing 1 Check STEP-4: True Payment Status: Full Payment Report Delivered
4	Test Name 13 Edited STEP-4: True Payment Status: Full Payment Report Delivered
5	Mild Stroke STEP-4: True Payment Status: Full Payment Report Delivered

« Page 5 of 5 »

4.2.16 Diagnostic Staff Dashboard Completed Reports:

STEP-7: PAYMENT COMPLETE AND REPORT SENT COMPLETED ORDERS				
NO.	Orders	DateTime	Reports	Status
1	Demo 5	July 19, 2019, 11:17 p.m.	reports_PDF/report_demo.pdf	Complete
2	Mild Stroke	July 19, 2019, 11 p.m.	reports_PDF/Convocation-Registration-Form_3Xmongu.pdf	Complete
3	Dengue Fever	July 19, 2019, 7:20 p.m.	reports/Convocation-Registration-Form_jpFqmuJ.pdf	Complete
4	Demo 1	July 19, 2019, 7:01 p.m.	reports/Convocation-Registration-Form_QQv97Vy.pdf	Complete
5	Demo 4	July 19, 2019, 6:53 p.m.	reports/Convocation-Registration-Form_Q9BZPNR.pdf	Complete

« Page 4 of 4 »

4.2.17 Diagnostic Staff Dashboard Due Payment:

STEP-8: PAYMENT NOT COMPLETE AND REPORT NOT SENT PENDING ORDERS				
NO.	Orders	DateTime	Reports	Status
1	Test Name 9 Edited	Aug. 27, 2019, 10:22 p.m.		Please Complete Your Full Payment for REPORT
2	Test Name 7	Aug. 27, 2019, 10:22 p.m.		Please Complete Your Full Payment for REPORT
3	Test Name 9 Edited	Aug. 27, 2019, 10:22 p.m.		Please Complete Your Full Payment for REPORT
4	Test Name 12 Edited	Aug. 27, 2019, 10:22 p.m.		Please Complete Your Full Payment for REPORT

Page 1 of 1.

4.2.18 Customer Registration, Login, Logout and Edit Profile:

SIGN IN

USERNAME:


Username

PASSWORD:

Password

REMEMBER ME: ☐

SIGN IN

 Login with Google

Not a member? Create new account

ODLRS

HomeCentersTestsCategoriesLoginSign Up

SIGN IN

USERNAME:
Username

PASSWORD:
Password

Sign In

SIGN UP WITH
GOOGLE OR SIGN UP

© Copyrights 2019. All rights reserved.

ODLRS

HomeCentersTestsCategoriesclient1

ARE YOU SURE YOU WANT TO SIGN OUT?

YES

© Copyrights 2019. All rights reserved.

ODLRS

HomeCentersTestsCategoriesclient1

Edit Your Profile



Profile name:
Sohanur Rahman

Image: Currently: [profile_pics/img4.jpg](#)
Change: No file chosen

Contact no:
00 123 345 567

Address:
Mohammadpur, Dhaka


Update

© Copyrights 2019. All rights reserved.

4.2.19 Customer Profile Details:

ODLRS

HomeCentersTestsCategoriesclient1




Siyam Al Galib

Email:
client1@email.com
Contact:
+880 1516113959
Address:
Mohammadpur, Dhaka

Order Name	Orders Status
Final Test 12	Order Confirmed. You are Ready For Test.
Final Test 11	Order Confirmed. You are Ready For Test.
Final Test 10	Order Rejected
Final Test 9	Order Pending...
Final Test 8	Order Pending...

Page 1 of 15. »



Siyam Al Galib

Edit Profile

View Profile

Reports

Logout

Developed By Galib and Sohanur

4.2.20 Customer Test Reports:

ODLRS

HomeCentersTestsCategoriesclient1

Reports

No.	Test Names	Validation DateTime	Reports	Status
1	Final Test 11	Aug. 31, 2019, 5:58 p.m.	reports_PDF/report_demo_OHQcv6Q.pdf	Complete
2	Final Test 12	Aug. 31, 2019, 5:56 p.m.	reports_PDF/report_demo_FTclIYi.pdf	Complete
3	Final Test 11	Aug. 31, 2019, 5:55 p.m.		Please Complete Your Full Payment for REPORT
4	joy	Aug. 31, 2019, 1 p.m.	reports_PDF/_working_SWxKYe7.pdf	Payment Done
5	joy	Aug. 31, 2019, 12:59 p.m.		Please make a full payment
6	Final Test 5	Aug. 31, 2019, 11:33 a.m.	reports_PDF/_working_xub25xP.pdf	Payment Done
7	Final Test 5	Aug. 31, 2019, 11:32 a.m.	reports_PDF/_working.pdf	Payment Done
8	Final Test 5	Aug. 30, 2019, 3:53 p.m.	reports_PDF/report_demo_v3dkXrV.pdf	Complete

Page 1 of 2. »

4.2.21 Homepage:

ODLRS
Home Centers Tests Categories client1

ONLINE TEST BOOKING

BOOK NOW ANY KIND OF TESTS

GET STARTED →

Search Tests

SEARCH

ALL TESTS

Blood Test

Final Test 17

Labaid Diagnostic Centre

Discount: 100 TK

Price: 600 TK

ORDER

ECG

Final Test 16

Labaid Diagnostic Centre

Discount: 100 TK

Price: 600 TK

ORDER

X-Ray

Final Test 15

Labaid Diagnostic Centre

Discount: 100 TK

Price: 600 TK

ORDER

ECG

Final Test 14

Labaid Diagnostic Centre

Discount: 100 TK

Price: 600 TK

ORDER

CT-Scan

Final Test 13

Labaid Diagnostic Centre

Discount: 100 TK

Price: 600 TK

ORDER

Others

Final Test 12

Labaid Diagnostic Centre

Discount: 100 TK

Price: 600 TK

ORDER

Category 6

Final Test 11

Labaid Diagnostic Centre

Discount: 100 TK

Price: 600 TK

ORDER

Category 5

Final Test 10

Labaid Diagnostic Centre

Discount: 100 TK

Price: 600 TK

ORDER

Category 4

Final Test 9

Labaid Diagnostic Centre

Discount: 100 TK

Price: 600 TK

ORDER

Category 3

Final Test 8

Labaid Diagnostic Centre

Discount: 100 TK

Price: 600 TK

ORDER

Category 2

Final Test 7

Labaid Diagnostic Centre

Discount: 100 TK

Price: 600 TK

ORDER

Category 1

Final Test 6

Labaid Diagnostic Centre

Discount: 100 TK

Price: 600 TK

ORDER

Page 1 of 5

Developed By Galib and Sohanur

4.2.22 Order Form:

ORDER

CONTACT NO:
+880 1516113959

EMAIL:
client1@email.com

ADDRESS:
Mohammadpur, Dhaka

PAYMENT TYPE:
Full Payment

PAYMENT METHOD:

BOOKED TIME SLOT:

BOOKED DATE:
____ - ____ - ____

ORDER

4.2.23 Order Details:

ORDER DETAILS


Name	Email
Siyam Al Galib	client1@email.com
Test Name	Category
Final Test 17	Blood Test
Center	Booked Time Slot:
Labaid Diagnostic Centre	10:00 AM - 2:00 PM
Booked Date:	Payment Type
March 10, 2019	Full Payment
Payment Method	Discount
BKASH	100
Test Price	Total Price:
600	500
Contact:	Address
+880 1516113959	Mohammadpur, Dhaka

CONFIRM

4.2.24 Order Status:

ODLRS

Home Centers Tests Categories client1



Siyam Al Galib

Email:
client1@email.com

Contact:
+880 1516113959

Address:
Mohammadpur, Dhaka

Order Name	Orders Status
Mild Stroke	Order Pending...
Dengue Fever	Order Pending...
Dengue Fever	Order Pending...
Final Test 3	Order Pending...
Final Test 3	Order Confirmed. You are Ready For Test.

Page 1 of 14. »

© Copyrights 2019. All rights reserved.

4.2.25 Order Completed and Due Payment:

Completed Order Details

Full Payment Complete

Name	Test Name
Siyam Al Galib	Final Test 11
Email	Contact
client1@email.com	+880 1516113959
Booked DateTime	Total Price
Sept. 15, 2019 (8:00 PM - 10:00 PM)	500Tk.
Received Payment	Payment Status
500Tk.	Full Payment (ON SPOT)
Validation DateTime	Download Report
Aug. 31, 2019, 5:58 p.m.	reports_PDF/report_demo_OHQcv6Q.pdf

Back to Home

Completed Order Details

Payment Due 250Tk.

Name	Test Name
Siyam Al Galib	Test Name 9 Edited
Email	Contact
client1@email.com	+880 1516113959
Booked DateTime	Received Payment
Nov. 9, 2019 (8:00 PM - 10:00 PM)	500
Due	Payment Status
250	Half Payment (CREDIT CARD)
Validation DateTime	Download Report
Aug. 27, 2019, 10:22 p.m.	

Back to Home

CompletePayment

5 Chapter 5: Software Testing

Software testing is a process of executing a program or application with the intent of finding the software bug. Testing assesses the quality of the product. In the software development life cycle it comes before implementation of software. It can also be stated as the process of validating and verifying that a software program or application or product:

- Meets the business and technical requirements that guided its design and development
- Works as expected
- Can be implemented with the same characteristic.

Among various types of software testing methods we used **Unit Testing** method in this project.

5.1 Unit Testing

A level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed. In this approach, Software testing methods are traditionally divided into **white-box** and **black-box** testing.

5.1.1 White-Box Testing:

The technique of testing in which the tester is aware of the internal workings of the product, have access to its source code and is conducted by making sure that all internal operations are performed according to the specifications is known as white box testing.

For this test on the system the following things were kept in mind while coding and testing:

1. **Statement coverage:** In this technique, all the statements are traversed at least once. Hence, each line of code is tested. Since all lines of code are covered, it helps in pointing out faulty code and recovering them.
2. **Branch Coverage:** In this technique, test cases are designed so that each branch from all decision points are traversed couple of times and thus give us bug free version of this product.
3. **Condition Coverage:** In this technique, all individual conditions are tested carefully, this means that each individual condition is one time true and false. In other words, we cover all conditions.

5.1.2 Black-Box Testing:

Black box testing is performed in a scenario when the software testing expert is unaware of the internal structure of the software. Generally, it is used to test the functionality and performance of the software or application from the user perspective. These tests can be functional or non-functional, though usually functional. In SRS section which functional requirements were mentioned, we found some bugs their while testing in Black-Box testing and we fixed them. Now all the functions are bug free and working correctly as expected.

6 Chapter 6: Conclusion and Future Work

Managing the diagnostic test booking and reporting system is difficult if there is no good system. The **Diagnostic Lab Reporting System** is very efficient and convenient to use and can be used in any diagnostic center or e-commerce organization to order product and processing effectively. Using this system, diagnostic center officials can provide better service to the patients. So, the better the booking and reporting the better the improvement of both centers and customers.

We tried to give our best to achieve all goals from this application but still, we need a lot of improvements. In future there could be several improvements done and adding some feature will make this app more usable and widen. List of some features and improvements are given below:

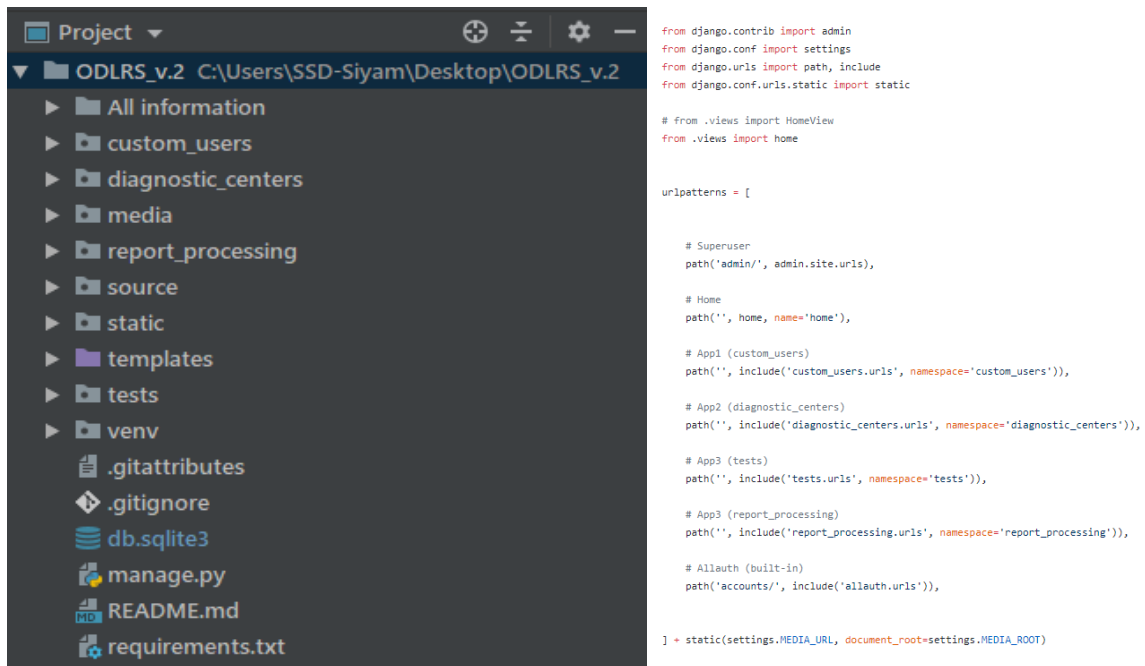
- Push Notifications.
- Payment Gateways.
- Advance Search Option.
- Cart system for multiple orders.

This project report tries to conclude everything about the whole project. So, to get a clear view on the system, we insist to read in carefully for better understanding.

7 Chapter 7: Important Source Codes

Some important source codes of the Diagnostic Lab Reporting System are given below:

7.1 Project Structure and Source PATHs:



7.2 Source codes for Backend setting:

```
"""Django settings for source project..."""
from django.contrib.messages import constants as messages
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

...

SECRET_KEY = 'f+@qe(ei%n#@h^*0$54v1#a9ukrgfa70wa3ed!47g7k#_c+76j'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []
```

```

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.sites',
    # Apps
    'custom_users.apps.CustomUsersConfig',
    'diagnostic_centers',
    'tests',
    'report_processing',
    # allauth
    'allauth',
    'allauth.account',
    'allauth.socialaccount',
    # Google Auth
    'allauth.socialaccount.providers.google',
    # Facebook Auth
    # 'allauth.socialaccount.providers.facebook',
    # GitHub Auth
    # 'allauth.socialaccount.providers.github',
]

```

```

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'source.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates'), os.path.join(BASE_DIR, 'templates', 'account')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

```

```

AUTHENTICATION_BACKENDS = (

    # Needed to login by username in Django admin, regardless of `allauth`
    'django.contrib.auth.backends.ModelBackend',

    # `allauth` specific authentication methods, such as login by e-mail
    'allauth.account.auth_backends.AuthenticationBackend',

)

WSGI_APPLICATION = 'source.wsgi.application'

# Database
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

```

```

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Asia/Dhaka'

USE_I18N = True

USE_L10N = True

USE_TZ = True

```

```

STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static')
]

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

LOGIN_REDIRECT_URL = 'home'
LOGOUT_REDIRECT_URL = 'home'

SITE_ID = 1

```

```

ACCOUNT_EMAIL_REQUIRED = True
ACCOUNT_USERNAME_REQUIRED = True

EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'

PASSWORD_HASHERS = (
    'django.contrib.auth.hashers.MD5PasswordHasher',
)

MESSAGE_TAGS = {
    messages.INFO: 'alert alert-info',
    messages.SUCCESS: 'alert alert-success',
    messages.WARNING: 'alert alert-warning',
    messages.ERROR: 'alert alert-danger',
    messages.DEBUG: 'alert alert-info',
}

```

7.3 Some source codes for Diagnostic Admin Login, Logout, Dashboard:

```

def admin_login(request):
    admin_login_form = AdminLoginForm()

    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')

        try:
            DiagnosticAdmin.objects.get(username=username, password=password)
            return redirect('diagnostic_centers:admin-dashboard', username)

        except DiagnosticAdmin.DoesNotExist:
            return redirect('diagnostic_centers:admin-login')

    template = 'diagnostic_centers/admin_login.html'

    context = {'admin_login_form': admin_login_form}

    return render(request, template, context)

```

```
def admin_dashboard(request, username=None):
    admin = DiagnosticAdmin.objects.get(username=username)
    validated_orders = TestOrder.objects.filter(accepted=True, validation=True, test_info_center=admin.center)
    completed_orders = PaymentValidation.objects.filter(approved_order_payment_type='Full Payment')

    # Completed Paginator
    paginator = Paginator(completed_orders, 10)
    page = request.GET.get('page')
    completed_orders_paginator_data = paginator.get_page(page)

    template = 'diagnostic_centers/admin_dashboard.html'

    context = {
        'admin': admin,
        'completed_orders': completed_orders_paginator_data
    }

    return render(request, template, context)
```

```
def admin_logout(request):
    messages.success(request, 'Logged Out.', extra_tags='html_safe')
    return redirect('diagnostic_centers:admin-login')
```

7.4 Some source codes for Add Test and Category:

```
def add_category_by_admin(request, username=None):
    if request.method == 'POST':
        category_add_form = CategoryAddForm(request.POST)

        if category_add_form.is_valid():
            add_category = category_add_form.save(commit=False)

            admin = DiagnosticAdmin.objects.get(username=username)
            add_category.center = DiagnosticCenter.objects.get(id=admin.center.id)

            add_category.save()

            return redirect('tests:filtered-categories-by-admin', username)

    else:
        category_add_form = CategoryAddForm()

    template = 'tests/add_category.html'
    context = {'category_add_form': category_add_form}

    return render(request, template, context)
```

```
def add_test_by_admin(request, username=None):
    if request.method == 'POST':
        test_add_form = TestAddForm(request.POST)

        if test_add_form.is_valid():
            add_test = test_add_form.save(commit=False)

            admin = DiagnosticAdmin.objects.get(username=username)
            add_test.center = DiagnosticCenter.objects.get(id=admin.center.id)

            add_test.save()

            return redirect('tests:added-tests-list-staff-admin', username)

    else:
        test_add_form = TestAddForm()

    template = 'tests/add_test.html'
    context = {'test_add_form': test_add_form}

    return render(request, template, context)
```

7.5 Some source codes for Order System:

```

@login_required
def test_order(request, id=None):
    try:
        current_profile = Profile.objects.get(user=request.user)
        email = current_profile.user.email
        contact_no = current_profile.contact_no
        address = current_profile.address
        initial_data = {
            'email': email,
            'contact_no': contact_no,
            'address': address
        }
    except:
        return redirect('account_login')
    if request.method == 'POST':
        test_order_form = TestOrderForm(request.POST)
        if test_order_form.is_valid():
            order = test_order_form.save(commit=False)
            order.client_info = Profile.objects.get(user=request.user)
            order.test_info = Test.objects.get(id=id)
            order.save()
            return redirect('tests:order-details', id=order.id)
    context = {
        'test_order_form': TestOrderForm(initial=initial_data),
        'test_id': id,
    }
    template = 'tests/order.html'
    return render(request, template, context)

```

```

def order_details_info(request, id=None):
    order_details = TestOrder.objects.get(id=id)

    total_price = int(order_details.test_info.price - order_details.test_info.discount)

    template = 'tests/order_details.html'

    context = {
        'order_details': order_details,
        'total_price': total_price,
    }

    return render(request, template, context)

```



```

def staff_approved(request, id=None, username=None):
    staff_order_detail = TestOrder.objects.get(id=id)
    staff_order_detail.accepted = True
    staff_order_detail.staff_check = True
    staff_order_detail.save()

    return redirect('diagnostic_centers:staff-dashboard', username=username)

#####

def staff_rejected(request, id=None, username=None):
    staff_order_detail = TestOrder.objects.get(id=id)
    staff_order_detail.accepted = False
    staff_order_detail.staff_check = True
    staff_order_detail.save()

    return redirect('diagnostic_centers:staff-dashboard', username=username)

```

```

def confirm_payment_message(request, id=None, username=None):
    order_details = TestOrder.objects.get(id=id)

    order_details.order_confirmed = True
    order_details.save()

    total_price = int(order_details.test_info.price - order_details.test_info.discount)

    template = 'tests/confirm_payment_message.html'
    context = {
        'order_details': order_details,
        'total_price': total_price,
        'staff_username': username,
    }

    return render(request, template, context)

#####

def came_for_test(request, id=None, username=None):
    staff_order_detail = TestOrder.objects.get(id=id)
    staff_order_detail.came_for_test = True
    staff_order_detail.save()

    return redirect('diagnostic_centers:staff-dashboard', username=username)

```

7.6 Some source codes for Reporting System:

```
def all_reports(request):
    all_reports_query = PaymentValidation.objects.all()
    paginator = Paginator(all_reports_query, 5)
    page = request.GET.get('page')
    all_reports_paginator = paginator.get_page(page)
    template = 'report_processing/all_reports.html'
    context = {'all_reports_query': all_reports_paginator}

    return render(request, template, context)

#####

def single_report_details(request, id=None):
    report_details = PaymentValidation.objects.get(id=id)
    total_price = int(report_details.approved_order.test_info.price - report_details.approved_order.test_info.discount)
    due_price = int(
        (report_details.approved_order.test_info.price - report_details.approved_order.test_info.discount) / 2)

    template = 'report_processing/single_report_details.html'
    context = {
        'report_details': report_details,
        'total_price': total_price,
        'due_price': due_price,
    }

    return render(request, template, context)
```

```
def filtered_report(request, id=None):
    user = get_object_or_404(User, id=id)

    filtered_reports = PaymentValidation.objects.filter(approved_order__client_info__user=user)

    # Filtered reports Paginator
    paginator = Paginator(filtered_reports, 20)
    page = request.GET.get('page')
    filtered_reports_paginator = paginator.get_page(page)

    template = 'account/custom_users/filtered_reports.html'

    context = {'filtered_reports': filtered_reports_paginator}

    return render(request, template, context)
```

```
def complete_due_payment(request, id=None, report_id=None):
    existing_order = TestOrder.objects.get(id=id)
    existing_order.payment_type = 'Full Payment'
    existing_order.validation = False
    existing_order.save()

    report_details = PaymentValidation.objects.get(id=report_id)

    total_price = int(report_details.approved_order.test_info.price - report_details.approved_order.test_info.discount)

    due_price = int(
        (report_details.approved_order.test_info.price - report_details.approved_order.test_info.discount) / 2)

    template = 'report_processing/single_report_details.html'

    context = {
        'report_details': report_details,
        'total_price': total_price,
        'due_price': due_price,
    }

    return render(request, template, context)
```

7.7 Some source codes for Search, Pagination, Categories, Profile Edit, and Delete Tests:

```
def test_categories(request, template_name='tests/categories.html'):
    categories = TestCategory.objects.all().order_by('-id')

    context = {'categories': categories}

    return render(request, template_name, context)

#####

def categorise_tests(request, id=None):
    filtered_tests = Test.objects.filter(category__id=id)

    template = 'tests/categorise_tests.html'
    context = {'filtered_tests': filtered_tests}

    return render(request, template, context)
```

```
def delete_test(request, id=None, username=None):
    test_object = Test.objects.get(id=id)
    admin = DiagnosticAdmin.objects.get(username=username)
    test_object.center = DiagnosticCenter.objects.get(id=admin.center.id)

    if test_object.center is not None:
        test_object.delete()
    return redirect('tests:added-tests-list-staff-admin', username)

#####

def edit_test(request, id=None, username=None):
    test_query = Test.objects.get(id=id)
    edit_form = TestAddForm(request.POST or None, instance=test_query)
    if request.method == 'POST':
        if edit_form.is_valid():
            edited = edit_form.save(commit=False)
            admin = DiagnosticAdmin.objects.get(username=username)
            edited.center = DiagnosticCenter.objects.get(id=admin.center.id)
            edited.save()
            return redirect('tests:added-tests-list-staff-admin', username)
    context = {'edit_form': edit_form}
    template = 'tests/edit_test.html'
    return render(request, template, context)
```

```

def search_paginator(request):
    Centers = DiagnosticCenter.objects.all()

    query = request.GET.get('q')
    if query:
        Centers = Centers.filter(
            Q(name__icontains=query) |
            Q(website__icontains=query)
        )
        print(Centers)

    paginator = Paginator(Centers, 12)
    page = request.GET.get('page')
    all_centers = paginator.get_page(page)

    context = {'Centers': all_centers,}
    template_name = 'diagnostic_centers/all_centers.html'

    return render(request, template_name, context)

```

```

@login_required()
def profile_edit(request, template_name='account/custom_users/profile_edit.html'):
    existing_profile = get_object_or_404(Profile, user=request.user)
    profile_form = ProfileUpdateForm(instance=existing_profile)

    if request.method == 'POST':
        profile_form = ProfileUpdateForm(request.POST, request.FILES, instance=existing_profile)

        if profile_form.is_valid():
            profile = profile_form.save(commit=False)
            profile.user = request.user
            profile.save()
            messages.success(request, 'Profile Updated for {}'.format(request.user.username), extra_tags='html_safe')
            return redirect('custom_users:orders-by-user')

    context = {
        'profile_form': profile_form,
        'existing_profile': existing_profile,
    }

    return render(request, template_name, context)

```

7.8 All PATHs for Customer, Diagnostic Center, Test, and Report Applications:

```

from django.urls import path

from .views import (

    profile,
    profile_edit,
    orders_by_user,
    filtered_report,

    # MyLoginView,
    # MySignupView,
    # MyPasswordResetView,
    # MyPasswordChangeView,
)

app_name = 'custom_users'

urlpatterns = [

    # path('login/', MyLoginView.as_view(), name='account_login'),

    path('profile/', profile, name='profile'),

    path('profile-edit/', profile_edit, name='profile-edit'),

    path('orders-by-user/', orders_by_user, name='orders-by-user'),

    path('filtered-reports-by-user/<int:id>/', filtered_report, name='filtered-reports-by-user')

    # path('password_reset', MyPasswordResetView.as_view(), name='account_reset_password'),

    # path('password/change/', MyPasswordChangeView, name='account_change_password'),

]

```

```

from django.urls import path

from .views import (

    search_paginator,

    admin_login,
    admin_logout,
    admin_dashboard,

    staff_login,
    staff_logout,
    staff_dashboard,

    center_details,

)

app_name = 'diagnostic_centers'

urlpatterns = [

    path('all-centers/', search_paginator, name='all-centers'),

    path('admin-login/', admin_login, name='admin-login'),
    path('admin-logout/', admin_logout, name='admin-logout'),
    path('admin-dashboard/<username>', admin_dashboard, name='admin-dashboard'),

    path('staff-login/', staff_login, name='staff-login'),
    path('staff-logout/', staff_logout, name='staff-logout'),
    path('staff-dashboard/<username>', staff_dashboard, name='staff-dashboard'),
    path('staff-dashboard/<int:id>/<username>', staff_dashboard, name='staff-dashboards'),

    path('center-details/<int:id>', center_details, name='center-details'),

]

from django.urls import path

from .views import all_reports, single_report_details, complete_due_payment

app_name = 'report_processing'

urlpatterns = [

    path('all-reports/', all_reports, name='all-reports'),

    path('single-report-details/<int:id>', single_report_details, name='single-report-details'),

    path('complete-due-payment/<int:id>/<int:report_id>', complete_due_payment, name='complete-due-payment'),

]

```

```

from django.urls import path

from .views import (

    all_tests,
    test_details,

    test_order,
    order_details_info,

    test_categories,
    categorise_tests,

    staff_approved,
    staff_rejected,

    add_test_by_admin,
    add_category_by_admin,
    filtered_categories_by_admin,
    filtered_categories_for_staff,

    delete_test,
    edit_test,

    edit_category,
    delete_category,

    all_tests_list_for_staff,
    added_tests_list_for_staff_admin,

    confirm_payment_message,

    came_for_test,

)

app_name = 'tests'

urlpatterns = [

    path('all-tests/', all_tests, name='all-tests'),

    path('categories/', test_categories, name='categories'),

    path('categorise-tests/<int:id>/', categorise_tests, name='categorise-tests'),

    path('test-details/<int:id>/', test_details, name='test-details'),

    path('order/<int:id>/', test_order, name='order'),

    path('order-details/<int:id>/', order_details_info, name='order-details'),

    path('staff-approved/<int:id>/<username>/', staff_approved, name='staff-approved'),

    path('staff-rejected/<int:id>/<username>/', staff_rejected, name='staff-rejected'),

    path('add-test/<username>/', add_test_by_admin, name='add-test'),

    path('add-category/<username>/', add_category_by_admin, name='add-category'),

    path('filtered-categories-by-admin/<username>/', filtered_categories_by_admin, name='filtered-categories-by-admin'),

    path('filtered-categories-for-staff/<username>/', filtered_categories_for_staff, name='filtered-categories-for-staff'),

    path('all-tests-list-staff/<username>/', all_tests_list_for_staff, name='all-tests-list-staff'),

    path('added-tests-list-staff-admin/<username>/', added_tests_list_for_staff_admin, name='added-tests-list-staff-admin'),

    path('delete-test/<int:id>/<username>/', delete_test, name='delete-test'),

    path('delete-category/<int:id>/<username>/', delete_category, name='delete-category'),

    path('edit-test/<int:id>/<username>/', edit_test, name='edit-test'),

    path('edit-category/<int:id>/<username>/', edit_category, name='edit-category'),

    path('confirm-payment-message/<int:id>/', confirm_payment_message, name='confirm-payment-message'),

    path('staff-confirm-payment-message/<int:id>/<username>/', confirm_payment_message, name='staff-confirm-payment-message'),

    path('came-for-test/<int:id>/<username>/', came_for_test, name='came-for-test'),

```

8 Chapter 8: References

- 1) <https://www.draw.io/>
- 2) <https://www.figma.com/>
- 3) <https://app.sqldb.com/>
- 4) <https://www.python.org/>
- 5) <https://getbootstrap.com>
- 6) <https://stackoverflow.com>
- 7) <https://www.youtube.com>
- 8) <https://www.behance.net/>
- 9) <https://www.wikipedia.org/>
- 10) <https://www.w3schools.com/>
- 11) <http://www.pharmaid.co/category/medicine/>
- 12) <https://www.djangoproject.com/start/overview/>
- 13) https://www.youtube.com/watch?v=FdVuKt_iuSI&t
- 14) <https://tech.marksblogg.com/passwords-in-django.html>
- 15) <https://developers.google.com/identity/protocols/OAuth2>
- 16) <https://django-notify-x.readthedocs.io/en/latest/usage.html>
- 17) <https://codeburst.io/django-render-html-to-pdf-41a2b9c41d16>
- 18) <https://django-allauth.readthedocs.io/en/latest/installation.html>
- 19) <https://fosstack.com/how-to-add-google-authentication-in-django/>
- 20) <https://wsvincent.com/django-user-authentication-tutorial-signup/>
- 21) <https://www.geeksforgeeks.org/software-engineering-white-box-testing/>
- 22) <https://stackoverflow.com/questions/37308246/django-how-to-save-my-hashed-password>
- 23) <https://www.digialocean.com/community/tutorials/how-to-send-web-push-notifications-from-django-applications>
- 24) <https://simpleisbetterthancomplex.com/tips/2018/02/10/django-tip-22-designing-better-models.html>
- 25) <https://medium.com/@techbloggers/white-box-testing-vs-black-box-testing-19754e950398>
- 26) <https://simpleisbetterthancomplex.com/tutorial/2016/07/22/how-to-extend-django-user-model.html>
- 27) <https://simpleisbetterthancomplex.com/tutorial/2017/02/18/how-to-create-user-sign-up-view.html>