

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 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](#_Toc7534350)

[1.1 Background 7](#_Toc7534351)

[1.2 Purpose 7](#_Toc7534352)

[1.3 Quick Overview 8](#_Toc7534353)

[Chapter 2: Requirement Engineering 9](#_Toc7534359)

[2.1 User requirements 9](#_Toc7534360)

[2.2 Software Requirement Specification 10](#_Toc7534361)

[2.3 Purpose 10](#_Toc7534362)

[2.4 Scope of project 10](#_Toc7534363)

[2.4 Definitions 10](#_Toc7534364)

[2.5 Overall Description 11](#_Toc7534365)

[2.6 Project Perspective 11](#_Toc7534366)

[2.7 Software Required 12](#_Toc7534367)

[2.7.1 Software Required 12](#_Toc7534368)

[2.7.2 Hardware Required 12](#_Toc7534369)

[2.8 Users of the System 12](#_Toc7534370)

[2.9 User Characteristics 12](#_Toc7534371)

[2.10 User functionalities and their scope 12](#_Toc7534372)

[2.11 Constraints 14](#_Toc7534373)

[2.12 External Interface Requirements 15](#_Toc7534374)

[2.12.1 User Interfaces 15](#_Toc7534375)

[2.12.2 Hardware Interfaces 15](#_Toc7534376)

[2.12.3 Software Interfaces 15](#_Toc7534377)

[2.12.4 Communications Interfaces 15](#_Toc7534378)

[2.13 Functional Requirement 16](#_Toc7534379)

[2.13.1 User Class 1 : System Admin 16](#_Toc7534380)

[2.13.2 User Class 2 : Diagnostic Admin 18](#_Toc7534381)

[2.13.3 User Class 3 : Diagnostic Staff 19](#_Toc7534382)

[2.13.4 User Class 2 : Customer 21](#_Toc7534383)

[Chapter 3: System Design & Implementation 23](#_Toc7534384)

[3.1 Software Development Method 24](#_Toc7534385)

[3.2 Work Process 24](#_Toc7534386)

[3.3 state transition diagram 25](#_Toc7534387)

[3.4 DATA Flow Diagram 26](#_Toc7534387)

[3.5 Entity Relationship Diagram 27](#_Toc7534387)

[3.6 Database Design 28](#_Toc7534388)

[3.6.1 List of Tables 28](#_Toc7534389)

[Chapter 4: Implementation 33](#_Toc7534390)

[4.1 Development Environment 33](#_Toc7534391)

[4.1.1 Common Issues 33](#_Toc7534392)

[4.2 Web Interface 33](#_Toc7534393)

[Chapter 5: Software Testing 51](#_Toc7534394)

[5.1 Unit Testing 51](#_Toc7534395)

[5.1.1 White-Box Testing 51](#_Toc7534396)

[5.1.2 Black-Box Testing 52](#_Toc7534397)

[Chapter 6: Conclusion 53](#_Toc7534398)

[Chapter 7: Important Source Codes 54](#_Toc7534399)

[Chapter 8: References 68](#_Toc7534400)

**Chapter 1:**

**1.1Background**  
  
Diagnostic Lab Reporting Systemexplicitly 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.

**1.2Purpose**  
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 coeds.

**Chapter 8**

Includes references.

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

**Chapter 2: Requirement Engineering**   
 **2.1 User Requirements**

1. Diagnostic admin should be able to assign a staff for certain tasks.
2. Diagnostic admin should be able to manage test, order, staff and customer operations.
3. Diagnostic admin should be able to create, delete new tests and categories.
4. Diagnostic admin should be able to view new added tests, new added categories, completed order details, and staff status.
5. Diagnostic staff should be able to approve, reject and verify the order.
6. Diagnostic staff should be able to confirm and complete the order.
7. Diagnostic staff should be able to upload, send messages and test reports to the customer.
8. Diagnostic staff should be able to view newly added tests and categories, completed and not completed order details and reporting admin.
9. Customers should be able to register, login, logout and edit their profile information.
10. Customers should be able to order tests and view the order processing status, order details, and report list from their profile.
11. 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.
12. System Admin should be able to manage own profile.
13. System admin should be able to add new diagnostic centers including admin and staff’s username and password.
14. System admin should be able to perform various maintainability.
15. The overall system should be clean, efficient and convenient to use.
16. 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 Systemand could be used by any Diagnostic Center and Ecommerce organizations. Scopes of the project is given below.

1. Will be used to insert and maintain customer’s order and personal information in a secure system.
2. Will be used to create diagnostic centers, admins, staffs and maintain their information efficiently.
3. The system will be able to show and update the information of diagnostic centers, admins, staffs and customers.
4. This system will help the diagnostic admins and staffs to maintain the order process efficiently.
5. Activities like inserting, updating, modification, deletion should be efficient.
6. On type of user shouldn’t be able to access other types of user’s page.

* 1. **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. |

## **Overall Description**

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

## **Project Perspective**

This project is a web based system implementing client-server model. This Diagnostic Lab Reporting Systemprovides 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. Diagnosticadmin 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 Diagnosticadmin and also add new programs.

## **Software Required**

### **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.

### **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.

## **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.

## **User Characteristics**

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

1. System Admin
2. DiagnosticAdmin
3. DiagnosticStaff
4. Customer

## **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 | 1. Can login, add, and modify own profile information. 2. Can add new diagnostic centers. 3. Can add new diagnosticadmin and set/reset password. 4. Can add new diagnosticstaff and set/reset password. 5. Can delete and modify everything. |
| DiagnosticAdmin | 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. |
| DiagnosticStaff | 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 | 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. |

## **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.

## **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.

### **User Interfaces**

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

### **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.

### **Software Interfaces**

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

### **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.

## **Functional Requirement**

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

### **User Class 1: System Admin**

#### **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.

#### **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.

#### **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.

#### **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.

#### **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.

#### **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.

#### **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.

#### **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.

### **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.*

#### **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.

#### **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.

#### **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.

#### **Functional Requirement 4**

ID : FR4

Title : Edit test.

Rational : In order to edit test.

Description : The diagnostic admin will be able to edit test.

#### **Functional Requirement 5**

ID : FR5

Title : Edit category.

Rational : In order to edit category.

Description : The diagnostic admin will be able to edit category.

#### **Functional Requirement 6**

ID : FR6

Title : Delete test.

Rational : In order to edit test.

Description : The diagnostic admin will be able to delete test.

#### **Functional Requirement 7**

ID : FR7

Title : Delete category.

Rational : In order to edit category.

Description : The diagnostic admin will be able to delete category.

#### **Functional Requirement 8**

ID : FR8

Title : View test.

Rational : In order to view test.

Description : The diagnostic admin will be able to view tests.

#### **Functional Requirement 9**

ID : FR9

Title : View category.

Rational : In order to view category.

Description : The diagnostic admin will be able to view categories.

#### **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.

#### **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.

### **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*.

#### **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.

#### **Functional Requirement 2**

ID : FR2

Title : View test.

Rational : In order to view test.

Description : The diagnostic staff will be able to view tests.

#### **Functional Requirement 3**

ID : FR3

Title : View category.

Rational : In order to view category.

Description : The diagnostic staff will be able to view categories.

#### **Functional Requirement 4**

ID : FR4

Title : Approve orders.

Rational : In order to approve orders.

Description : The diagnostic staff will be able to approve orders.

#### **Functional Requirement 5**

ID : FR5

Title : Reject orders.

Rational : In order to reject orders.

Description : The diagnostic staff will be able to reject orders.

#### **Functional Requirement 6**

ID : FR6

Title : Verify orders.

Rational : In order to verify orders.

Description : The diagnostic staff will be able to verify orders.

#### **Functional Requirement 7**

ID : FR7

Title : Upload report.

Rational : In order to upload report.

Description : The diagnostic staff will be able to upload report.

#### **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.

### **User Class 4: Customer**

#### **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.

#### **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.

#### **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.

#### **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.

#### **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.

#### **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.

#### **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.

#### **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.

1. Chapter 3: System Design & Implementation
   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

1. **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).

1. **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.

1. **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.

1. **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.

* 1. 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.

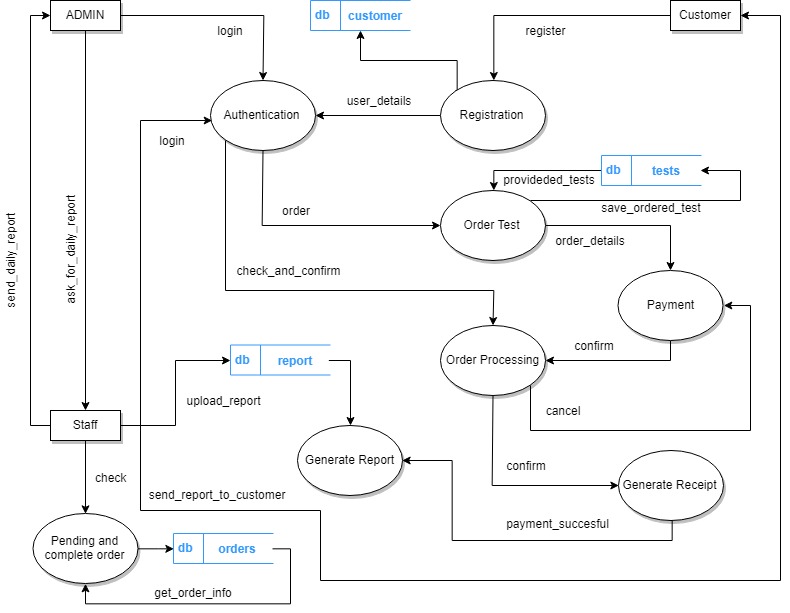
* 1. State Transition Diagram (STD)

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**

* 1. **Data Flow Diagram (DFD)**

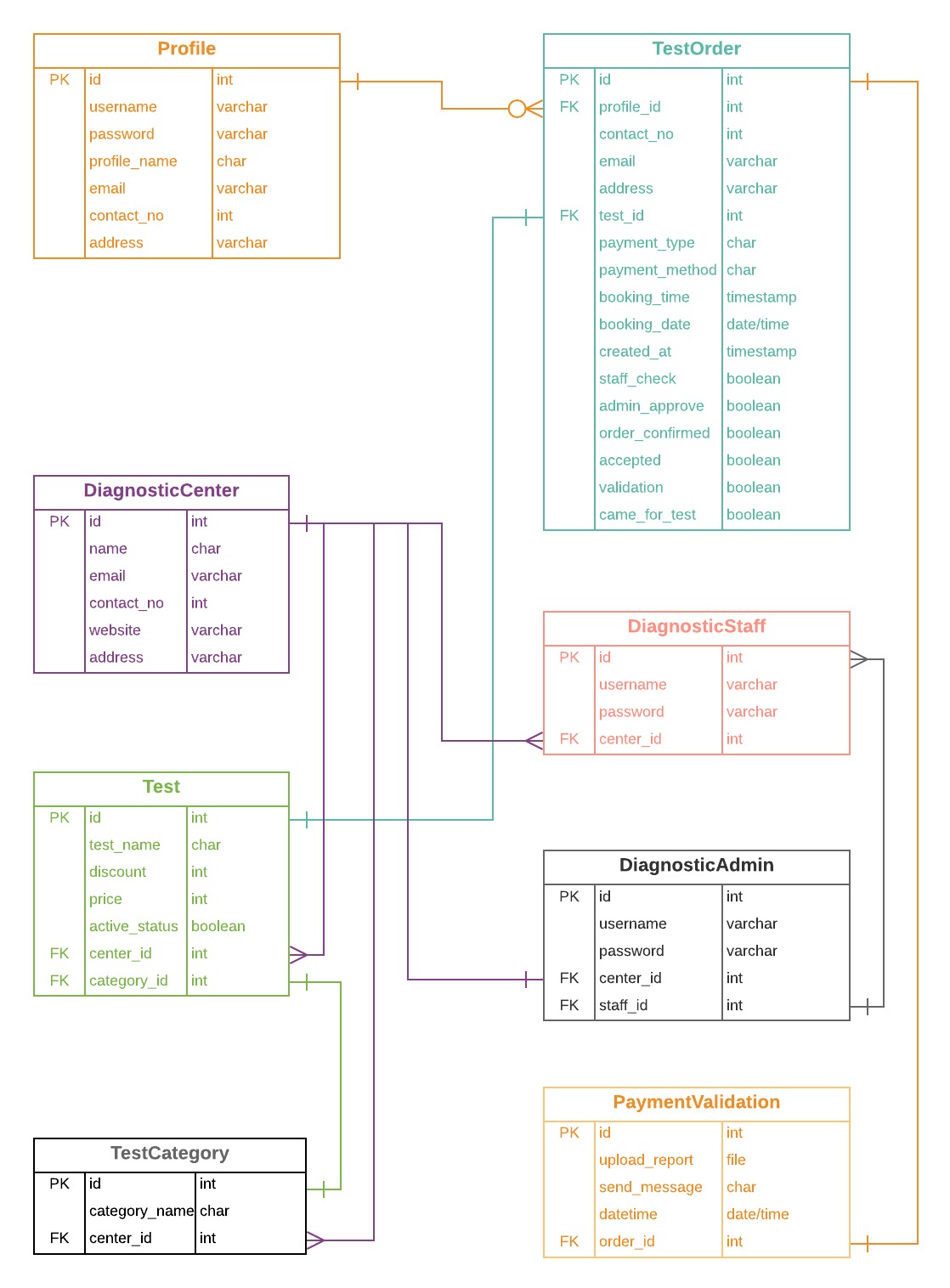
A data-flow diagram is a way of representing a flow of a data of a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. The final DFD for this project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



**Fig: Data Flow Diagram for Diagnostic Lab Reporting System**

* 1. **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**

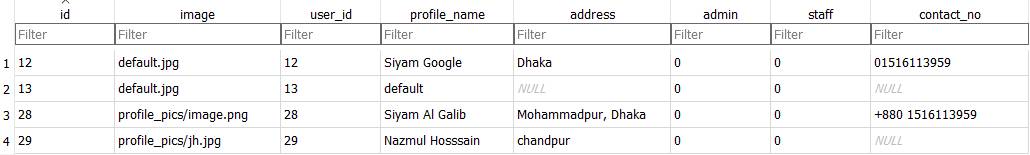
* 1. Database Design

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.

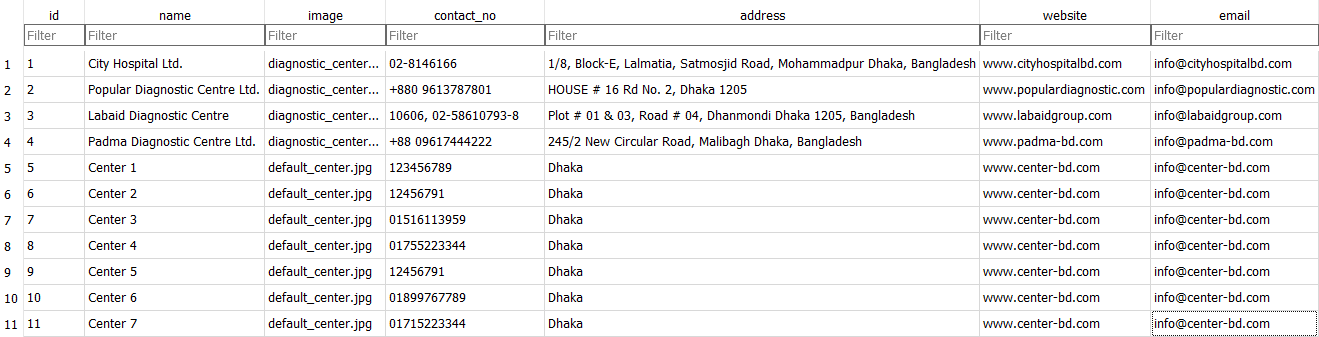
|  |  |
| --- | --- |
| **Table Name** | **Purpose/Description** |
| Profile | Holds the information of all customers |
| Diagnostic Center | Holds the information of all diagnostic centers |
| Diagnostic Admin | Holds the information of all diagnostic admins |
| Diagnostic Staff | Holds the information 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 |

#### **Table structure image for ‘Profile’**



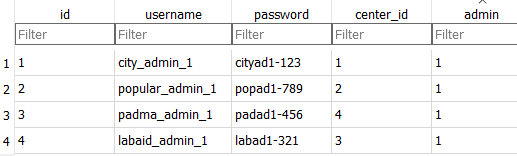
**Fig: Profile table structure**

#### **Table structure image for ‘Diagnostic Center’**



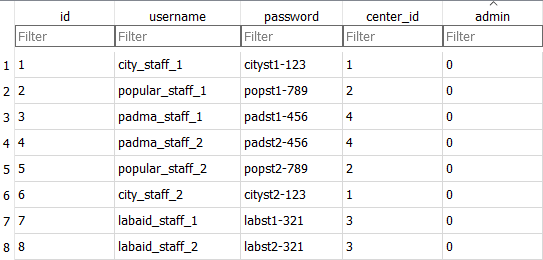
**Fig: Diagnostic Center table structure**

#### **Table structure image for ‘Diagnostic Admin’**



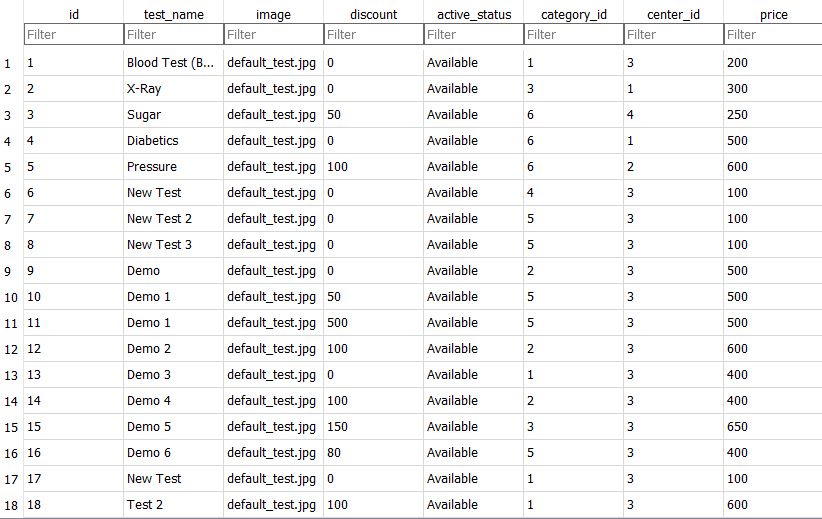
**Fig: Diagnostic Admin table structure**

#### **Table structure image for ‘Diagnostic Staff’**



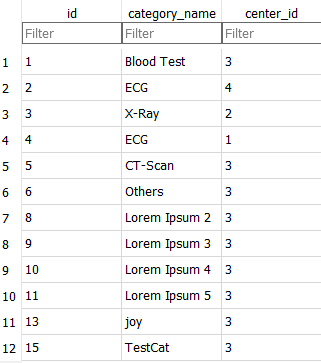
**Fig: Diagnostic Staff table structure**

#### **Table structure image for ‘Test’**



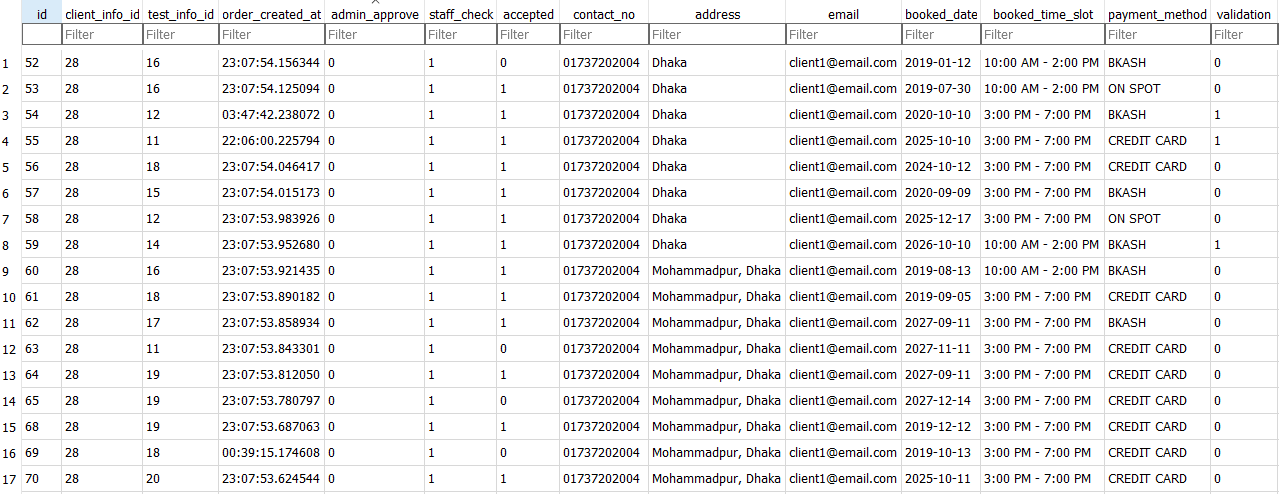
**Fig: Test table structure**

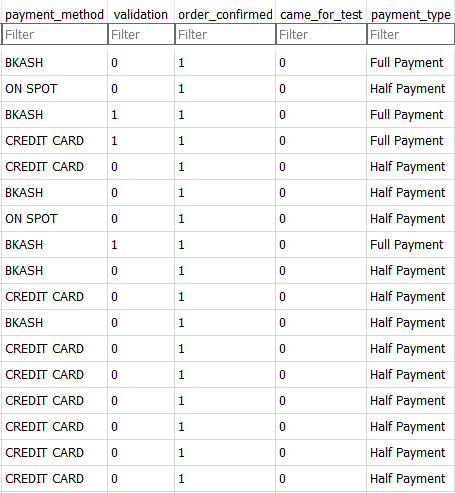
#### **Table structure image for ‘Test Category’**



**Fig: Test Category table structure**

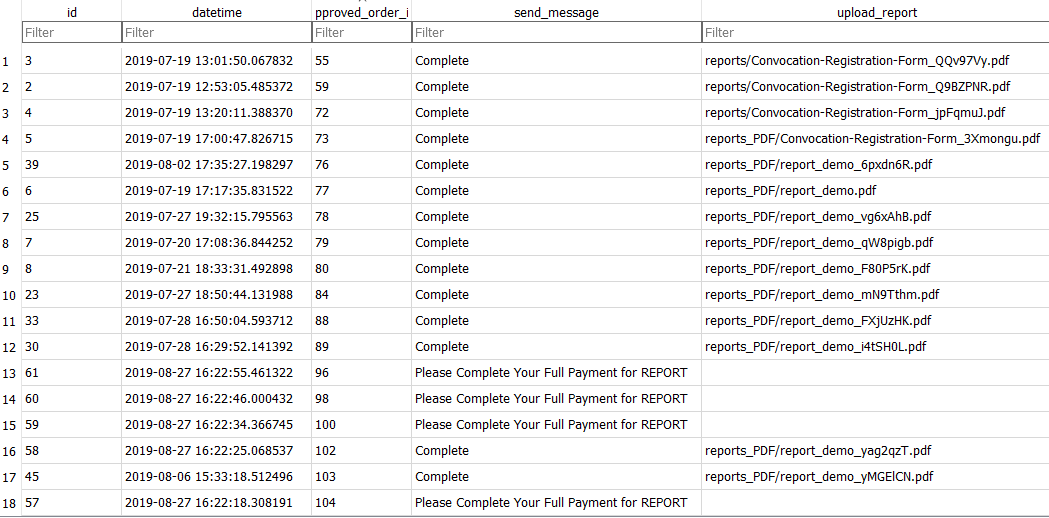
#### **Table structure image for ‘Test Order’**





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

#### **Table structure image for ‘Payment Validation’**



**Fig: Payment Validation table structure**

1. Chapter 4: Implementation
   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.

* + 1. Common Issues

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

#### **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.

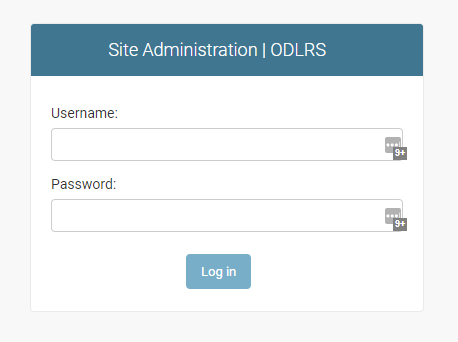
#### **b. Making a User-Friendly UX:** To make the UX as friendly as possible, we had to trim many complications. We go through a lot of testing and which UX is more efficient to use, we tested the application with some testers and from their feedback we come to the first version of the application.

#### **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.

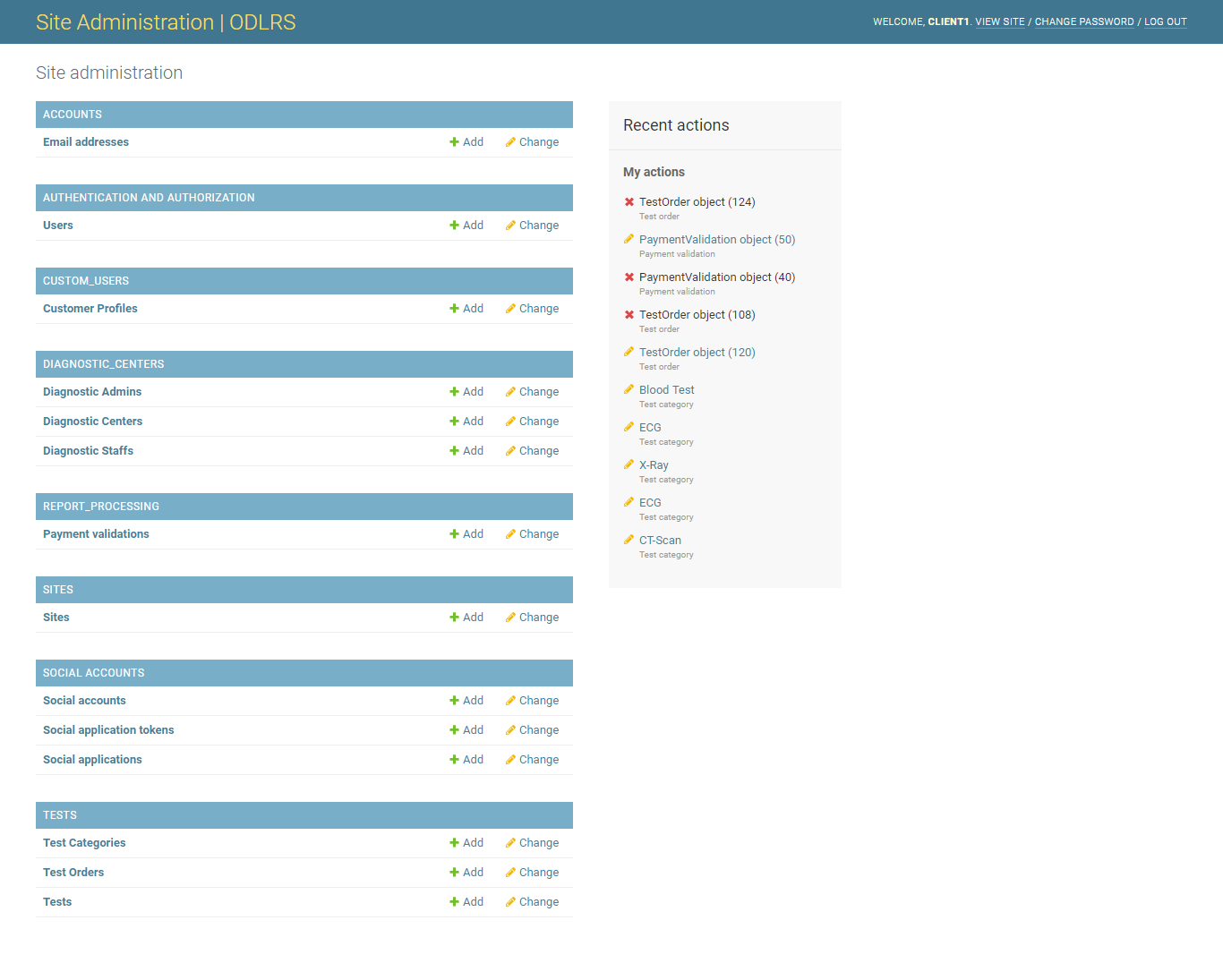
* 1. Web Interface

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

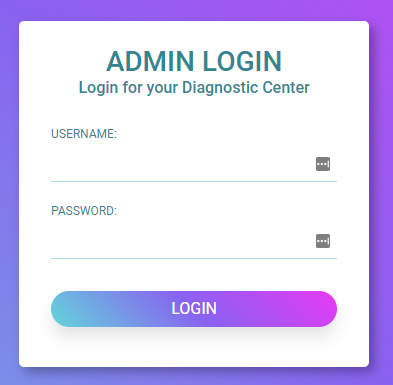
* + 1. **System Admin Login:**

****

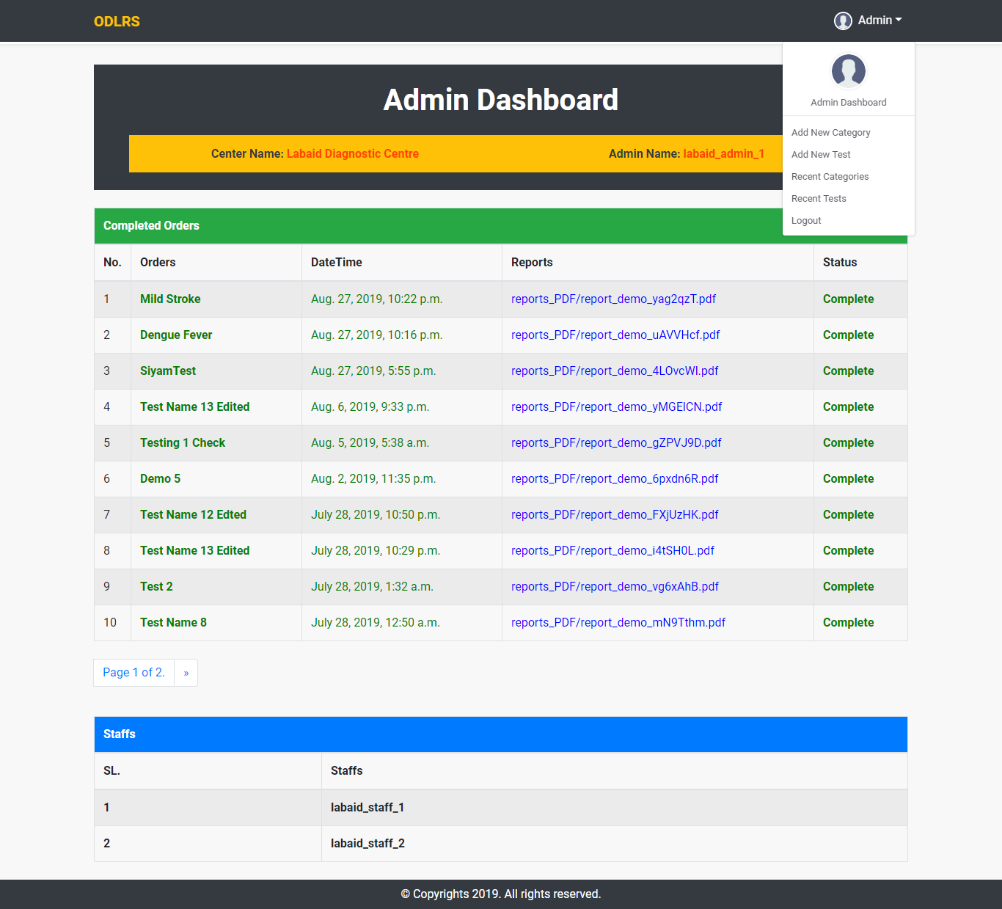
* + 1. **System Admin Dashboard:**

****

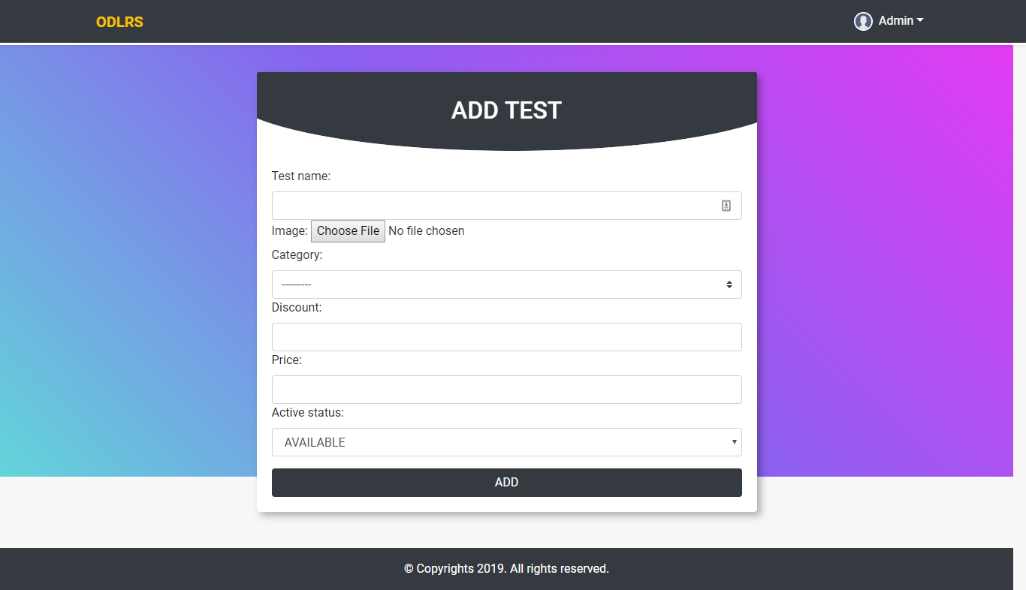
* + 1. **Diagnostic Admin Login:**

****

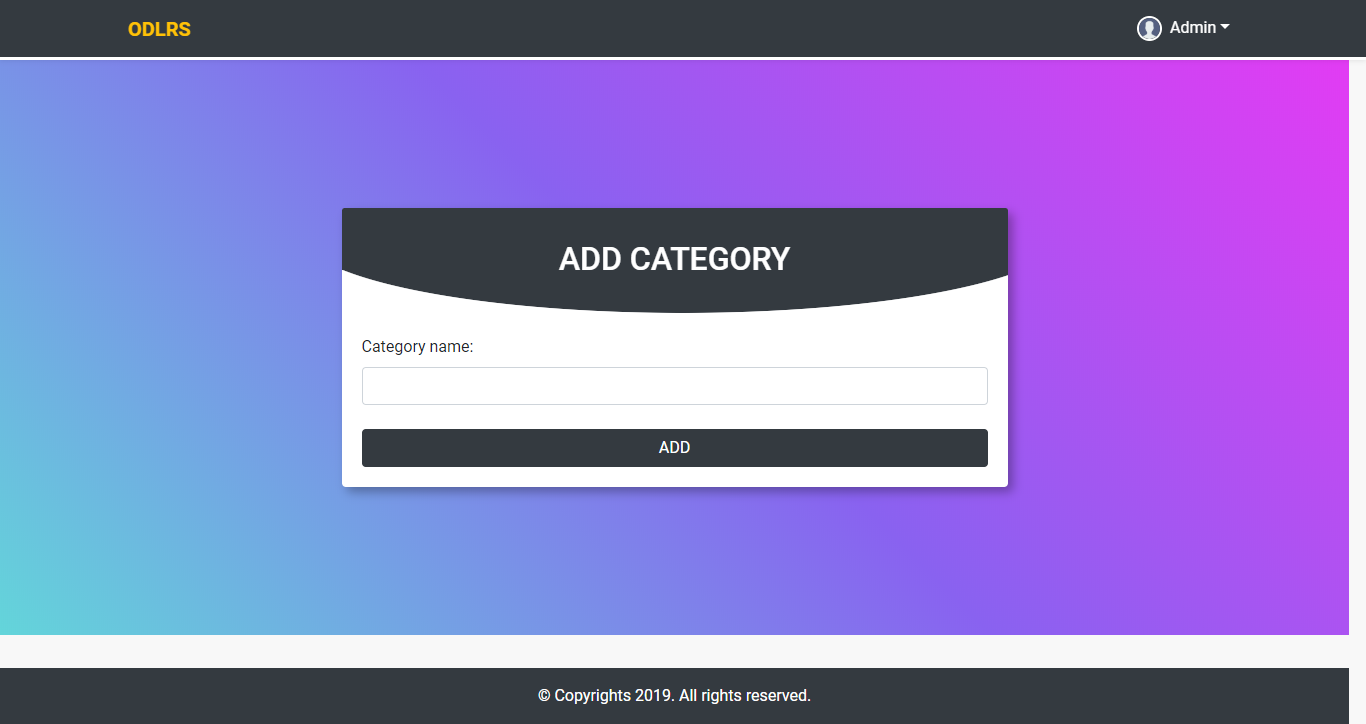
* + 1. **Diagnostic Admin Dashboard:**

****

* + 1. **Diagnostic Admin Add New Test:**

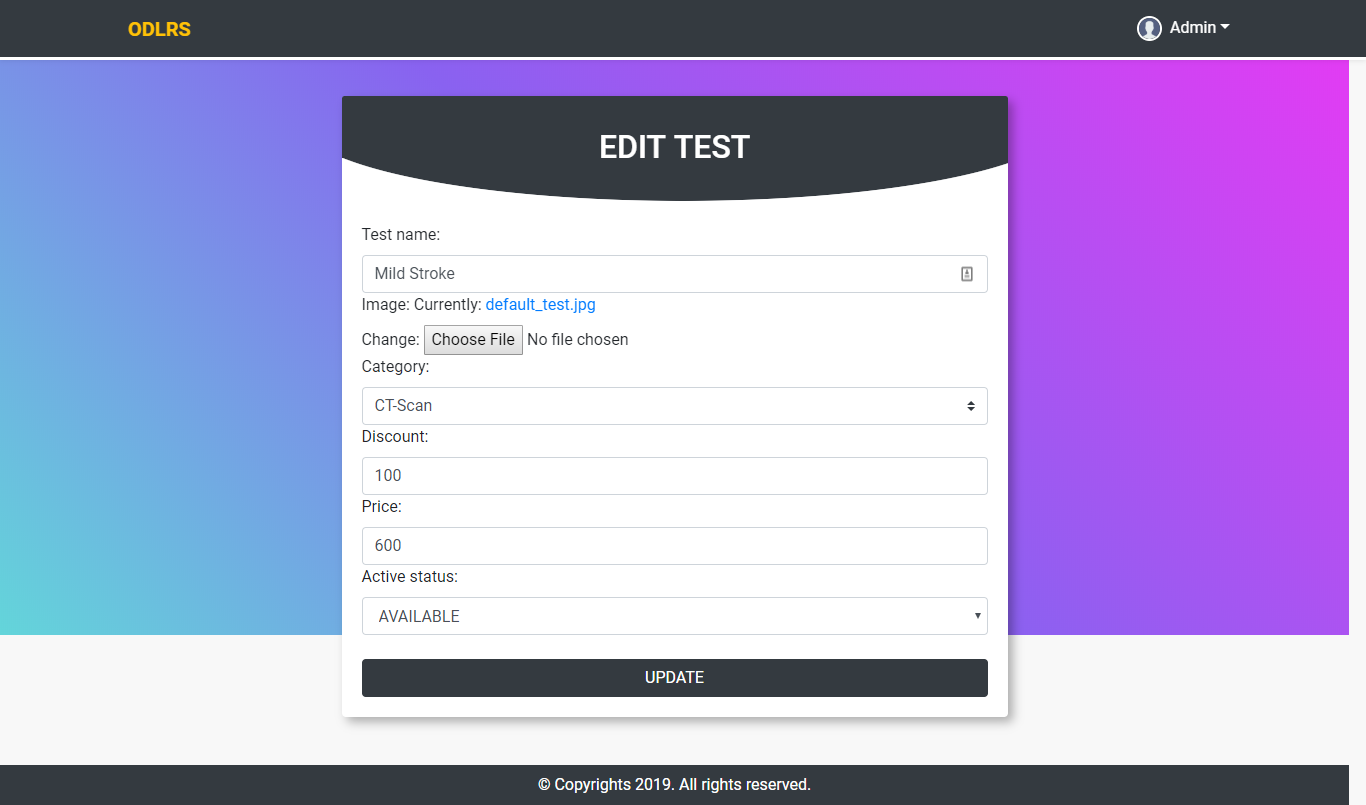
****

* + 1. **Diagnostic Admin Add New Category:**

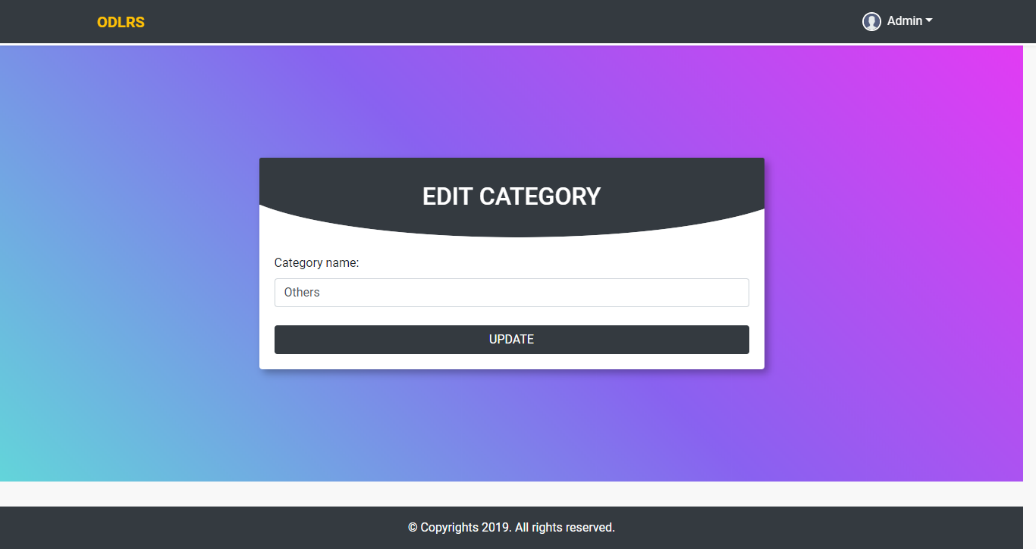
****

****

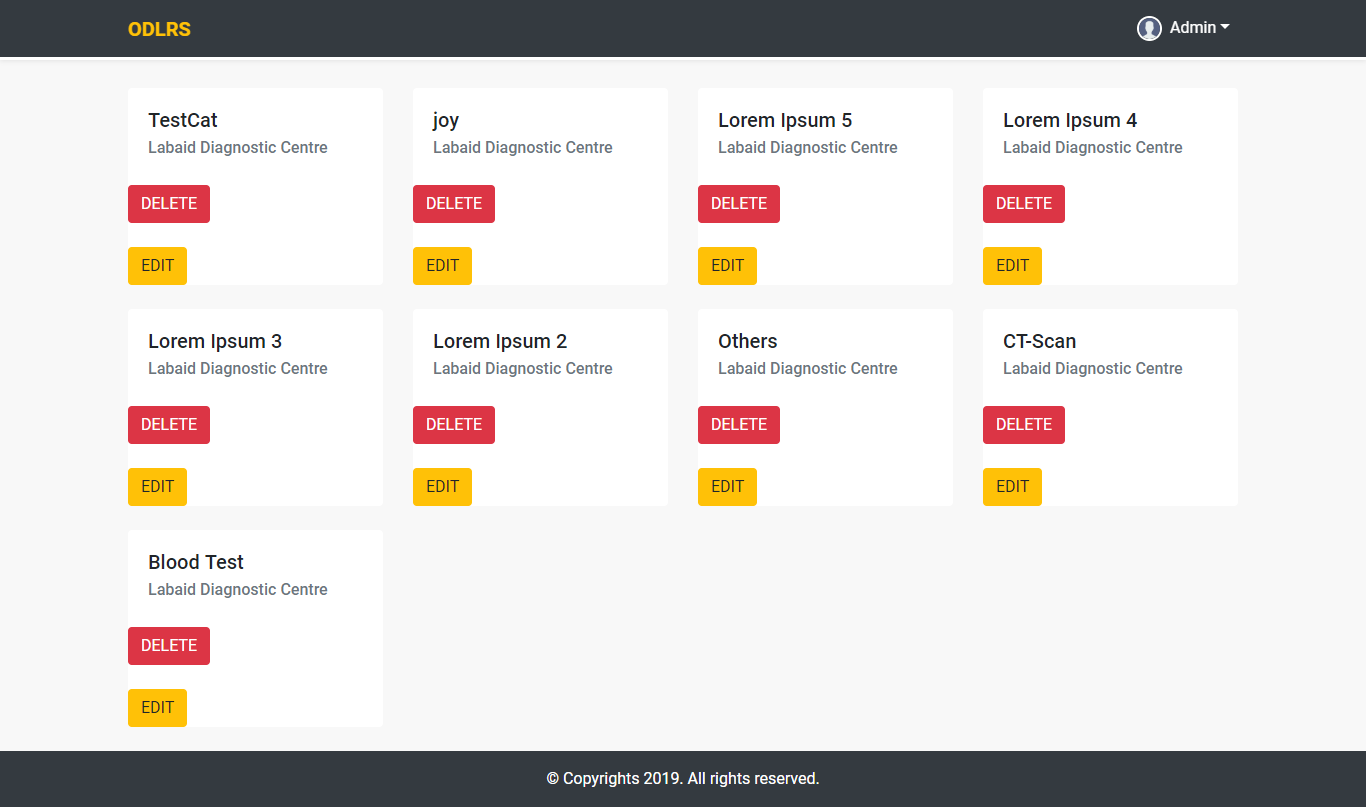
* + 1. **Diagnostic Admin Edit Test:**

****

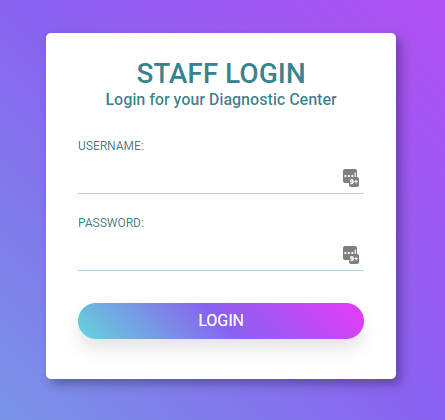
* + 1. **Diagnostic Admin Edit Category:**

****

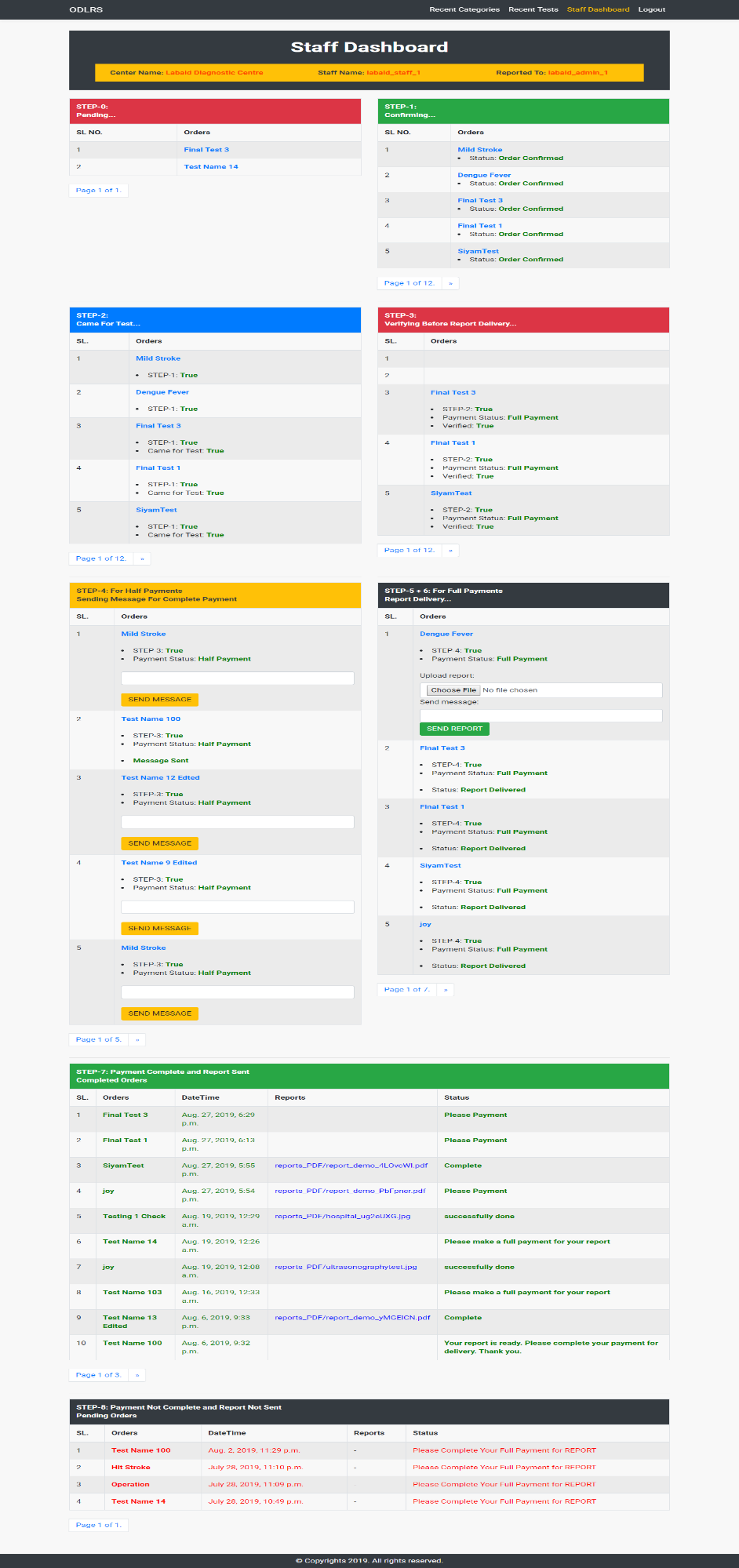
* + 1. **Diagnostic Admin Edit and Delete Contents:**

****

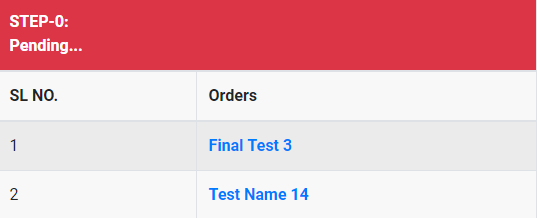
* + 1. **Diagnostic Staff Login:**

****

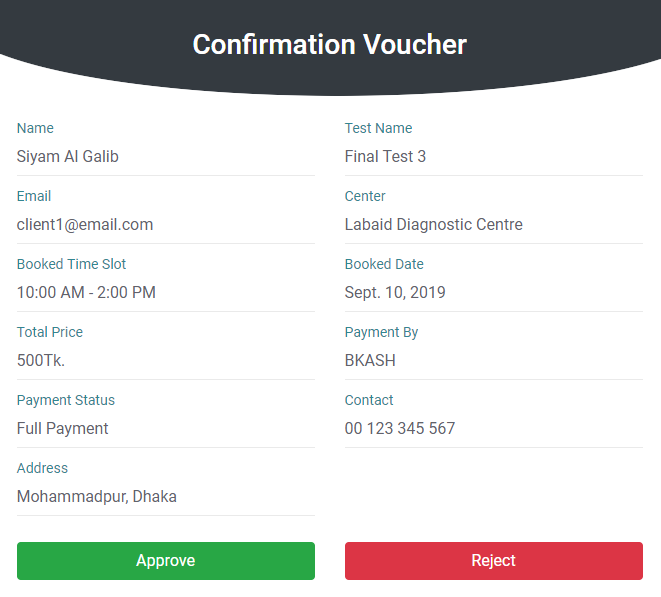
* + 1. **Diagnostic Staff Dashboard:**

****

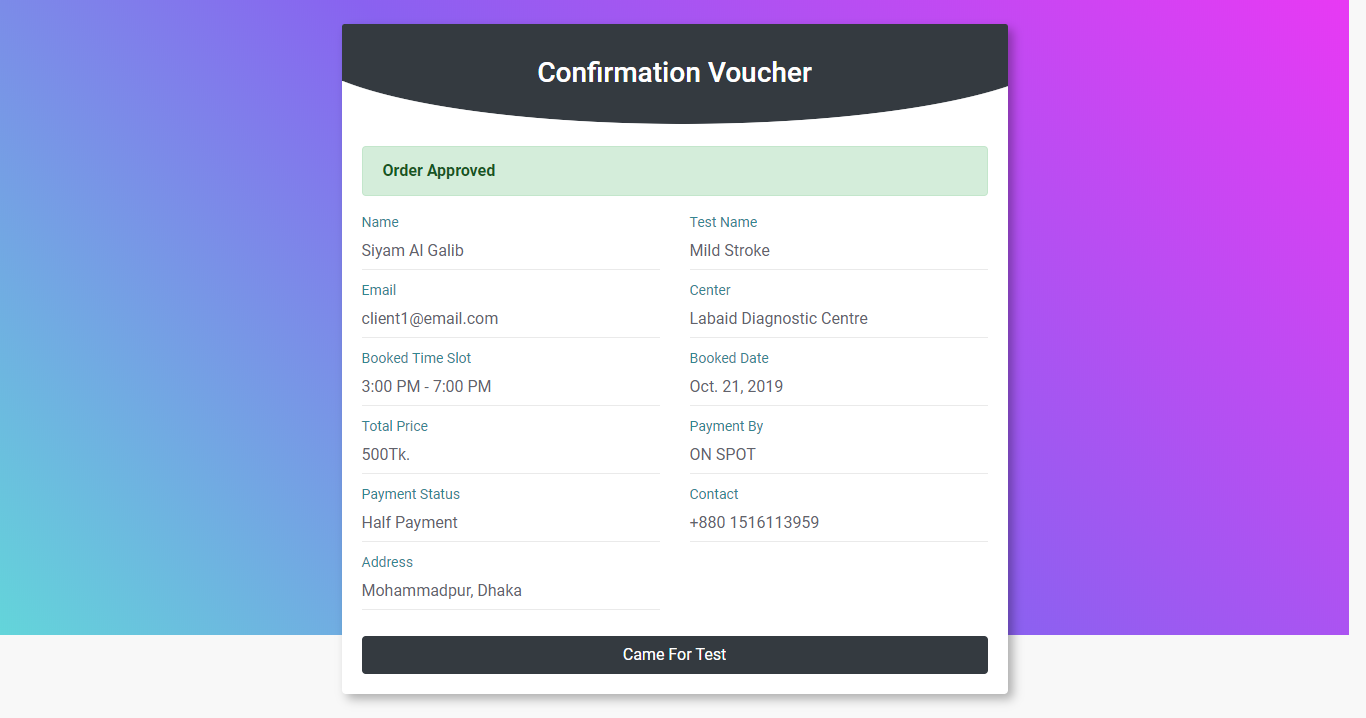
* + 1. **Diagnostic Staff Dashboard Pending Orders:**

****

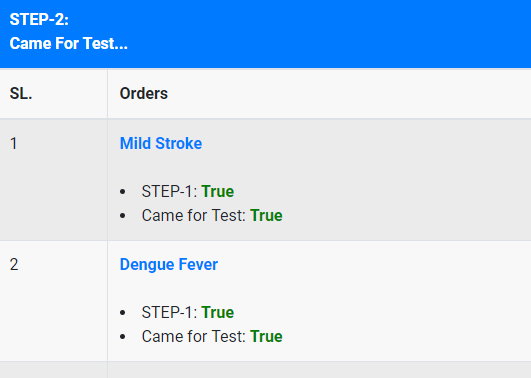
* + 1. **Diagnostic Staff Dashboard Approve or Reject Orders:**

****

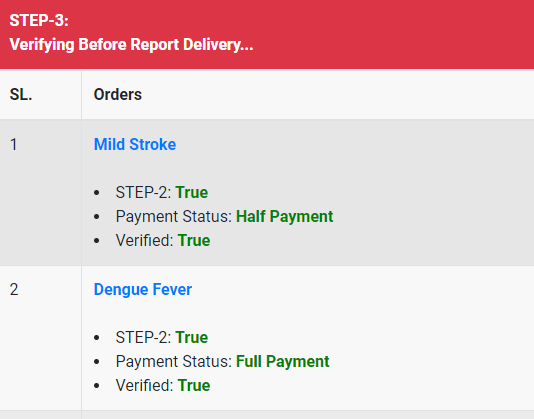
* + 1. **Diagnostic Staff Dashboard Patient Came or Not:**

****

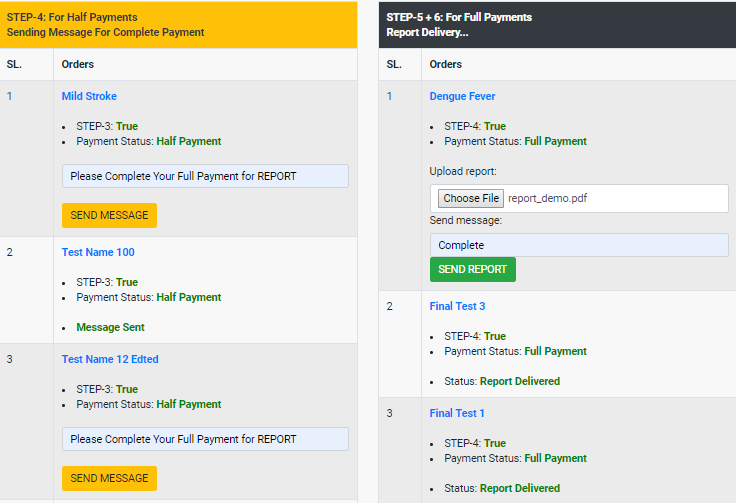
* + 1. **Diagnostic Staff Dashboard Patient Came or Not Status:**

****

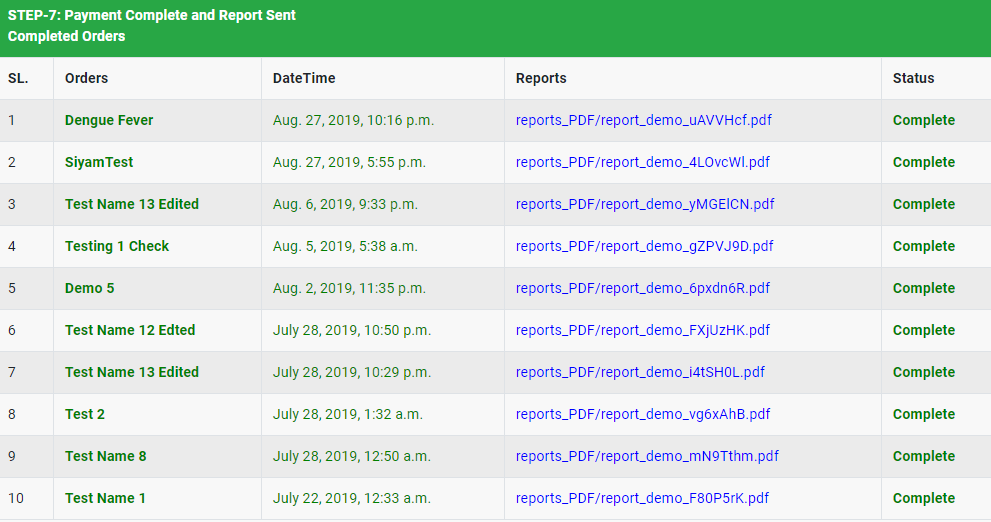
* + 1. **Diagnostic Staff Dashboard Payment Verification:**

****

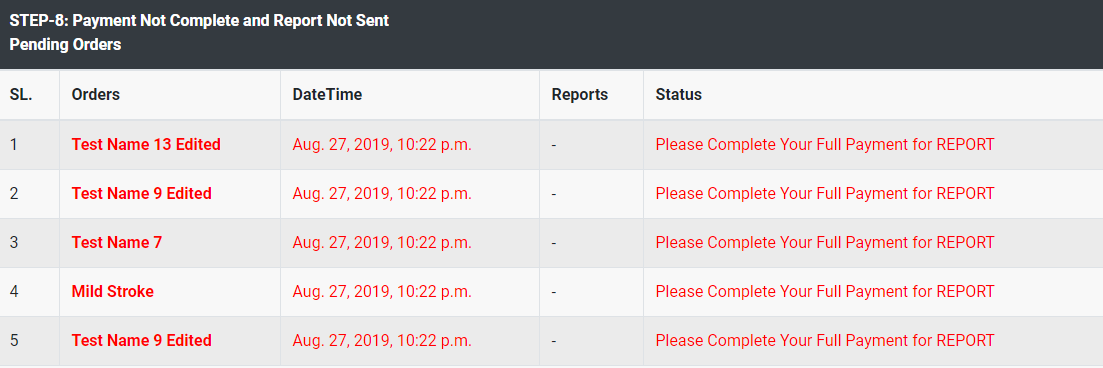
* + 1. **Diagnostic Staff Dashboard Upload Report and Sending Message:**

****

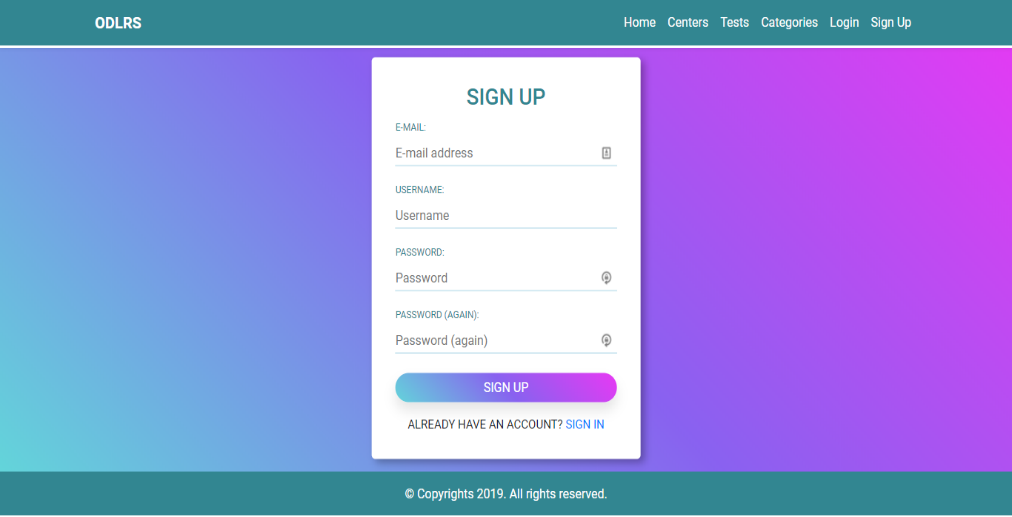
* + 1. **Diagnostic Staff Dashboard Completed Reports:**

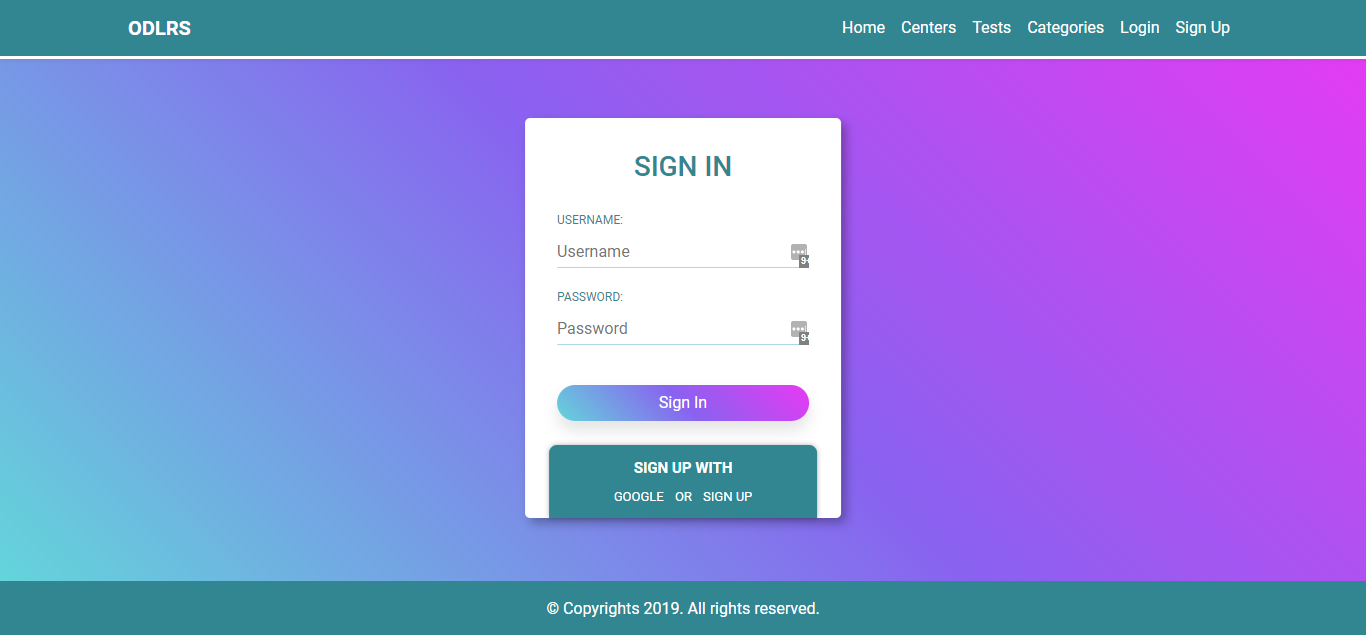
****

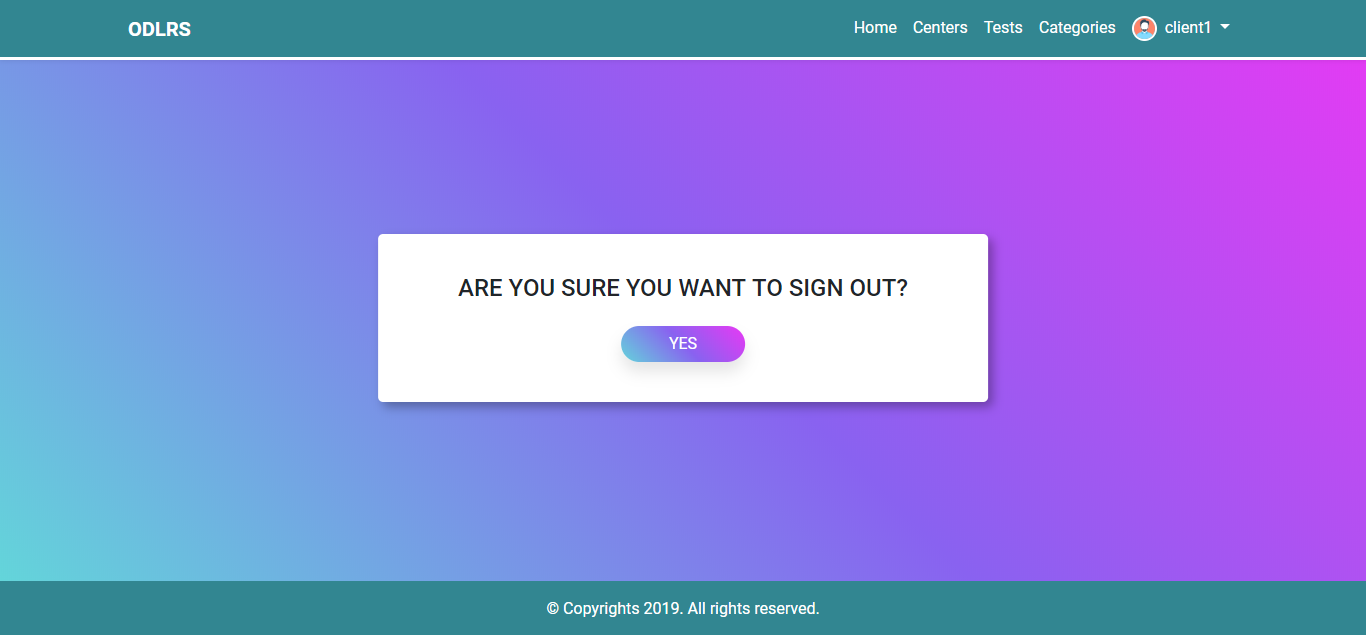
* + 1. **Diagnostic Staff Dashboard Due Payment:**

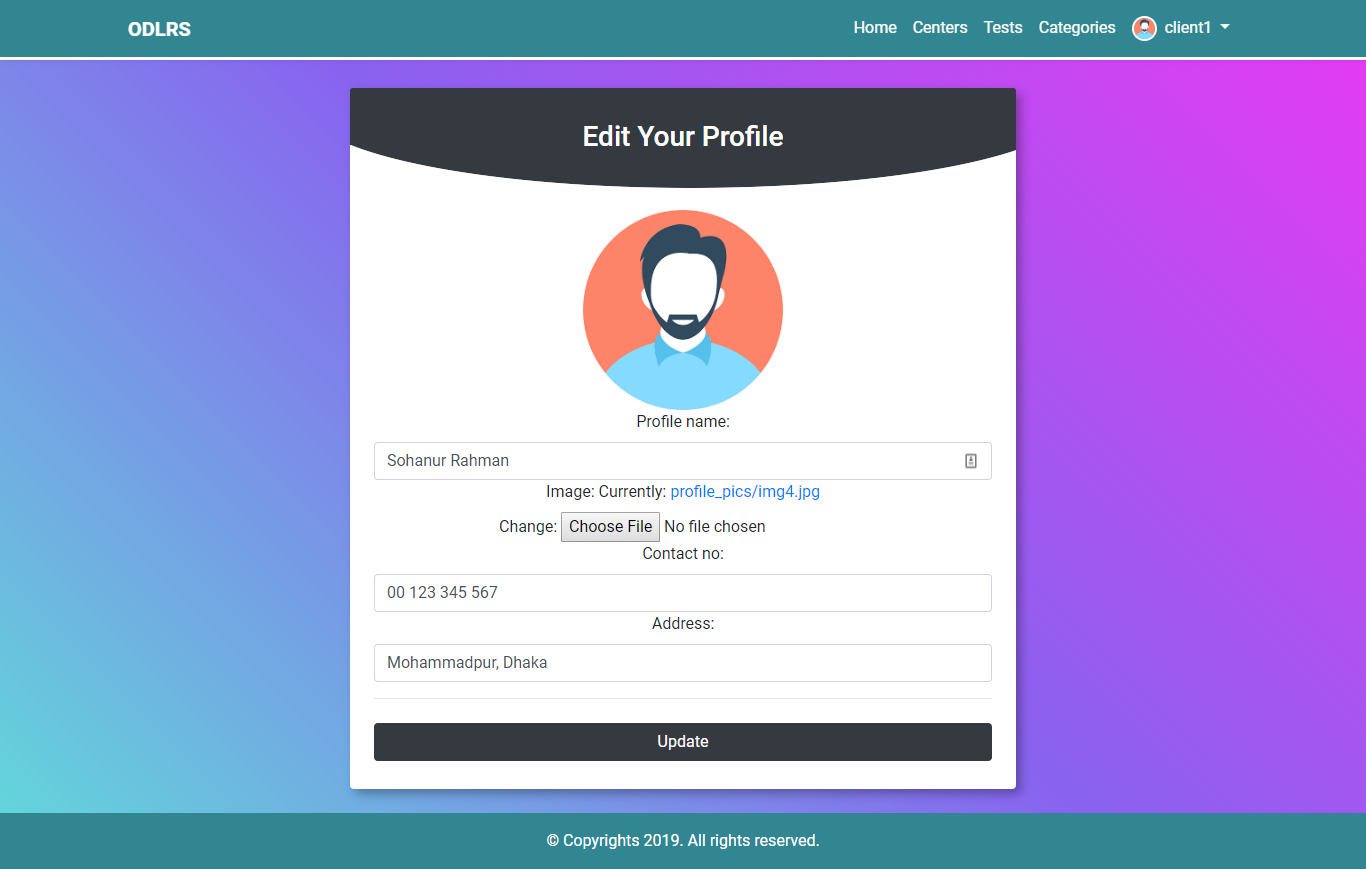
****

* + 1. **Customer Registration, Login, Logout and Edit Profile:**

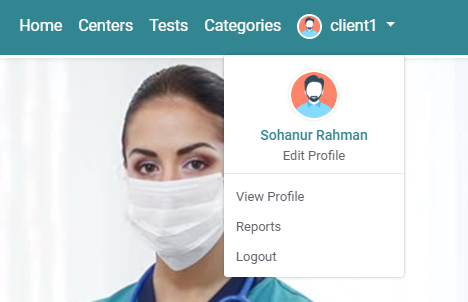
****

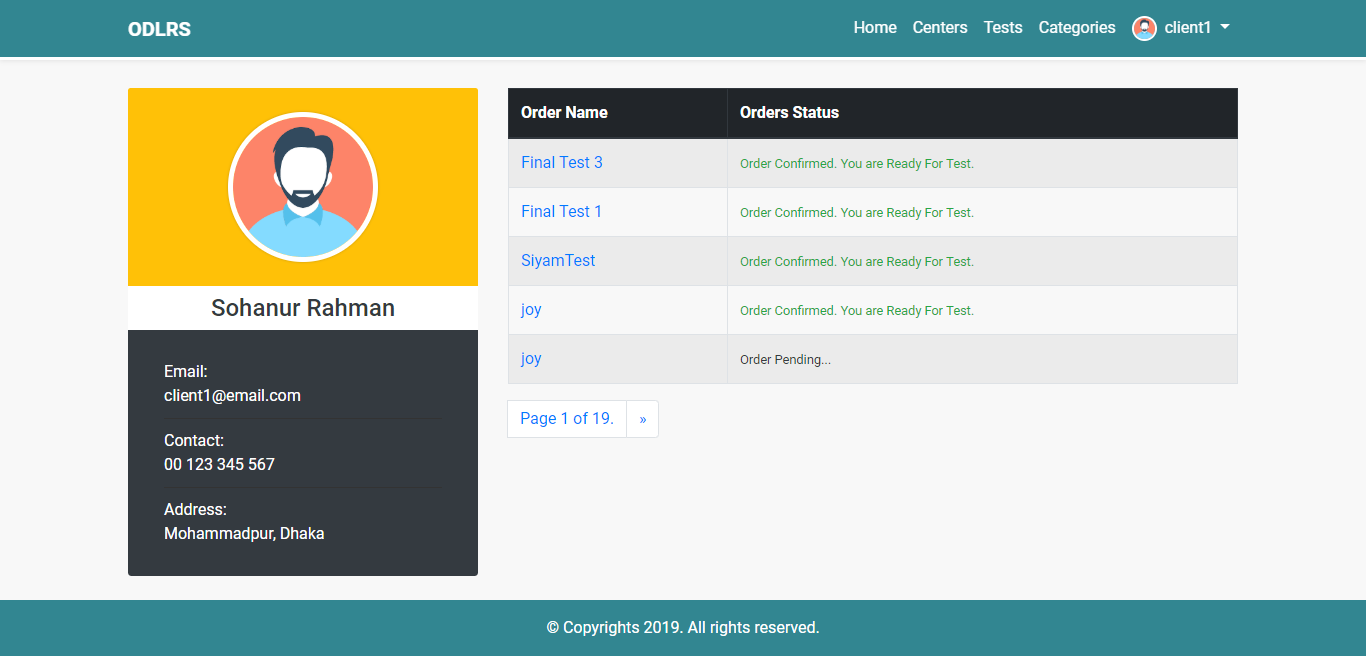
****

****

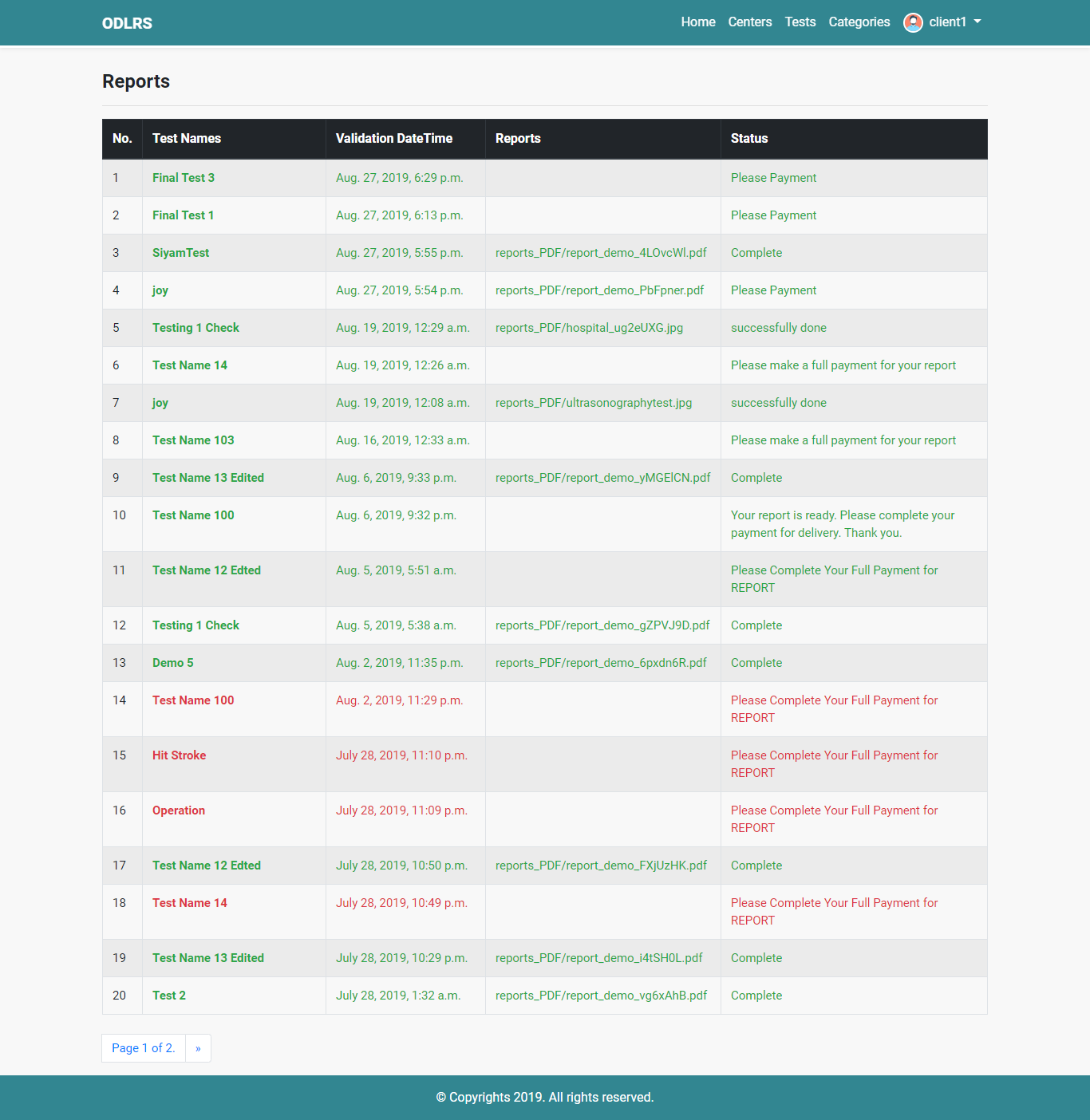
****

* + 1. **Customer Profile Details:**

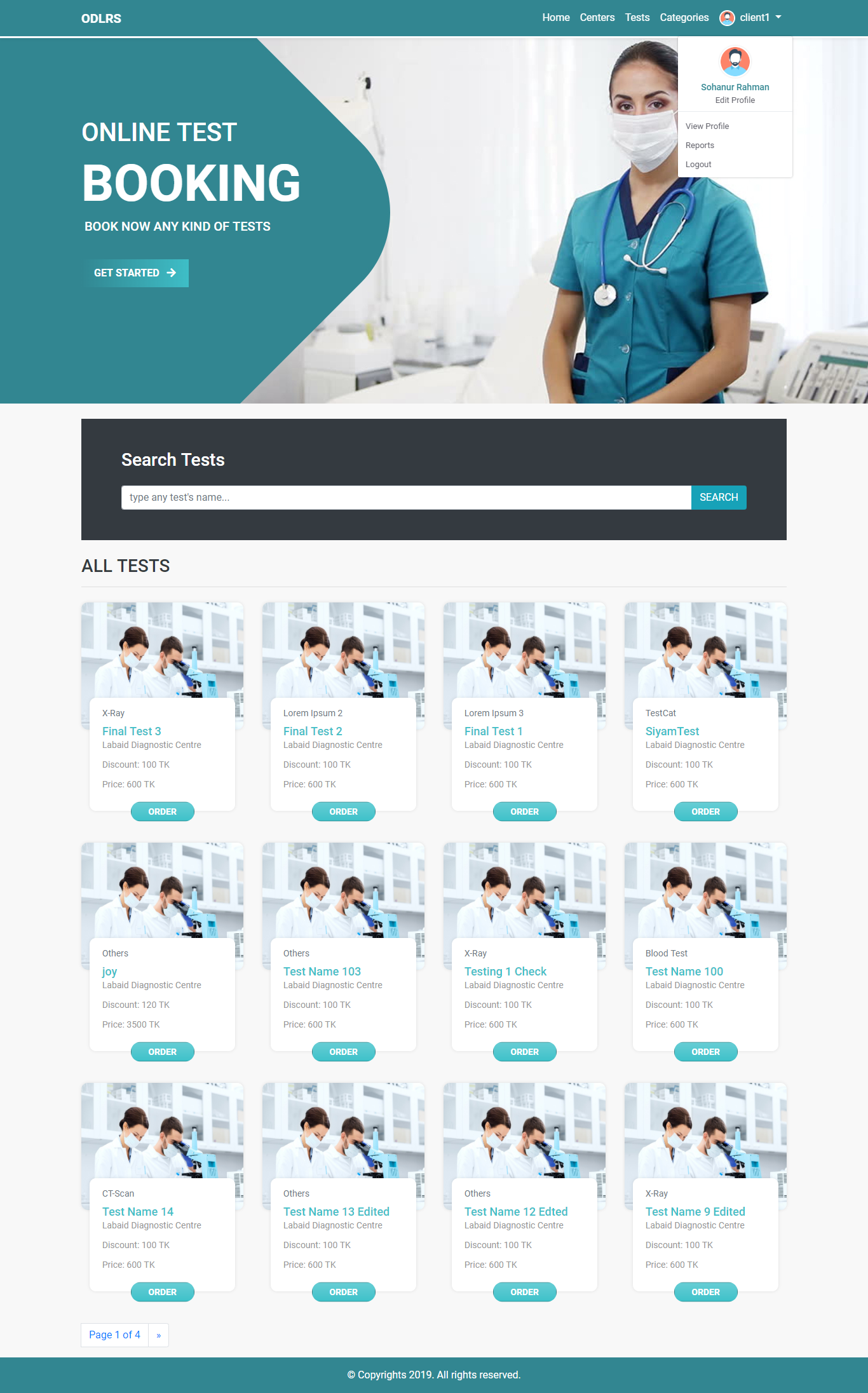
****

****

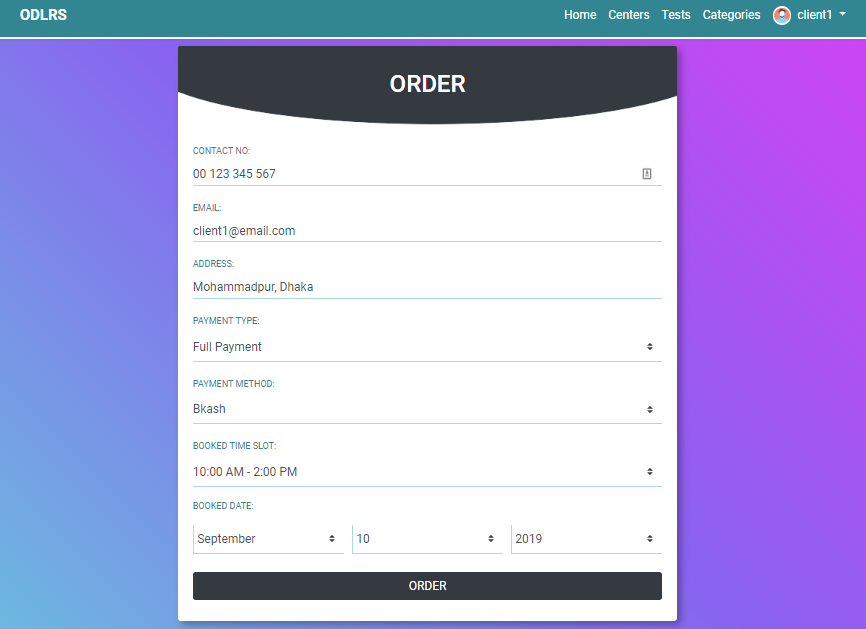
* + 1. **Customer Test Reports:**

****

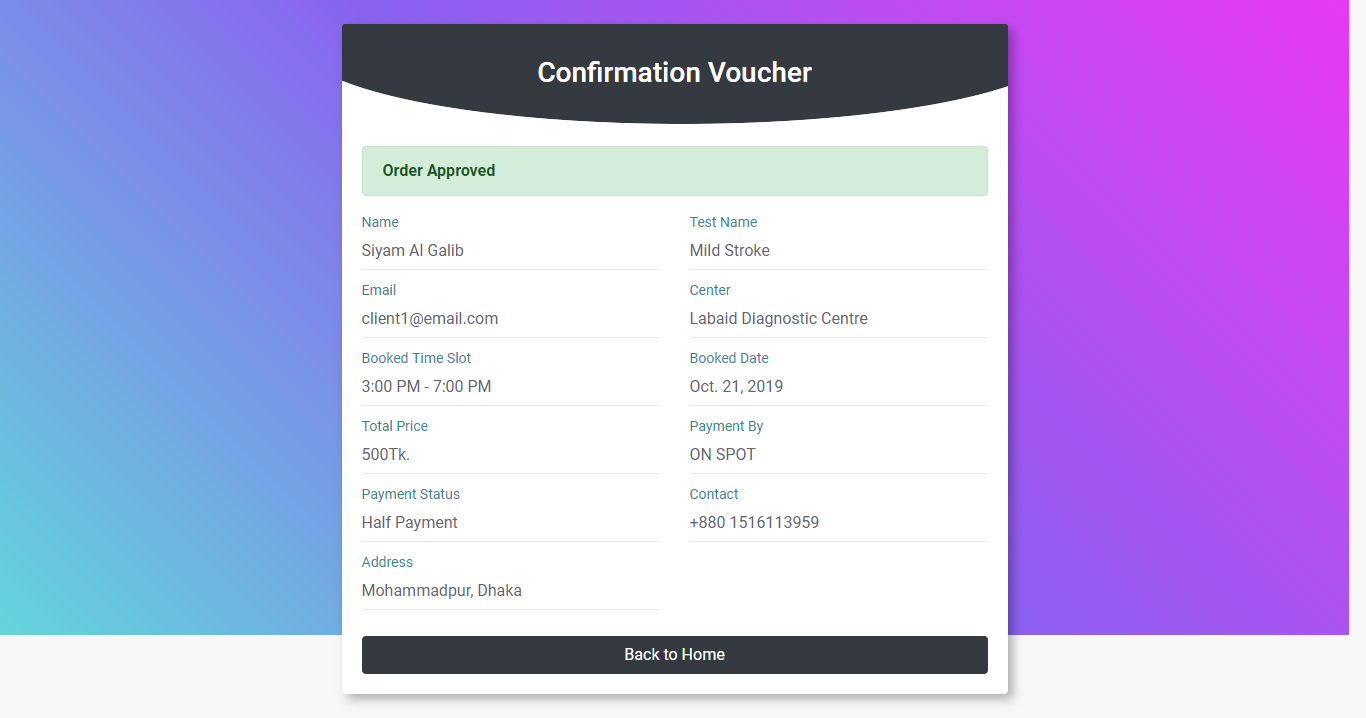
* + 1. **Homepage:**

****

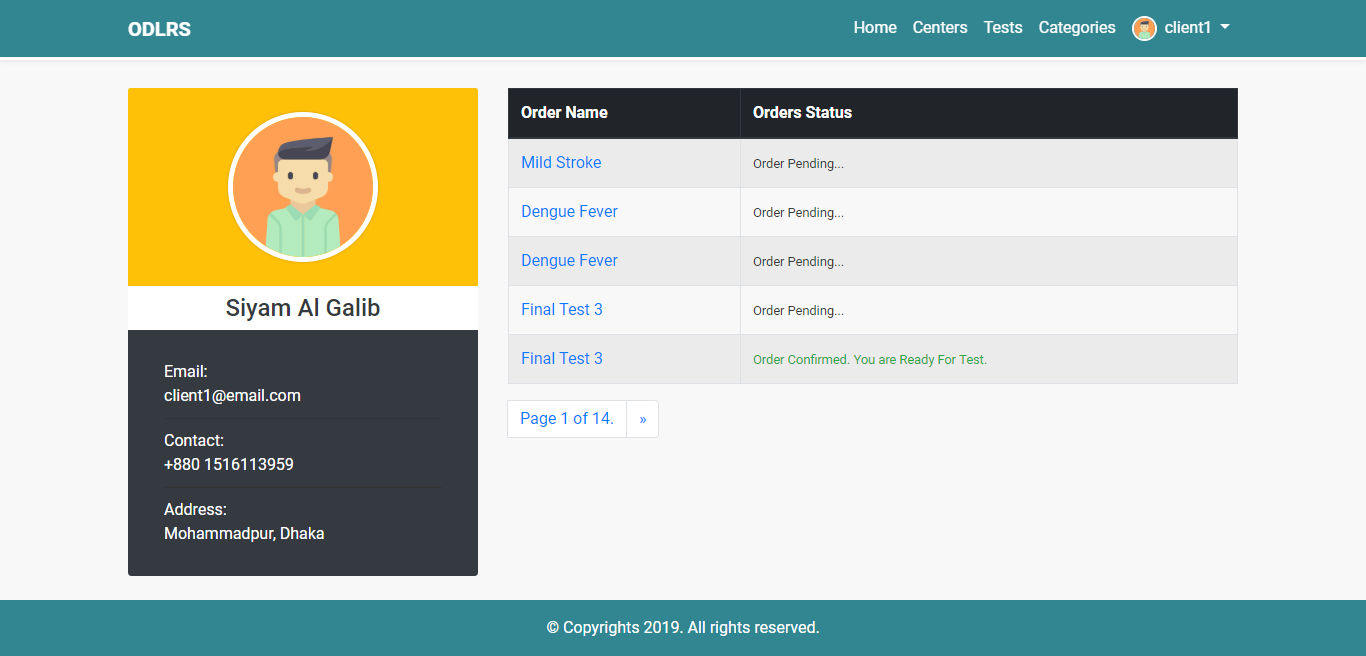
* + 1. **Order Form:**

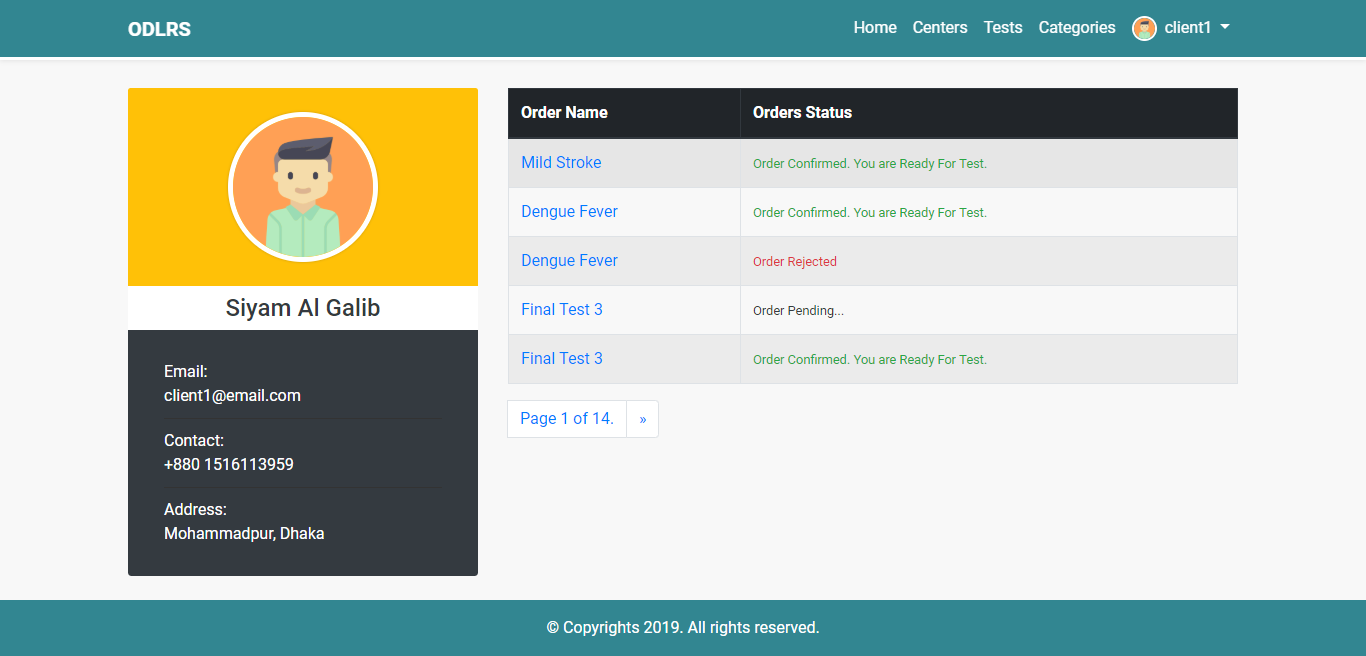
****

* + 1. **Order Details:**

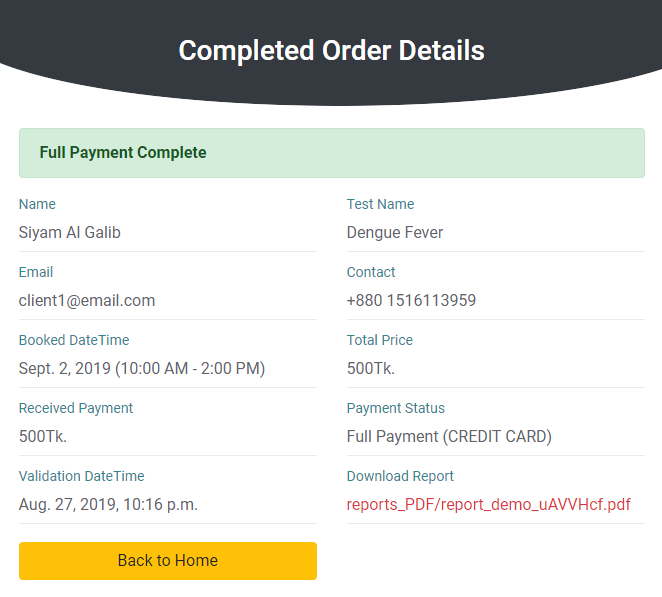
****

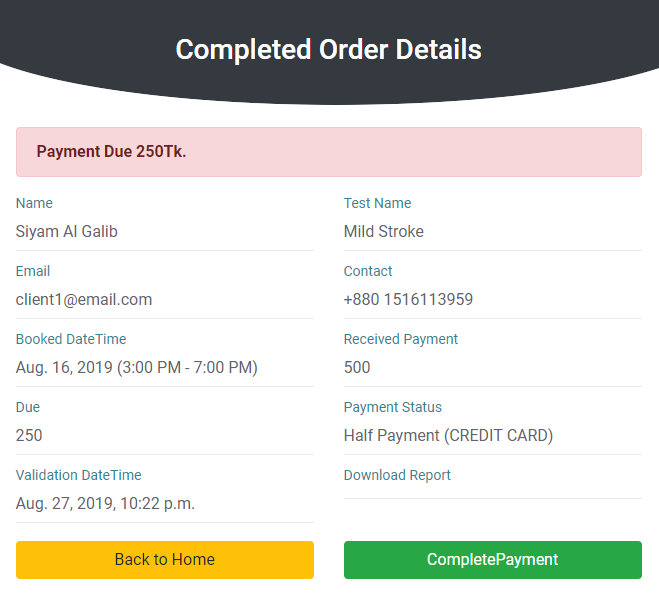
* + 1. **Order Status:**

****

****

* + 1. **Order Completed and Due Payment:**

****

****

1. 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.

* 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.

* + 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**.**
   * 1. 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.

1. 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.

*Note: This project has been temporarily deployed to a free server for testing purpose. Some media contents and other features will not be displayed because of free hosting. Please click the link below to see the live application.*

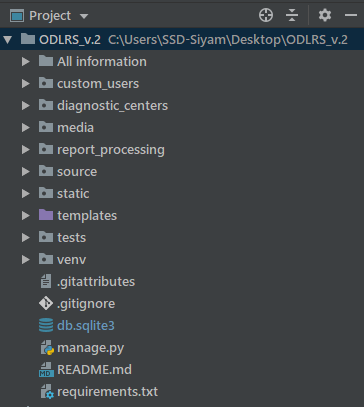
*Live Application: https://*

*This project is also available for further development and contribution at this link:* [*https://github.com/siyam04/ODLRS\_v.2*](https://github.com/siyam04/ODLRS_v.2)

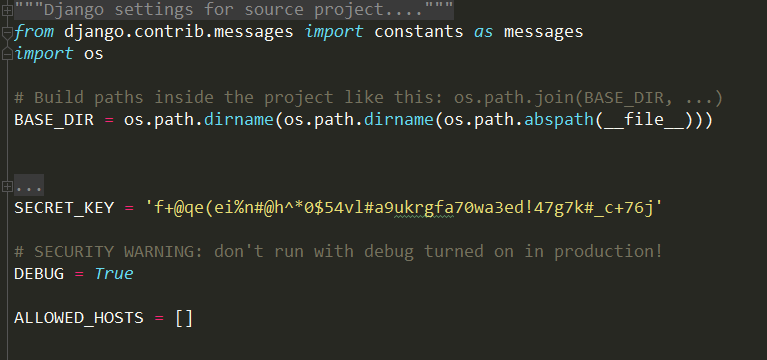
1. Chapter 7: Important Source Codes

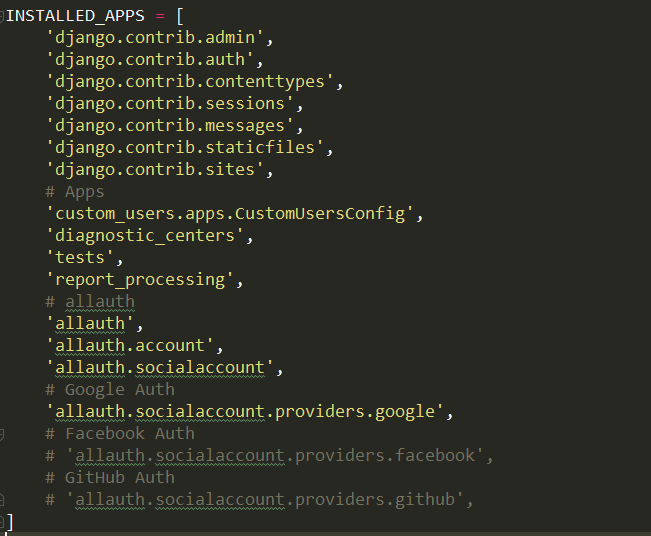
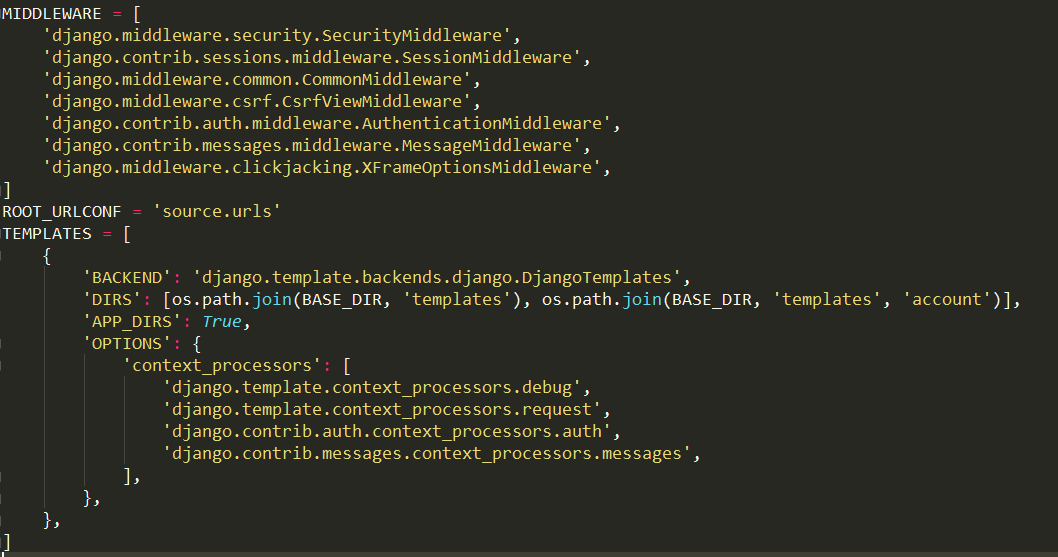
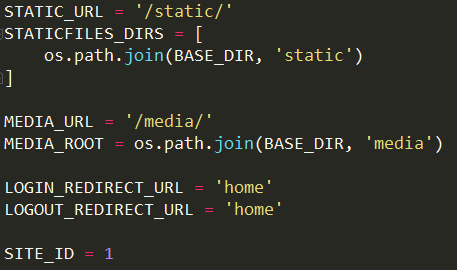
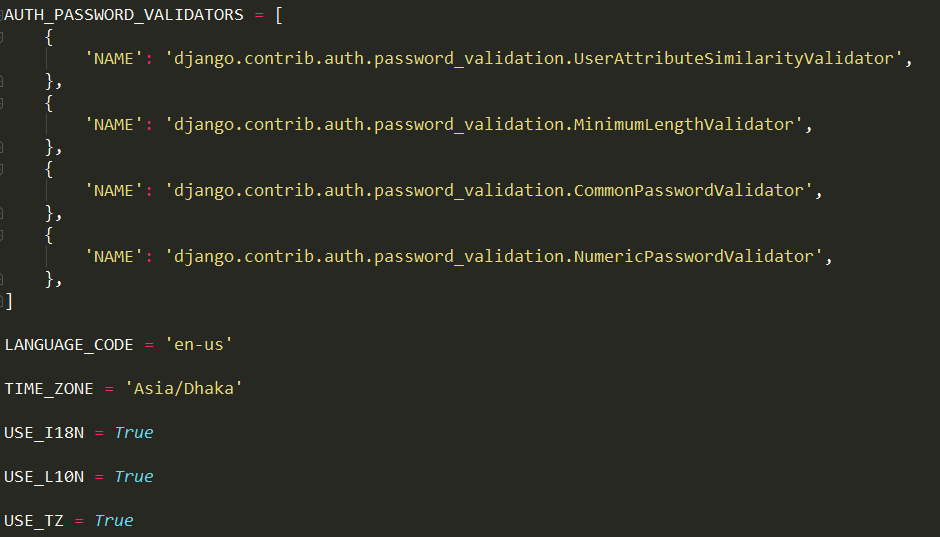
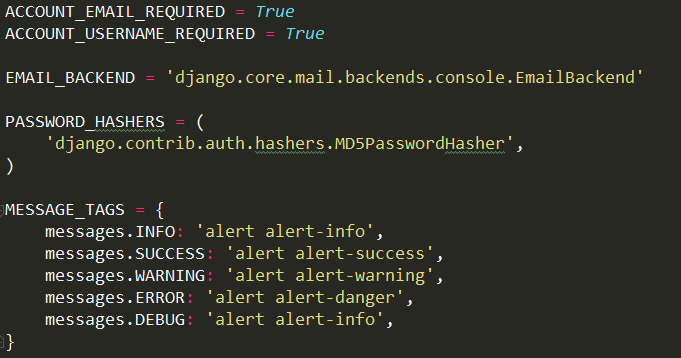
Some important source codes of the Diagnostic Lab Reporting System are given bellow:

* 1. Project Structure and Source PATHs:



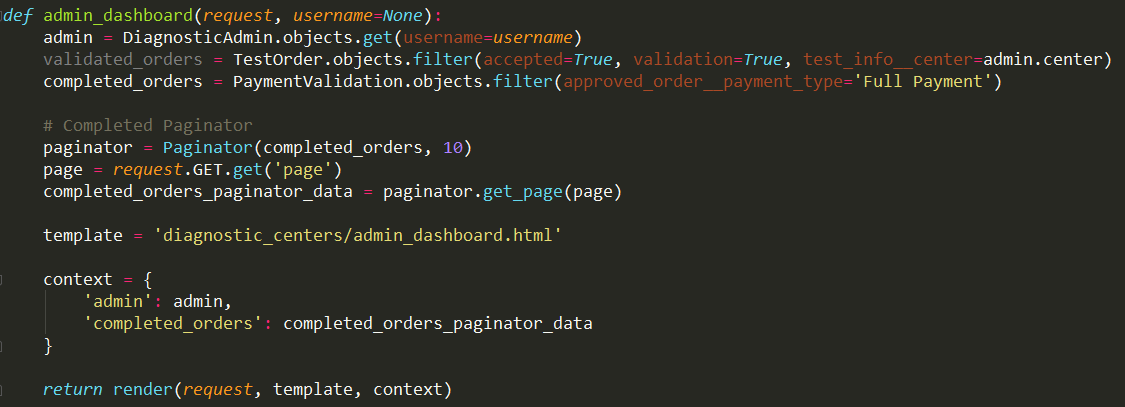
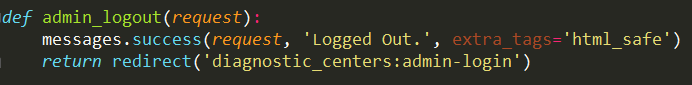
* 1. Source codes for Backend setting:



* 1. Some source codes for Diagnostic Admin Login, Logout, Dashboard:

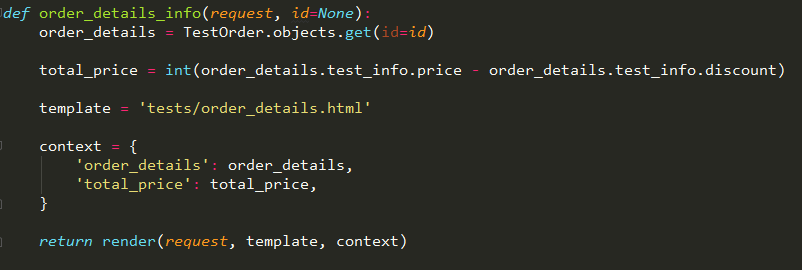


* 1. Some source codes for Add Test and Category:


* 1. Some source codes for Order System:

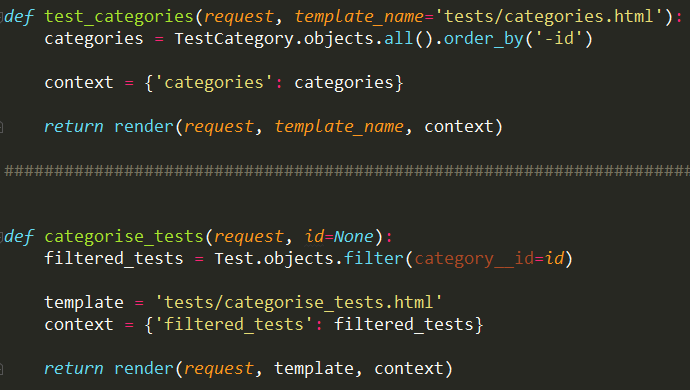




* 1. Some source codes for Reporting System:


* 1. Some source codes for Search, Pagination, Categories, Profile Edit, and Delete Tests:






* 1. All PATHs for Customer, Diagnostic Center, Test, and Report Applications:





8 Chapter 8: References

1. <https://www.draw.io/>
2. <https://www.figma.com/>
3. <https://app.sqldbm.com/>
4. <https://www.python.org/>
5. [https://getbootstrap.com](https://getbootstrap.com/)
6. [https://stackoverflow.com](https://stackoverflow.com/)
7. [https://www.youtube.com](https://www.youtube.com/)
8. <https://www.behance.net/>
9. <https://www.wikipedia.org/>
10. <https://www.w3schools.com/>
11. <http://www.pharmaaid.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.digitalocean.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>