PATIENT APPOINTMENT BOOKING

A PROJECT REPORT

Submitted By

Mudit Kumar
2100290140088
Shipra Aggarwal
2100290140125
Gaurav Srivastav
2100290140065

**Submitted in partial fulfilment of the
Requirements for the Degree of**

MASTER OF COMPUTER APPLICATION

**Under the Supervision of
Dr. Vipin Kumar
Associate Professor**

Submitted to

**DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206**

(JUNE 2023)

DECLARATION

This is to declare that the project entitled "**PATIENT APPOINTMENT BOOKING**" submitted by us in partial fulfilment of the requirements for the award of the degree of Master of Computer Application, in the department of MCA of Dr. APJ Abdul Kalam Technical University, is a bona-fide record of the project work carried out by us in college KIET Group of Institutions, Ghaziabad during the period of fourth semester under the supervisor **Dr. Vipin Kumar (Associate Professor)** and that it has not been submitted previously by us at any other university for the award of any degree.

Mudit Kumar-2100290140088
Shipra Aggarwal-2100290140125
Gaurav Srivastav-2100290140065
Branch: Master of Computer Application

(Candidate Signature)

CERTIFICATE

Certified that **Mudit Kumar (2100290140088), Shipra Aggarwal (2100290140125), Gaurav Srivastav (2100290140065)** has/ have carried out the project work having "**Patient Appointment Booking"** for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Mudit Kumar (2100290140088)
Shipra Aggarwal (2100290140125)
Gaurav Srivastav(2100290140065)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Vipin Kumar
(Associate Professor)
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Arun Tripathi
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

The purpose of Patient Appointment Booking is to automate the existing manual system by the help of computerized equipment's and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

Patient Appointment Booking, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

The aim is to automate its existing manual system by the help of computerized equipment's and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. Basically the project describes how to manage for good performance and better services for the clients.

In such issues, patients in a hospital environment need to visit various resource forms sequentially so that they can undergo or be diagnosed with medication. Therefore a particular route over a subset of the assets deemed is allocated to each patient and each move needs to be planned. The primary objective of these problems is to allow the services in their subset to be visited by each individual within the allocated time.

Additionally, with hospital appointment scheduling, hospitals have the occasion to supplement patient fulfilment, allowing the enduring to appointment the hospital less repeatedly. A classification proposal is projected and categorize on hospital appointment scheduling in hospitals published before the end of 2020. The explanations show that hospital appointments are attractive gradually more admired. In fact, the appointment arrangement problems in hospitals are now in advance gradually more drive in the intellectual literature.

ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Dr. Vipin Kumar** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Mudit Kumar (2100290140088)

Shipra Aggarwal (2100290140125)

Gaurav Srivastav (2100290140065)

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1. Introduction

If anybody is ill and wants to visit a doctor for check-up, he or she needs to visit the hospital and waits until the doctor is available. The patient also waits in a queue while getting appointment. If the doctor cancels the appointment for some emergency reasons then the patient is not able to know about the cancelation of the appointment unless or until he or she visits the hospital. As the mobile communication technology is developing rapidly, therefore, one can use the mobile's applications to overcome such problems and inconvenience for the patients. The proposed work in this paper is an Online Hospital Management Application that uses an android platform that makes the task of making an appointment from the doctor easy and reliable for the users. Android based online doctor appointment application contains two modules. One module is the application designed for the patient that contains a login screen. The patient has to register himself before logging in to the application. After logging in, the patient can select a hospital and can view the hospital details. The patient has the option of selecting a doctor from the list of doctors and can view the doctor's details. The patient can request for an appointment on his/her preferred day/time. The selected day/time slot will be reserved and patient will receive the notification of the successfully added appointment. The patient can view the location of the hospital on map. In addition, the patient can contact to the hospital and the doctor by making a call or may send an email to the doctor. There are considerable online scheduling tools in the internet, a few of which are trait loaded, simple to setup and economical For practitioners, online appointment reservation and scheduling delivers a lot of merit added benefits and services, like captivating the patient, composing the patient to feel welcomed, and being capable to save patients' details safely for future information. But the most admirable and useful preference is that online appointment reservation and scheduling is remarkably in expensive .Both doctors and patients can access the portal through their unique ID's.

1.1 PROJECT SCOPE

The scope of this project will focus on the patient, doctors and hospitals who will use the system to make and manage the appointment via online services. This project will be implemented and useful for all doctors and patient. The doctor and hospital to manage the appointment for the patients those who would like to have the appointment for the doctor in specific place date via online system anytime and anywhere. And this system does, after the patient arrive hospital, nurse collects the patient's information and sends it to the doctor before the patient is diagnosed.

The system has been facing problems due to its paper-based appointment system. With the increase in the number of patients visiting, it has become difficult to manage the appointment system manually. Recording of appointments and creating registers by pen and paper has become a tedious task. And also it's difficult to manage huge number of patient database.

This online web application gives solution to the patients and employees. This system which manages complete details in a single application and in a single database. The users will use this system to handle all the functionalities easily. Doctors will also use the system to keep track of the patients consulting to them. The intentions of the system are to reduce over-time pay and increase the number of patients that can be treated accurately..

- We can add printer in future.
- We can give more advance software for Patient Appointment Booking including more facilities
- We will host the platform on online servers to make it accessible worldwide
- Integrate multiple load balancers to distribute the loads of the system
- Create the master and slave database structure to reduce the overload of the database queries
- Implement the backup mechanism for taking backup of codebase and database on regular basis on different servers

1.2 PROJECT DESCRIPTION

The proposed project is a smart appointment booking system that provides patients or any user an easy way of booking a doctor’s appointment online. This is a web based application that overcomes the issue of managing and booking appointments according to user’s choice or demands. The task sometimes becomes very tedious for the compounder or doctor himself in manually allotting appointments for the users as per their availability.

Hence this project offers an effective solution where users can view various booking slots available and select the preferred date and time. The already booked space will be marked yellow and will not be available for anyone else for the specified time. This system also allows users to cancel their booking anytime. The system provides an additional feature of calculating monthly earnings of doctor. Doctor has to just feed the system regularly with daily earnings and the system automatically generates a report of total amount earned at the end of the month. The application uses Asp.net as a front-end and sql database as the back-end

1. Login Module: Used for managing the login details
2. Users Module: Used for managing the users of the system
3. Doctor Module: Used for managing the Doctor details and checking appointments appointed to doctors.
4. Medicine Module: Used for managing the details of Medicine and the medicine prescribed to patients
5. Appointment Management Module: Used for managing the information and details of the Appointment.
6. Patient Module: Used for managing the Patient details and patient appointment history
7. Booking Module: Used for managing the appointment booking information and appointment details of the patients

1.3 TECHNOLOGIES

1.3.1 SOFTWARE REQUIREMENTS

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view.

Language	React JS, Java
Database	MySQL Server
Browser	Any Browser
Scripting Language Enable	Node JS (Node JavaScript)
Database JDBC Driver	MySQL Jconnector

1.3.2 HARDWARE REQUIREMENTS

The hardware requirements are the requirements of a hardware device. Most hardware only has operating system requirements or compatibility. For example, a printer may be compatible with Windows XP but not compatible with newer versions of Windows like Windows 10, Linux, or the Apple macOS.

Name of component	**Specification**
Processor	Pentium 4 or Above
Ram	4GB RAM
Hard Disk	20GB
Monitor	15" Colour monitor
Keyboard	122 keys need

CHAPTER 2

FEASIBILITY STUDY

2. Feasibility study

This feasibility study aims to assess the viability and potential success of implementing a patient appointment system for a healthcare facility. The patient appointment system is designed to streamline and optimize the process of scheduling appointments, reducing wait times, improving patient satisfaction, and enhancing overall operational efficiency. The primary objective of this feasibility study is to evaluate the technical, economic, operational, and schedule feasibility of implementing a patient appointment system. The study will analyze the benefits, costs, risks, and potential challenges associated with the proposed system.

2.1 Technical feasibility

- Evaluate the technical requirements of the patient appointment system, including software and hardware components.
- Assess the compatibility of the system with existing IT infrastructure and any necessary integration.
- Determine if the required technical expertise is available or can be acquired to develop, deploy, and maintain the system effectively.

2.2 Operational feasibility

- Analyze the impact of the patient appointment system on existing workflows and processes.

- Identify any potential challenges or resistance from staff members and develop strategies for addressing them.
- Evaluate the scalability and adaptability of the system to accommodate future growth and changing requirements.

2.3 Behavioral feasibility

Behavioral feasibility is an important aspect to consider when implementing a patient appointment system in a healthcare facility. It refers to the likelihood that the system will be accepted and used by its intended users, in this case, the healthcare staff and the patients. The following is a description of the behavioral feasibility of a patient appointment system.

2.4 Economical feasibility

This is a very important aspect to be considered while developing a project. We decided the technology based on minimum possible cost factor.

- All hardware and software cost has to be borne by the organization.
- Overall we have estimated that the benefits the organization is going to receive from the proposed system will surely overcome the initial costs and the later on running

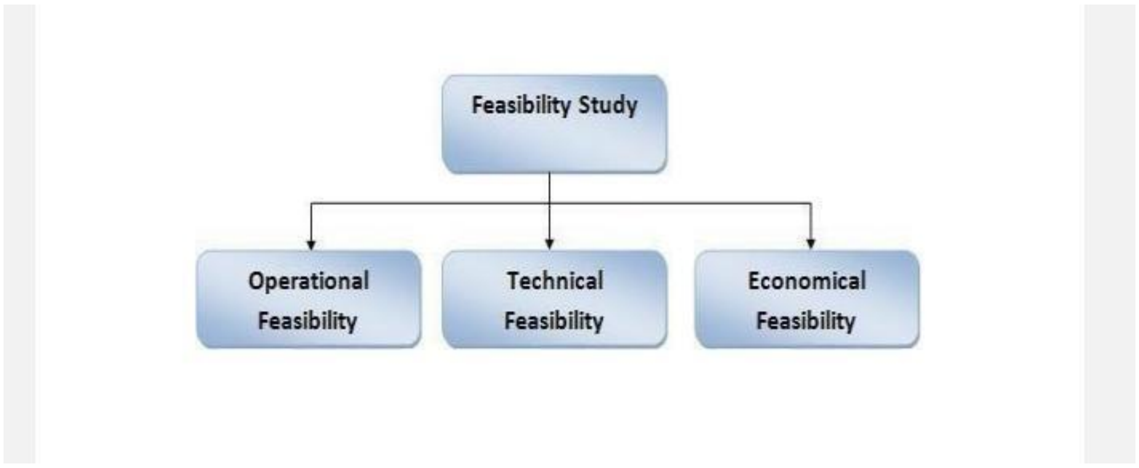


Figure 2.1 Feasibility Study

CHAPTER 3

DATABASE DESIGN

3.1 What is UML?

UML stands for Unified Modeling Language is the successor to the wave of Object Oriented Analysis and Design (OOA&D) methods that appeared in the late 80's. It most directly unifies the methods of Booch, Rumbaugh (OMT) and Jacobson. The UML is called a modeling language, not a method. Most methods consist at least in principle, of both a modeling language and a process, The Modeling language is that notation that methods used to express design.

3.2 Notations and Meta-Models:

The notation is the graphical stuff; it is the syntax of the modeling language. For instance, class diagram notation defines how items are concepts such as class. association, and multiplicity is represented. These are:

3.2.1 USE CASE DIAGRAM

Elicit requirement from users in meaningful chunks. Construction planning is built around delivering some use cases n each interaction basis for system testing.

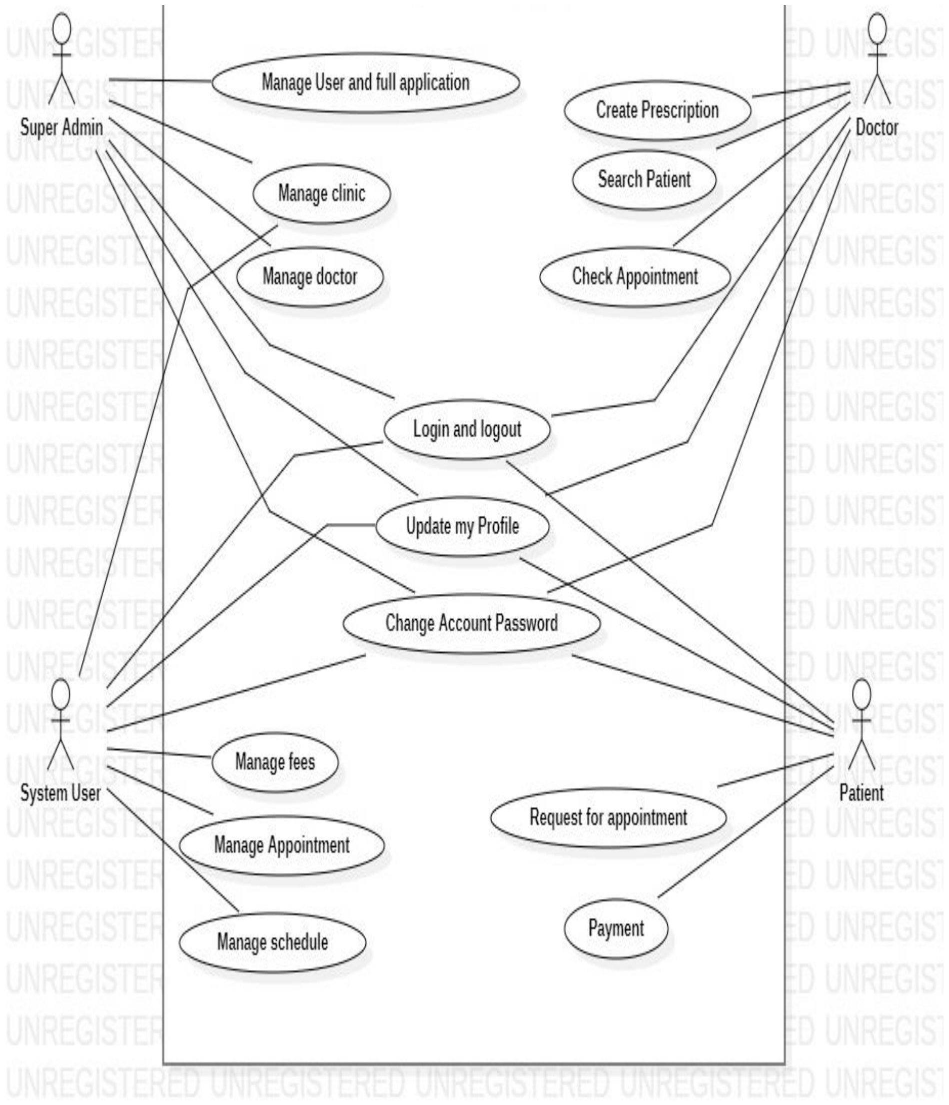


Figure 3.1 Use Case Diagram of Patient Appointment Booking

3.2.2 ACTIVITY DIAGRAM

Activity diagram shows behavior with control structure. Can show many objects over many uses, many object in single use case, or implementations methods encourage parallel behavior, etc.

- Test Activity
- Schedule Activity
- Doctors Activity
- Appointment Activity
- Fees Activity

Features Of The Activity UML Diagram Of Patient Appointment Booking

- Admin User can search Test view description of a selected Test add Test, update Test and delete Test
- Its shows the activity flow of editing: adding and updating of schedule User will be able to search and generate report of Doctors Appointment, Fees
- All objects such as (Test schedule, Fees) are interlinked Its snows the full description and now of Test Appointment Fees Doctors, schedule

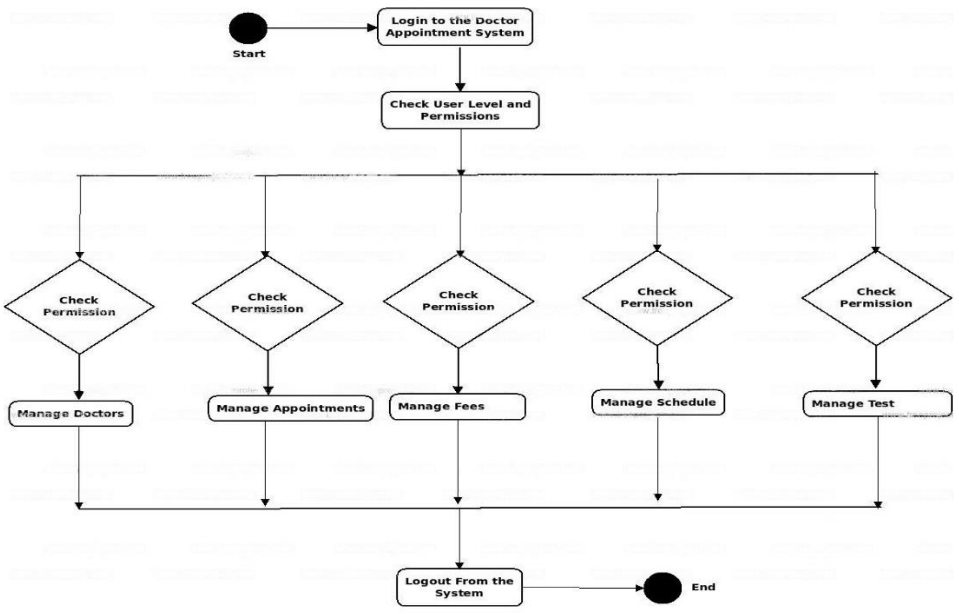


Figure 3.2 Activity Diagram of Patient Appointment Booking

3.2.3 CLASS DIAGRAM

It shows static structure of concepts, types and class. Concepts how users think about the world; type shows interfaces of software components; classes shows implementation of software components. Patient Appointment Booking Class Diagram describes the structure of a Patient Appointment Booking classes, their attributes, operations (or methods) and the relationships among objects. The main classes of the Patient Appointment Booking are Nurse, Doctor , Appointment ,Receptionist, Patient, Prescription

Classes of Patient Appointment Booking Class Diagram:

- Doctors Class Manage all the operations of Doctors Appointment Class Manage all the operations of Appointment
- Nurse Class Manage all the operations of nurse
- Appointment Class Manage all the operations of appointment

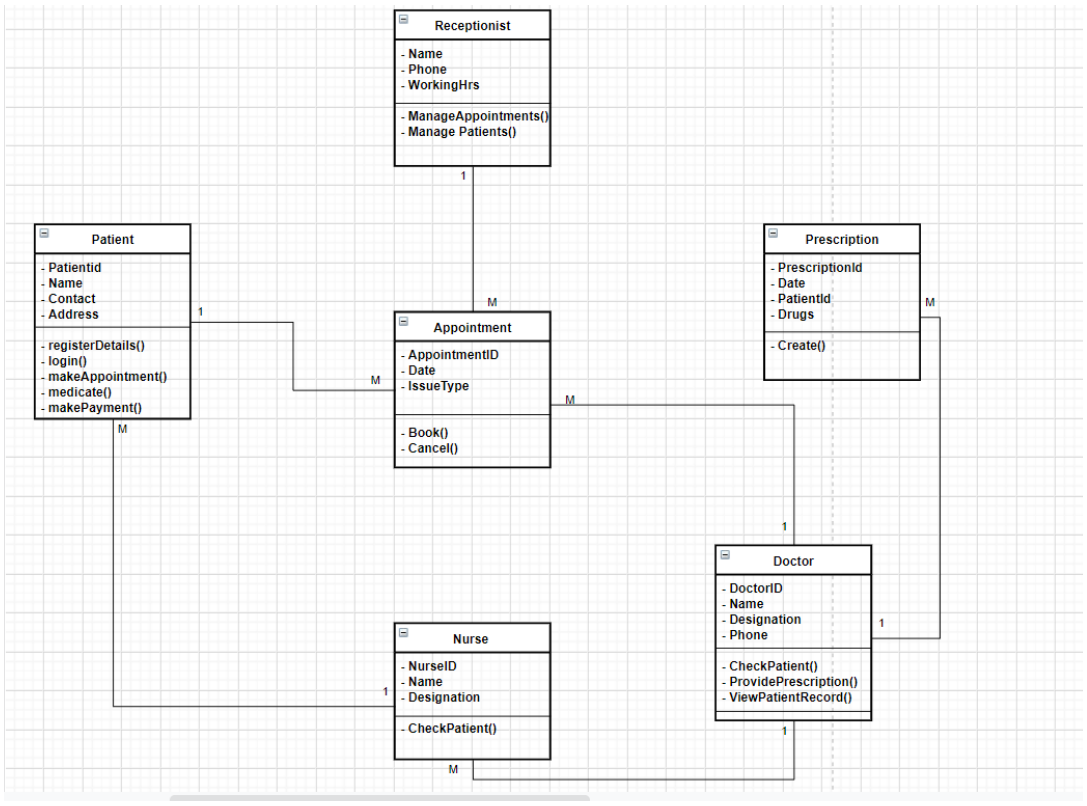


Figure 3.3 Class Diagram for Patient Appointment Booking

3.2.4 E-R DIAGRAM

This ER (Entity Relationship) Diagram represents the model of Patient Appointment Booking Entity The entity-relationship diagram of Patient Appointment Booking shows all the visual instrument of database tables and the relations between Appointment, Fees, Doctors, Clinics etc. It used structure data and to define the relationships between structured data groups of Patient Appointment Booking functionalities. The main entries of the Patient Appointment Booking are Doctors. Appointment Booking, Fees, schedule and Clinics.

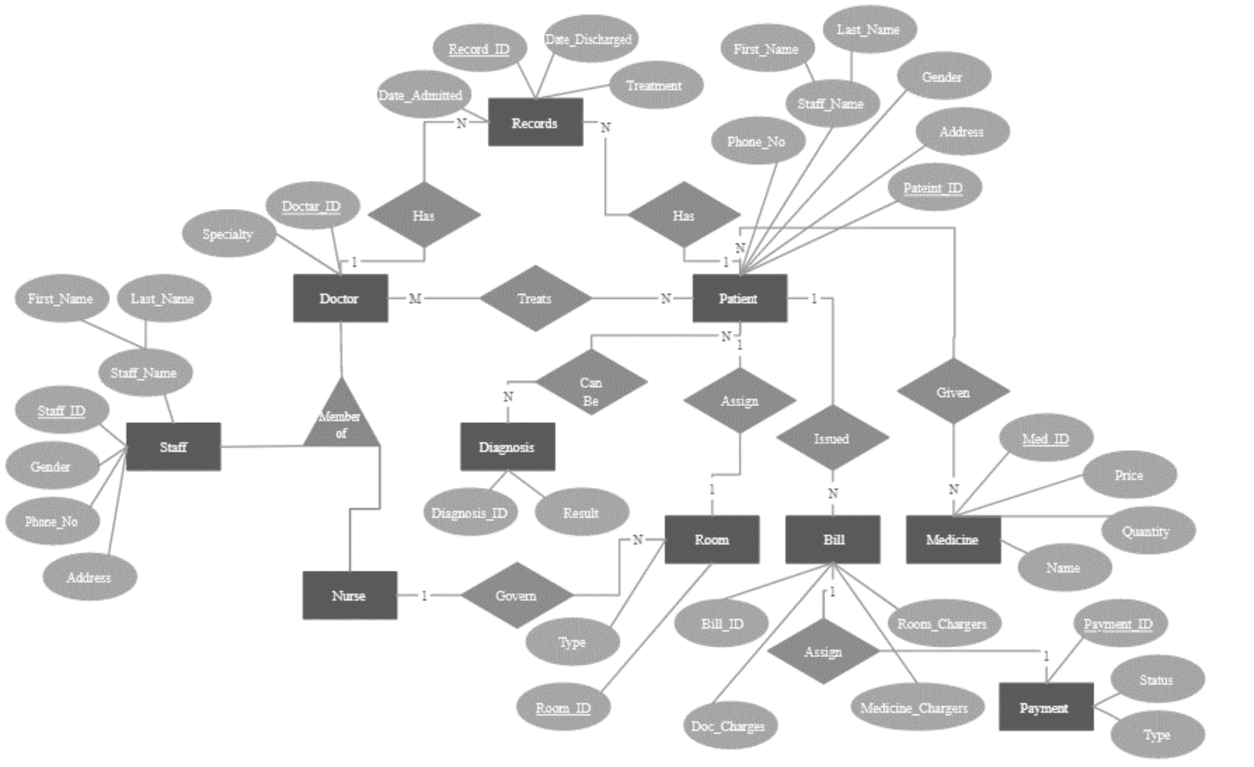


Figure 3.4 E-R Diagram for Patient Appointment Booking

3.2.5 DFD DIAGRAM

Patient Appointment Booking Data now diagram is often used as a preliminary step to create an overview of the Doctor Appointment without going into great detail, which can later be elaborated it normally consists of overall application dataflow and processes of the Doctor Appointment process.

Zero Level Data Flow Diagram(0 Level DFD) Of Patient Appointment Booking:

This is the Zero Level DFD of Patient Appointment Booking, where we have elaborated the high level process of Doctor Appointment. It's a basic overview of the whole Patient Appointment Booking or process being analyzed or modeled its designed to be an at-a-glance view of Timeslot Patent and Medicine showing the system as a single high-level process.

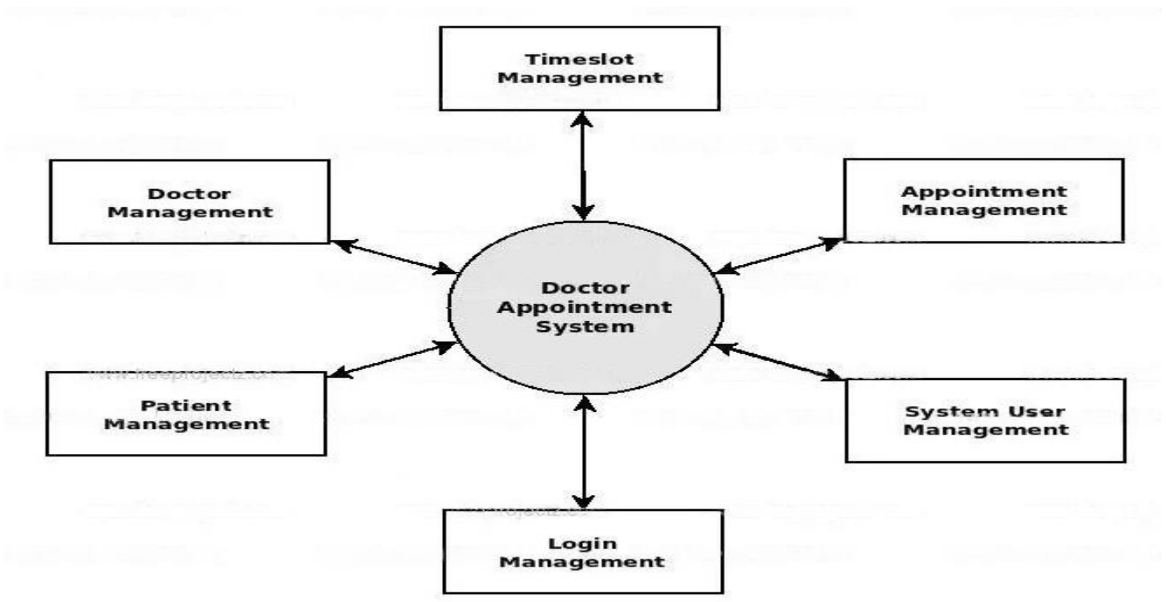


Figure 3.5 Zero Level DFD of Patient Appointment Booking

First Level Data Flow Diagram(1st Level DFD) Of Patient Appointment Booking:

First Level DFD (1st Level) of Patient Appointment Bookingshows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the Patient Appointment Bookingsystem as a whole It also identities internal data stores of Medicine, Patient Timeslot, Patient, Booking that must be present in order for the Patient Appointment Bookingto do its job, and shows the flow of data between the various parts of Doctor, Booking, Patient, Medicine Timeslot of the system DFD Level 1 provides a more detailed breakout of pieces of the 1st level DFD You will highlight the main functionalities of Doctor Appointment

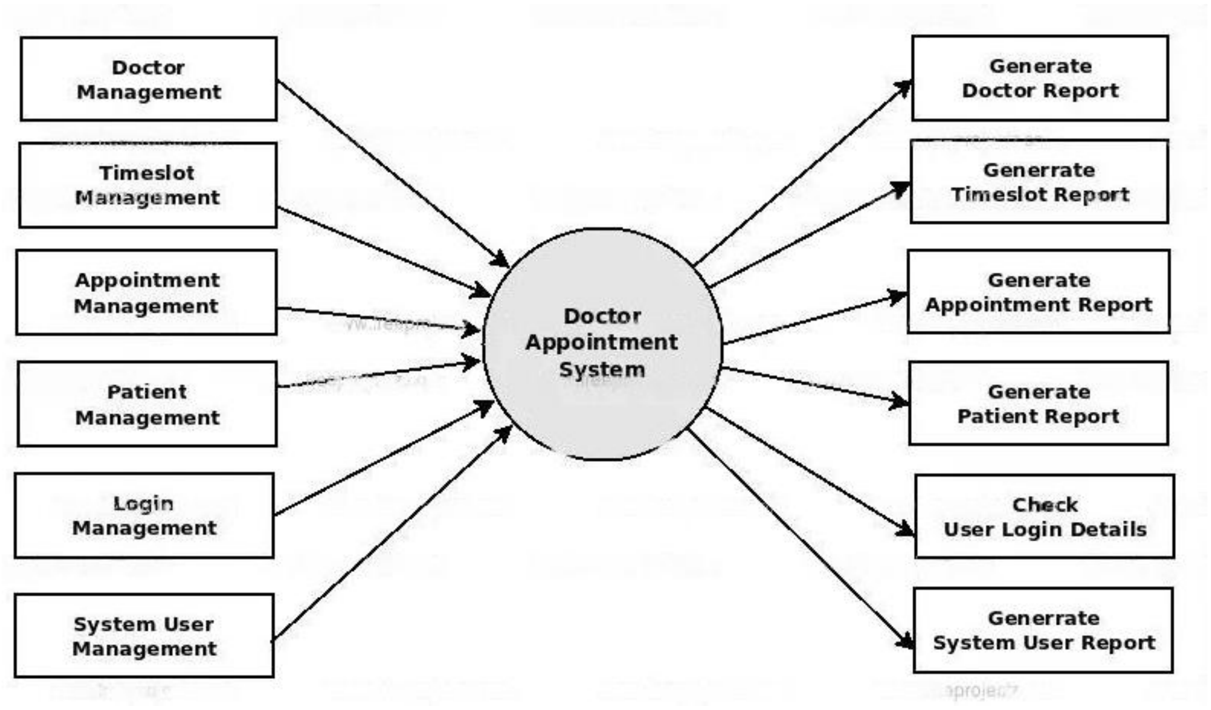


Figure 3.5 First Level DFD of Patient Appointment Booking

Second Level Data Flow Diagram(2nd Level DFD) Of Patient Appointment Booking:

DFD Level 2 then goes one step deeper into parts of Level 1 of Doctor Appointment. It may require more functionalities of Doctor Appointment to reach the necessary level of detail about the Doctor Appointment functioning First Level DFD (1st Level) of Patient Appointment Bookingshows how the system is divided into sub-systems (processes) The 2nd Level DFD contains more details of Medicine, Patient, Timeslot, Patient Booking. Appointment, Doctor

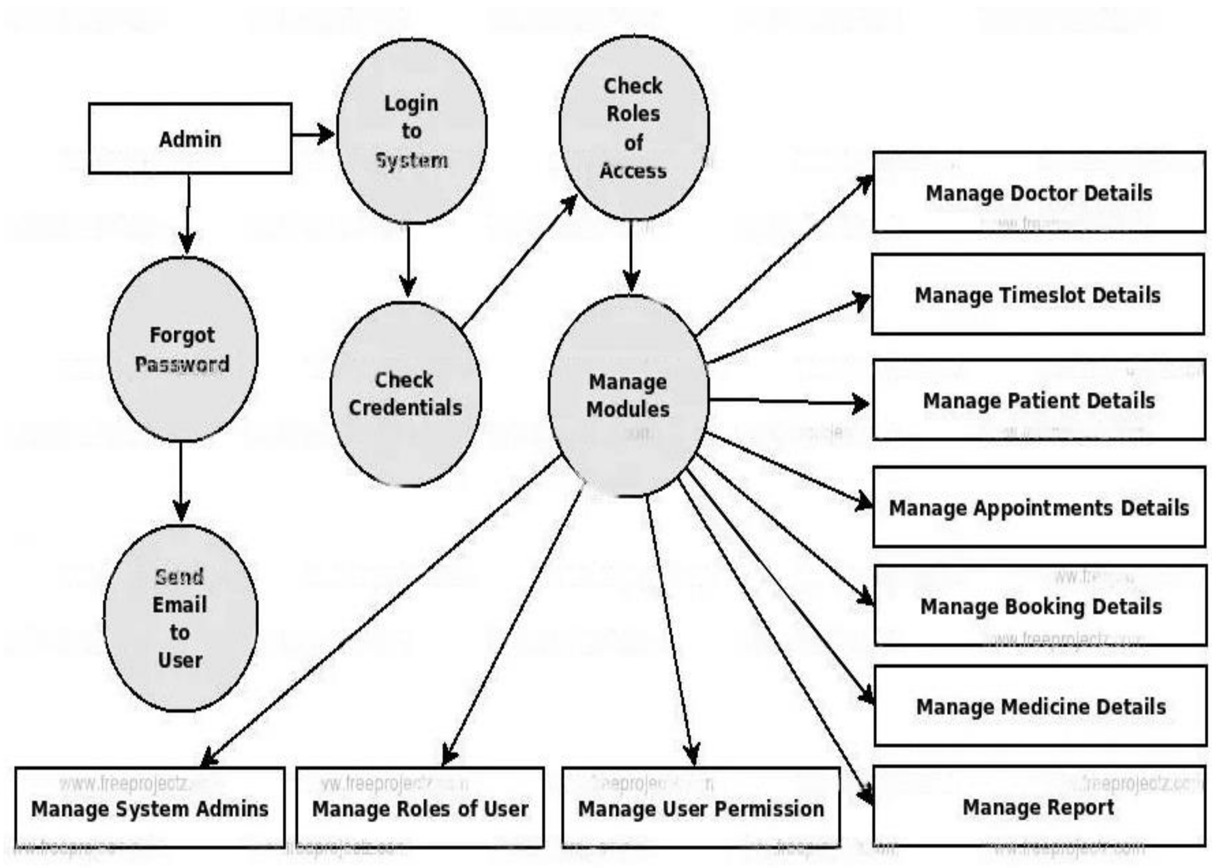


Figure 3.5 Second Level DFD of Patient Appointment Booking

3.2.6 SEQUENCE DIAGRAM

Sequence Diagram is defined as a dynamic model for a use case which is used for showing the interaction between classes for particular time period. This diagram include message, time.

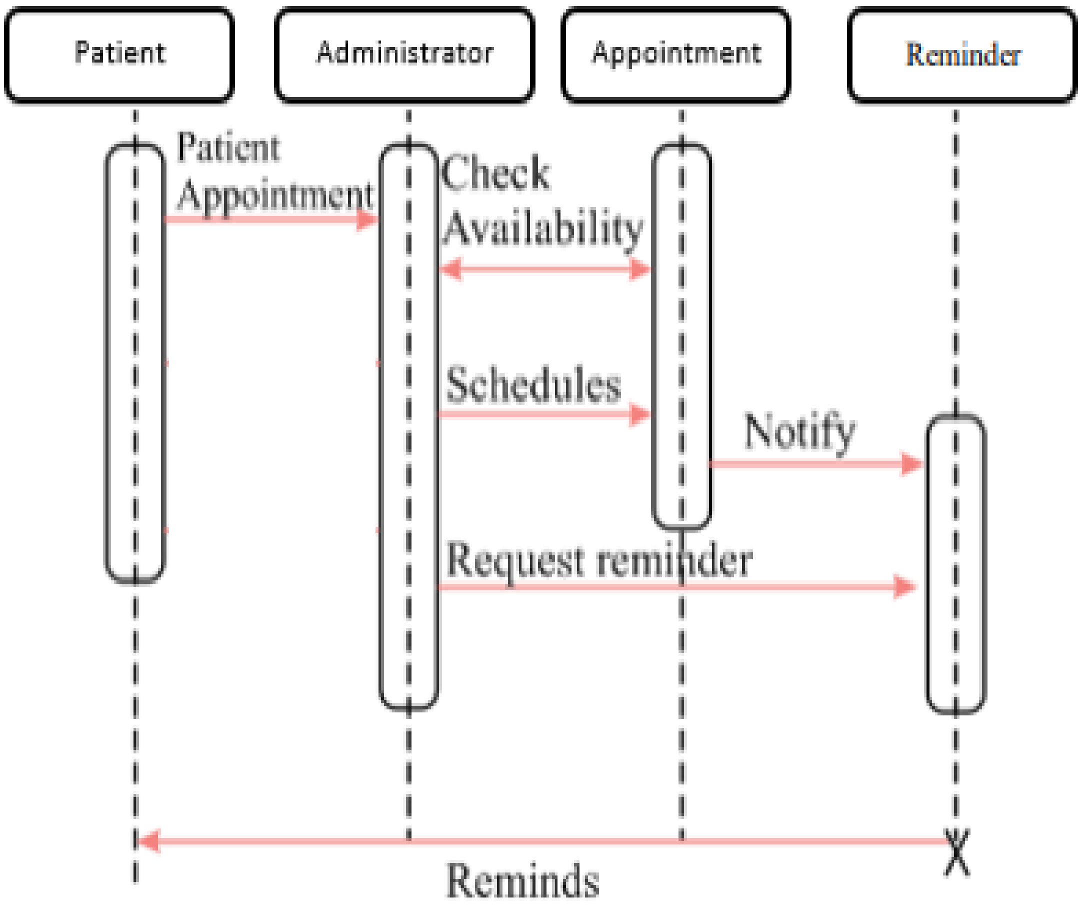


Figure 3.6 Sequence Diagram of Patient Appointment Booking

CHAPTER 4

FORM DESIGN

4.1 Login Page

A login form utilizes the credentials of a user, in order to authenticate their access. It generally consists of the typical username or email and password. But more fields can be added to improve the site's security

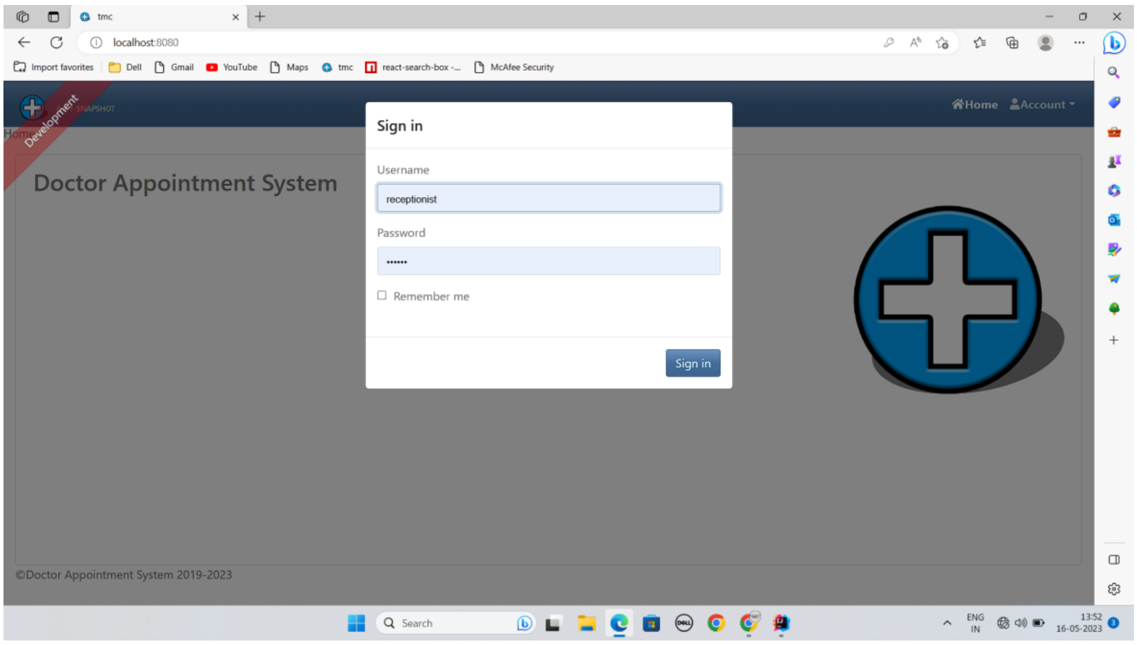
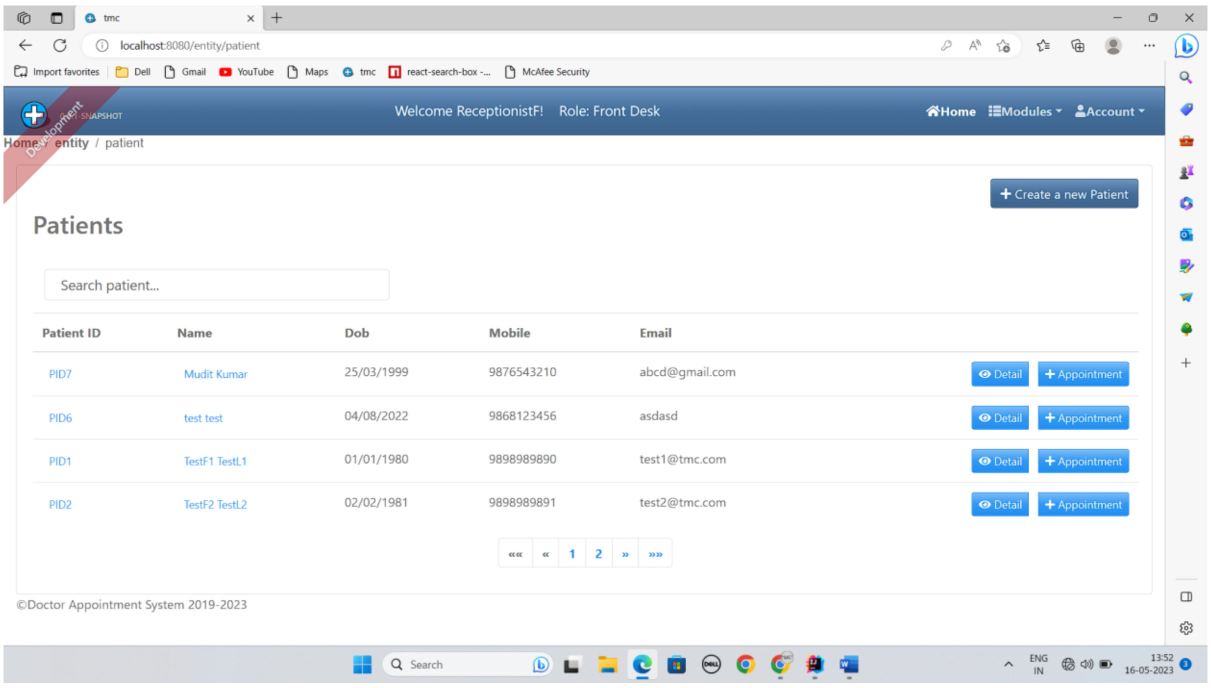


Figure 4.2Login Page

4.2 Patient Module

This module seamlessly takes care of admission, discharge and the transfer processes of patients. It enables the search on availability and manages the allocation of a bed, ward, and room to a patient according to the availability or cost associated and thereby manages the transfers.



Patients

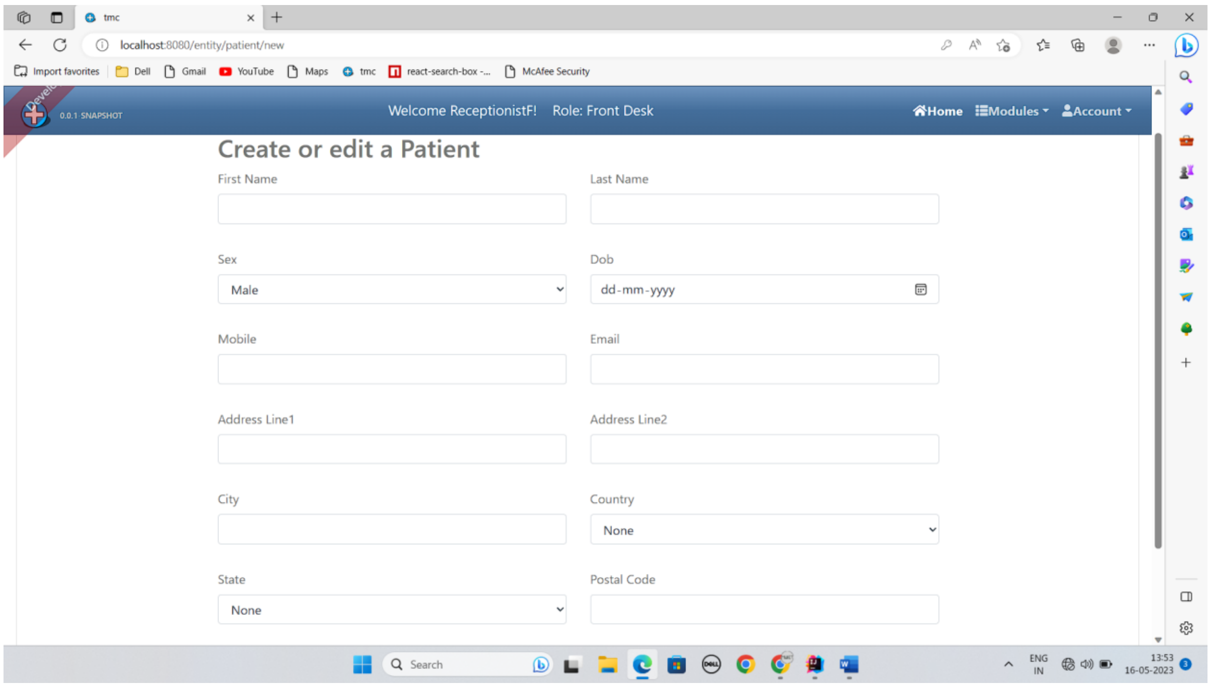
Patient ID	Name	Dob	Mobile	Email
PID7	Mudit Kumar	25/03/1999	9876543210	abcd@gmail.com
PID6	test test	04/08/2022	9868123456	asdasd
PID1	TestF1 TestL1	01/01/1980	9898989890	test1@tmc.com
PID2	TestF2 TestL2	02/02/1981	9898989891	test2@tmc.com

©Doctor Appointment System 2019-2023

Figure 4.3 Patient Page

4.3 Create or edit Patient.

This module is used to create or edit a patients details. We can create new patient and edit existing patients details.



localhost:8080/entity/patient/new
0.0.1 SNAPSHOT
Welcome ReceptionistF! Role: Front Desk
Home
Modules
Account

Create or edit a Patient

First Name

Last Name

Sex
Male

Dob
dd-mm-yyyy

Mobile

Email

Address Line1

Address Line2

City

Country
None

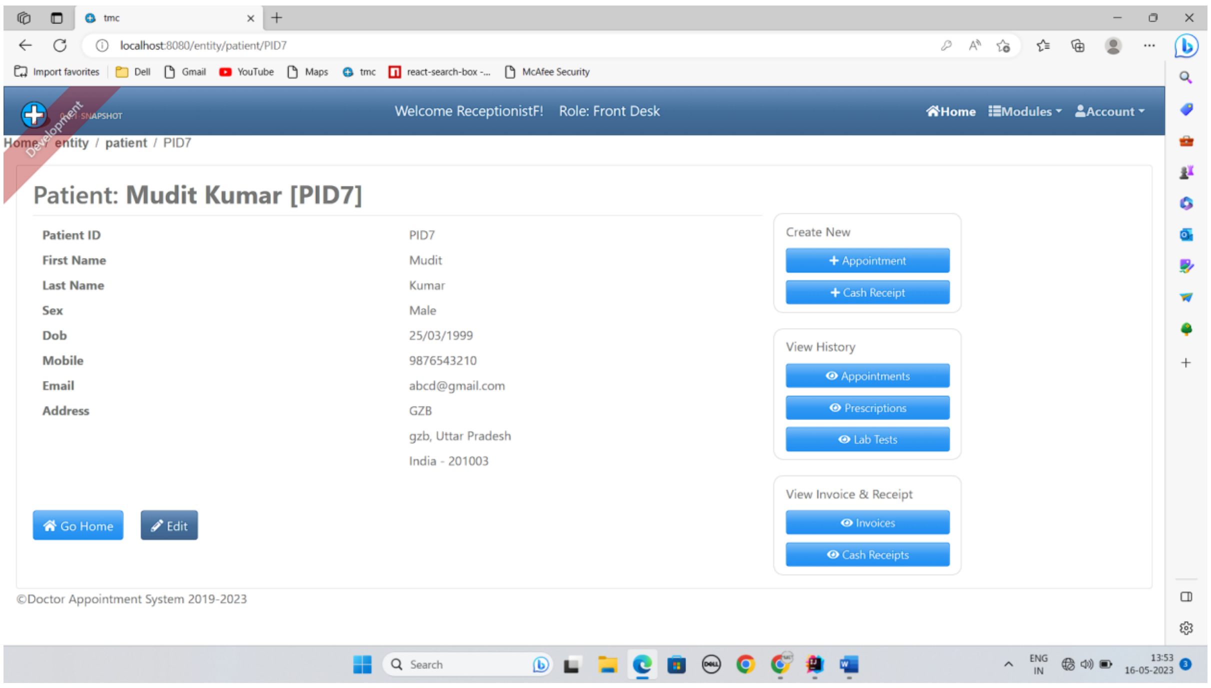
State
None

Postal Code

Figure 4.3 Create or Edit Patient Page

4.4 Patient Details

This Module is use to view patient details like name, dob, address, mob no. etc. we can view history of patient treatment and invoices.



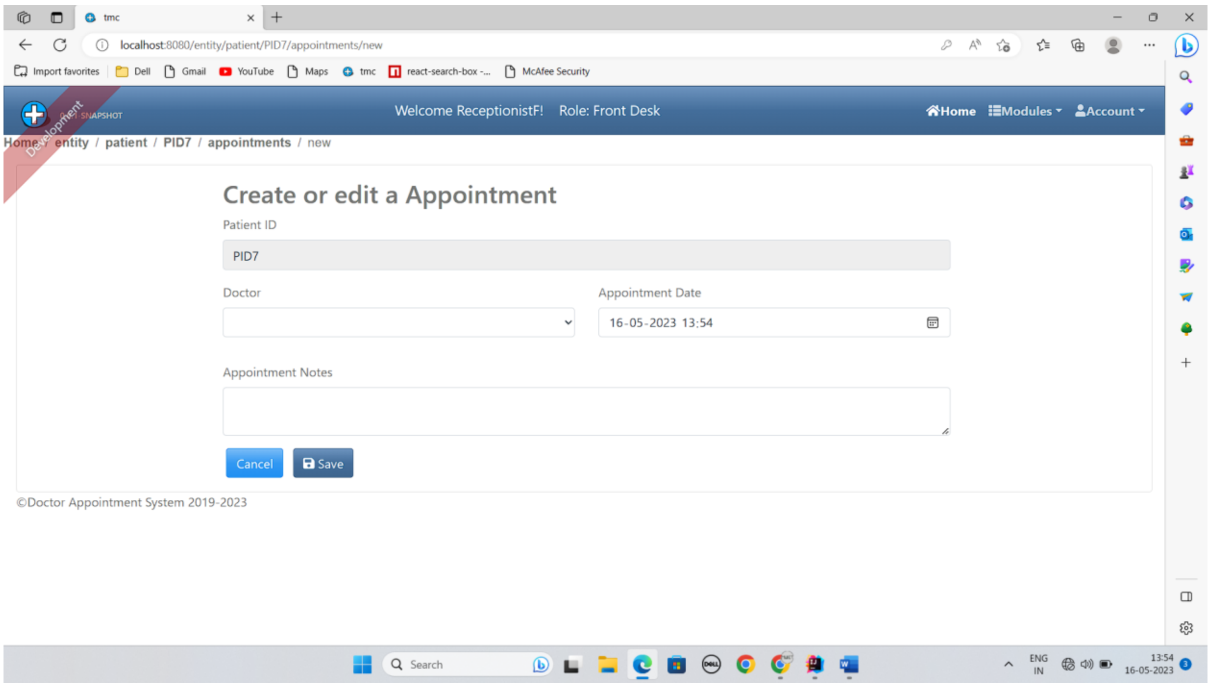
Patient: Mudit Kumar [PID7]

Patient ID	PID7
First Name	Mudit
Last Name	Kumar
Sex	Male
Dob	25/03/1999
Mobile	9876543210
Email	abcd@gmail.com
Address	GZB gzb, Uttar Pradesh India - 201003

Figure 4.4 Patient Details Page

4.5 Create or Edit Appointment

This module is used to create or edit a Appointment details. We can create new Appointment and edit existing Appointment details.



Welcome ReceptionistF! Role: Front Desk

Home / entity / patient / PID7 / appointments / new

Create or edit a Appointment

Patient ID

PID7

Doctor

Appointment Date

16-05-2023 13:54

Appointment Notes

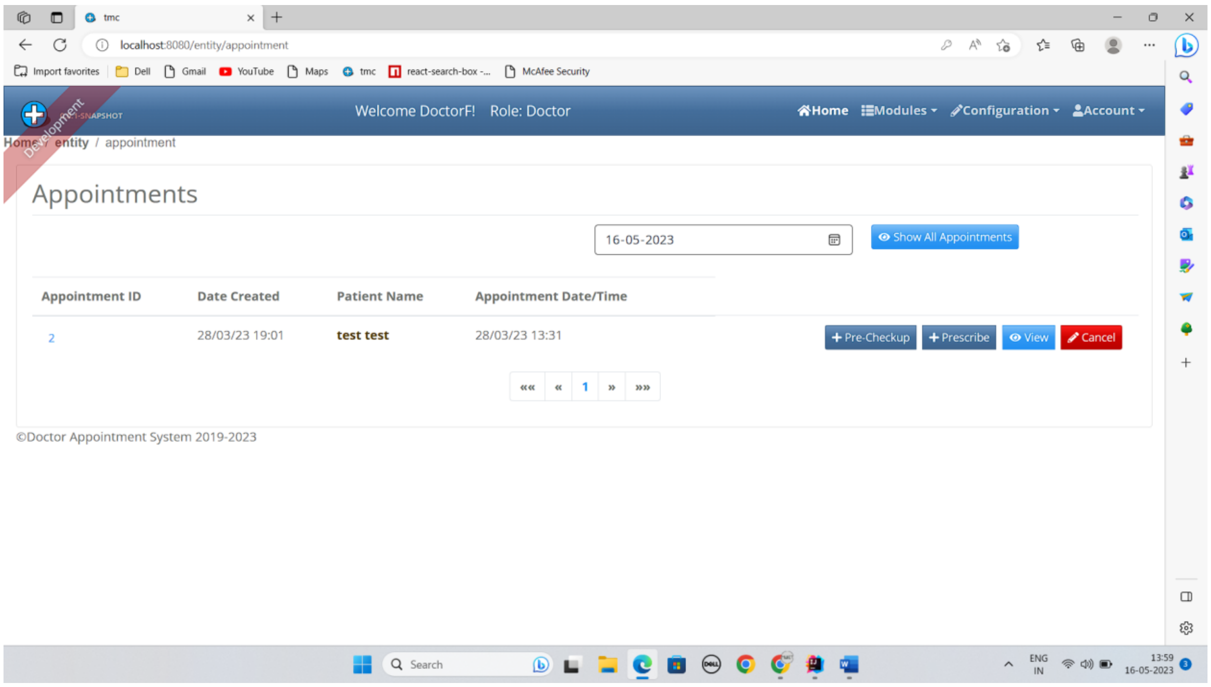
Cancel Save

©Doctor Appointment System 2019-2023

Figure 4.5 Create or Edit Appointment

4.6 Appointment Module

The Appointments module creates for each appointment scheduled reminders which sends events at a specified time period.



Appointments

16-05-2023

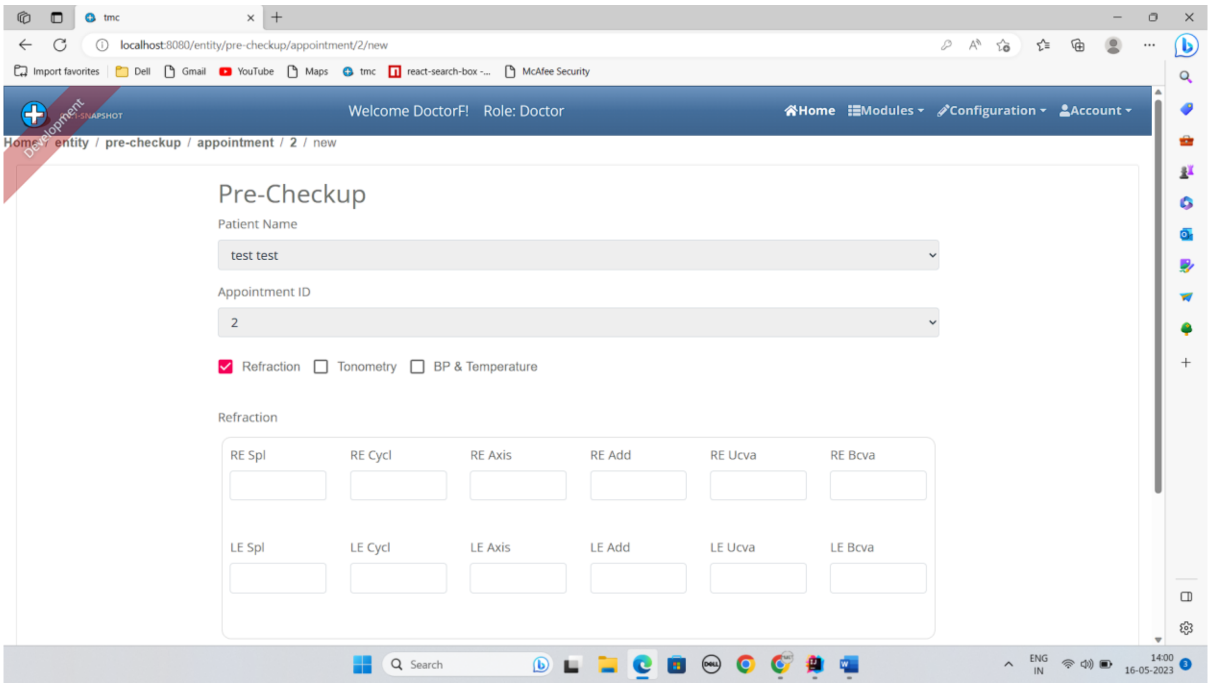
Appointment ID	Date Created	Patient Name	Appointment Date/Time
2	28/03/23 19:01	test test	28/03/23 13:31

©Doctor Appointment System 2019-2023

Figure 4.6 Appointment Page

4.7 Pre-Check Up

This Module allows staff to enter pre check up details prior to the Doctor consultation for the help of doctor to asses the problem.



Welcome DoctorF! Role: Doctor
Home
Modules
Configuration
Account
Home / entity / pre-checkup / appointment / 2 / new

Pre-Checkup

Patient Name: test test

Appointment ID: 2

[✓ Refraction] Tonometry BP & Temperature

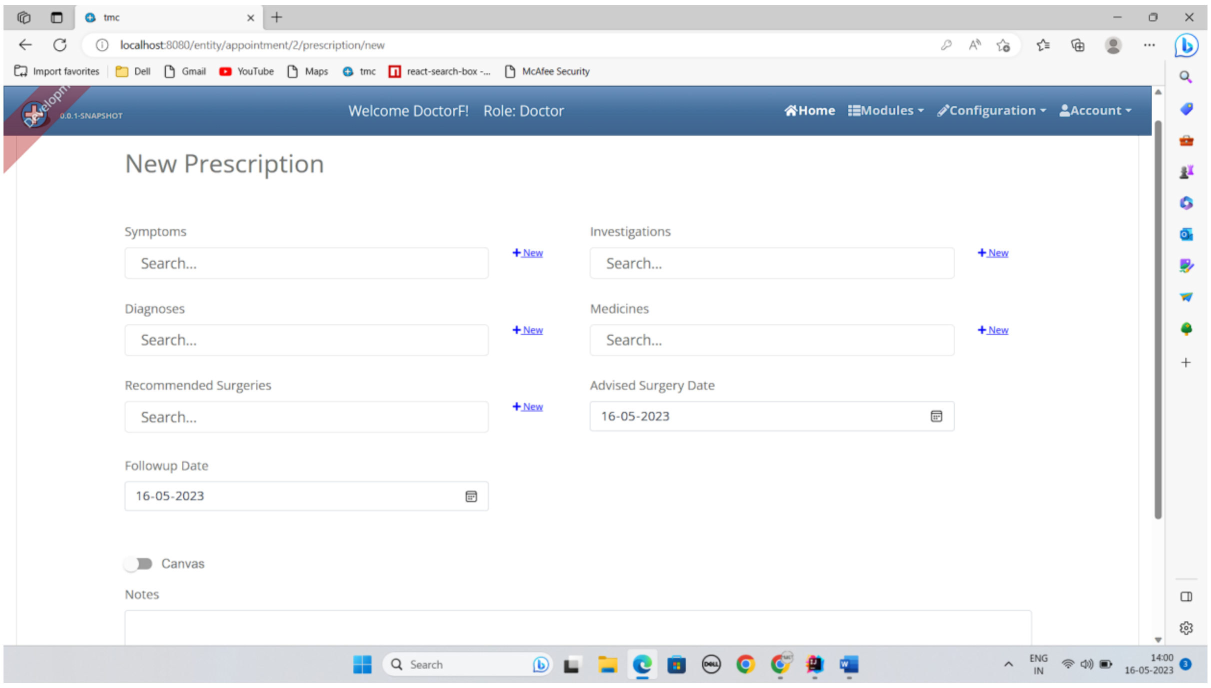
Refraction

RE Spl	RE Cycl	RE Axis	RE Add	RE Ucva	RE Bcva
LE Spl	LE Cycl	LE Axis	LE Add	LE Ucva	LE Bcva

Figure 4.7 Pre-Check Up Page

4.8 Create Prescription

This module is used to prescribe the medicine to patients on the basis of symptoms and investigation doctor can add new medicine and enter the diagnose details to get assisted by the software on the basis of symptoms.


localhost:8080/entity/appointment/2/prescription/new

Welcome DoctorF! Role: Doctor

New Prescription

Symptoms: Search...
Investigations: Search...
Diagnoses: Search...
Medicines: Search...
Recommended Surgeries: Search...
Advised Surgery Date: 16-05-2023
Followup Date: 16-05-2023
Canvas
Notes

Figure 4.8 New Prescription Page

CHAPTER 5

CODING

5.1 Login Module.

```
package com.tmc.domain;

import com.tmc.config.Constants;

import com.fasterxml.jackson.annotation.JsonIgnore;
import org.apache.commons.lang3.StringUtils;
import org.hibernate.annotations.BatchSize;
import org.hibernate.annotations.Cache;
import org.hibernate.annotations.CacheConcurrencyStrategy;

import javax.persistence.*;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;
import java.io.Serializable;
import java.time.Instant;
import java.util.HashSet;
import java.util.Locale;
import java.util.Set;

/**
 * A user.
```

```
            @NotNull
    @Pattern(regexp = Constants.LOGIN_REGEX)
    @Size(min = 1, max = 50)
    @Column(length = 50, unique = true, nullable = false)
    private String login;

    @JsonIgnore
    @NotNull
    @Size(min = 60, max = 60)
    @Column(name = "password_hash", length = 60, nullable = false)
    private String password;

    @Size(max = 50)
    @Column(name = "first_name", length = 50)
    private String firstName;

    @Size(max = 50)
    @Column(name = "last_name", length = 50)
    private String lastName;

    @Email
    @Size(min = 5, max = 254)
    @Column(length = 254, unique = true)
    private String email;

    @NotNull
    @Column(nullable = false)
    private boolean activated = false;

    @Size(min = 2, max = 6)
    @Column(name = "lang_key", length = 6)
    private String langKey;
    @Size(max = 256)
    @Column(name = "image_url", length = 256)
    private String imageUrl;

    @Size(max = 20)
    @Column(name = "activation_key", length = 20)
    @JsonIgnore
    private String activationKey;

    @Size(max = 20)
    @Column(name = "reset_key", length = 20)
    @JsonIgnore
    private String resetKey;

    @Column(name = "reset_date")
    private Instant resetDate = null;

    @JsonIgnore
    @ManyToMany
    @JoinTable(
        name = "jhi_user_authority",
        joinColumns = {@JoinColumn(name = "user_id", referencedColumnName =
"id")},
        inverseJoinColumns = {@JoinColumn(name = "authority_name",
referencedColumnName = "name")})
    @Cache(usage = CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
    @BatchSize(size = 20)
    private Set<Authority> authorities = new HashSet<>();

    @JsonIgnore
    @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true, mappedBy =
"user")
    @Cache(usage = CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
    private Set<PersistentToken> persistentTokens = new HashSet<>();
```

```
public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getLogin() {
    return login;
}

// Lowercase the login before saving it in database
public void setLogin(String login) {
    this.login = StringUtils.lowerCase(login, Locale.ENGLISH);
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}
```

```
public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getImageUrl() {
    return imageUrl;
}

public void setImageUrl(String imageUrl) {
    this.imageUrl = imageUrl;
}

public boolean getActivated() {
    return activated;
}

public void setActivated(boolean activated) {
    this.activated = activated;
}

public String getActivationKey() {
    return activationKey;
}

public void setActivationKey(String activationKey) {
    this.activationKey = activationKey;
}

public String getResetKey() {
    return resetKey;
}

public void setResetKey(String resetKey) {
    this.resetKey = resetKey;
}

public Instant getResetDate() {
    return resetDate;
}

public void setResetDate(Instant resetDate) {
    this.resetDate = resetDate;
}

public String getLangKey() {
    return langKey;
}

public void setLangKey(String langKey) {
    this.langKey = langKey;
}

public Set<Authority> getAuthorities() {
    return authorities;
}
```

```
public void setAuthorities(Set<Authority> authorities) {
    this.authorities = authorities;
}

public Set<PersistentToken> getPersistentTokens() {
    return persistentTokens;
}

public void setPersistentTokens(Set<PersistentToken> persistentTokens) {
    this.persistentTokens = persistentTokens;
}

@Override
public boolean equals(Object o) {
    if (this == o) {
        return true;
    }
    if (!(o instanceof User)) {
        return false;
    }
    return id != null && id.equals(((User) o).id);
}

@Override
public int hashCode() {
    return 31;
}

@Override
public String toString() {
    return "User{" +
        "login='" + login + '\'' +
        ", firstName='" + firstName + '\'' +
        ", lastName='" + lastName + '\'' +
            ", email='" + email + '\'' +
            ", imageUrl='" + imageUrl + '\'' +
            ", activated='" + activated + '\'' +
            ", langKey='" + langKey + '\'' +
            ", activationKey='" + activationKey + '\'' +
            "}";
    }
}
```

5.2 User Module

```
import './home.scss';

import React from 'react';
import { connect } from 'react-redux';
import { getSession, login } from 'app/shared/reducers/authentication';
import { Translate } from 'react-jhipster';
import { Col, Row } from 'reactstrap';
import LoginModal from 'app/modules/login/login-modal';
import { Redirect } from 'react-router-dom';
import {
  hasAnyAuthority,
  hasAdminRole,
  hasDoctorRole,
  hasFrontDeskRole,
  hasLabStaffRole,
  hasOTStaffRole
} from 'app/shared/auth/private-route';
import { AUTHORITIES } from 'app/config/constants';

export interface IHomeProp extends StateProps, DispatchProps {
  loginError: boolean;
  handleLogin: Function;
  isAdmin: boolean;
}

export class Home extends React.Component<IHomeProp> {
  showModalOnLogin = true;

  componentDidMount() {
    this.props.getSession();
  }

  handleLogin = (username, password, rememberMe = false) => {
    this.props.login(username, password, rememberMe);
  };

  handleClose = () => {
    // this.setState({ showModal: false });
```

```
};

handleRedirect = authorities => {
  if (hasAdminRole(authorities)) {
    return '/admin/user-management';
  } else if (hasDoctorRole(authorities)) {
    return '/entity/appointment';
  } else if (hasFrontDeskRole(authorities)) {
    return '/entity/patient';
  } else if (hasLabStaffRole(authorities)) {
    return '/entity/lab-test';
  } else if (hasOTStaffRole(authorities)) {
    return '/entity/surgical-procedure';
  } else {
    return '/entity/patient';
  }
};

render() {
  const { account, isAuthenticated, loginError, authorities } = this.props;

  return (
    <Row>
      <Col md="9">
```

```
        <h2>
          <Translate contentKey="home.title">Welcome to Patient Appointment
Booking!</Translate>
        </h2>
        <p className="lead">
          <Translate contentKey="home.subtitle">Please start with Login</Translate>
        </p>
        {account && account.login ? (
          <>
            <Redirect to={{ pathname: this.handleRedirect(authorities) }} />
          </>
        ) : (
          <div>
            <LoginModal
              showModal={this.showModalOnLogin}
              handleLogin={this.handleLogin}
              handleClose={this.handleClose}
              loginError={this.props.loginError}
            />
          </div>
        )}
      </Col>
      <Col md="3" className="pad">
        <span className="hipster rounded" />
      </Col>
      </Row>
    );
  }
}

const mapStateToProps = storeState => ({
  account: storeState.authentication.account,
  isAuthenticated: storeState.authentication.isAuthenticated,
  authorities: storeState.authentication.account.authorities,
  loginError: storeState.authentication.loginError
});

const mapDispatchToProps = { login, getSession };

type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof mapDispatchToProps;

export default connect(
  mapStateToProps,
  mapDispatchToProps
)(Home);
```

5.3 Patient Module

```
import React from 'react';
import { connect } from 'react-redux';
import { Link, RouteComponentProps } from 'react-router-dom';
import { Button, Col, Row, Table } from 'reactstrap';
import ReactSearchBox from 'react-search-box';
// tslint:disable-next-line:no-unused-variable
import {
  Translate,
  ICrudGetAllAction,
  TextFormat,
  JhiPagination,
  getPaginationItemsNumber,
  getSortState, IPaginationBaseState
} from 'react-jhipster';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';


import { IRootState } from 'app/shared/reducers';
import { ACTION_TYPES, getEntities } from './patient.reducer';
// tslint:disable-next-line:no-unused-variable
import { APP_DATE_FORMAT, APP_LOCAL_DATE_FORMAT } from
'app/config/constants';
import { escHackForSearchBar, fuseConfig, searchCallback,
showReactSearchDropDown } from 'app/shared/util/search-utils';
import { hasFrontDeskRole } from 'app/shared/auth/private-route';
import {ITEMS_PER_PAGE} from "app/shared/util/pagination.constants";
import {ISurgeryTypeState} from "app/entities/surgery-type/surgery-type";
```

```
export interface IPatientProps extends StateProps, DispatchProps,
RouteComponentProps<{ url: string }> {}

export type IPatientState = IPaginationBaseState;

export class Patient extends React.Component<IPatientProps> {

  state: IPatientState = {

    ...getSortState(this.props.location, ITEMS_PER_PAGE),

    itemsPerPage:4

  };




  componentDidMount() {

    this.getEntities();

    document.addEventListener('keydown', escHackForSearchBar, false);

  }


  componentDidUpdate(prevProps, prevState) {

    this.props.patientList.map(

      (patient, i) => (patient.value = '[' + patient.id + '] ' + patient.firstName + ' ' +
patient.lastName + ' ' + patient.mobile)

    );

  }


  componentWillUnmount() {

    document.removeEventListener('keydown', escHackForSearchBar, false);
  }

  onSearchSelect = x => {
    this.props.history.push('/entity/patient/' + x.id);
  };

  sort = prop => () => {
    this.setState(
      {
        order: this.state.order === 'asc' ? 'desc' : 'asc',
        sort: prop
      },
      () => this.sortEntities()
    );
  };

  sortEntities() {
    this.getEntities();

this.props.history.push(`${this.props.location.pathname}?page=${this.state.activePage}
&sort=${this.state.sort},${this.state.order}`);
  }

  handlePagination = activePage => this.setState({ activePage }, () =>
this.sortEntities());
```

```
 getEntities = () => {
   const { activePage, itemsPerPage, sort, order } = this.state;
   this.props.getEntities(activePage - 1, itemsPerPage, `${sort},${order}`);
 };



 render() {
   const { patientList, match, loading, authorities,totalItems } = this.props;

   const isFrontDesk = hasFrontDeskRole(authorities);

   return (
     <div>
       <h2 id="patient-heading">
         {isFrontDesk ? (
           <>
             <Link to={`${match.url}/new`} className="btn btn-primary float-right jh-
create-entity" id="jh-create-entity">
               <FontAwesomeIcon icon="plus" />
               &nbsp;
               <Translate contentKey="tmcApp.patient.home.createLabel">Create new
Patient</Translate>
             </Link>
             <br />
```

```
      </>
    ) : null}


    <Translate contentKey="tmcApp.patient.home.title">Patients</Translate>
    <br />
    <br />
    <div className="form-group col-md-4">
      <ReactSearchBox
        placeholder="Search patient..."
        value=""
        fuseConfigs={fuseConfig}
        data={patientList}
        onSelect={this.onSearchSelect}
        onChange={showReactSearchDropDown}
        callback={searchCallback}
        inputBoxFontSize={'18px'}
        autoFocus
      />
    </div>
  </h2>
  <div className="table-responsive">
    <Table responsive>
      <thead>
      <tr>
        <th>Patient ID</th>

        <th>Name</th>

        <th>

          <Translate contentKey="tmcApp.patient.dob">Dob</Translate>

        </th>

        <th>

          <Translate contentKey="tmcApp.patient.mobile">Mobile</Translate>

        </th>

        <th>

          <Translate contentKey="tmcApp.patient.email">Email</Translate>

        </th>

        <th />

      </tr>

      </thead>

      <tbody>

      {loading ? (

        <p>Loading...</p>

      ) : (

        patientList.map((patient, i) => (

          <tr key={`entity-${i}`}>

            <td>

              <Button tag={Link} to={`${match.url}/${patient.id}`} color="link"
    size="sm">

                {patient.id}

              </Button>
```

```
              </td>
              <td>
                <Button tag={Link} to={`${match.url}/${patient.id}`} color="link"
size="sm">
                  {patient.firstName} {patient.lastName}
                </Button>
              </td>
              <td>
                <TextFormat type="date" value={patient.dob}
format={APP_LOCAL_DATE_FORMAT} />
              </td>
              <td>{patient.mobile}</td>
              <td>{patient.email}</td>
              <td className="text-right">
                <div className="btn-group flex-btn-group-container">
                  <Button tag={Link} to={`${match.url}/${patient.id}`} color="info"
size="sm">
                    <FontAwesomeIcon icon="eye" />{' '}
                    <span className="d-none d-md-inline">
                        <Translate contentKey="entity.action.detail">Detail</Translate>
                      </span>
                  </Button>

                  {isFrontDesk ? (
                    <>
                      &nbsp;&nbsp;&nbsp;
                      <Button tag={Link}
to={`/entity/patient/${patient.id}/appointments/new`} color="info" size="sm">
                        <FontAwesomeIcon icon="plus" />
                        &nbsp;
                        <span className="d-none d-md-inline">Appointment</span>
                      </Button>
                    </>
                  ) : null}
                </div>
              </td>
            </tr>
          ))
        )}
        </tbody>
      </Table>
    </div>
    <Row className="justify-content-center">
      <JhiPagination
        items={getPaginationItemsNumber(totalItems, this.state.itemsPerPage)}
        activePage={this.state.activePage}
        onSelect={this.handlePagination}
        maxButtons={5}
      />
    </Row>
```

```
      </div>
    );
  }
}


const mapStateToProps = ({ patient, authentication }: IRootState) => ({
  patientList: patient.entities,
  totalItems: patient.totalItems,
  loading: patient.loading,
  authorities: authentication.account.authorities
});

const mapDispatchToProps = {
  getEntities
};

type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof mapDispatchToProps;

export default connect(
  mapStateToProps,
  mapDispatchToProps
)(Patient);
```

5.4 Test Module

```
import React from 'react';
import { connect } from 'react-redux';
import { Link, RouteComponentProps } from 'react-router-dom';
import { Button, Col, Row, Table } from 'reactstrap';
// tslint:disable-next-line:no-unused-variable
import { Translate, IPaginationBaseState, getSortState } from 'react-jhipster';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';

import { IRootState } from 'app/shared/reducers';
import { getEntities, getLabTestsByPatientId } from './lab-test.reducer';
// tslint:disable-next-line:no-unused-variable
import { ITEMS_PER_PAGE } from 'app/shared/util/pagination.constants';
import { hasLabStaffRole } from 'app/shared/auth/private-route';

export interface ILabTestProps extends StateProps, DispatchProps,
RouteComponentProps<{ url: string; patientId: string }> {}

export type ILabTestState = IPaginationBaseState;

export class LabTest extends React.Component<ILabTestProps, ILabTestState> {
  state: ILabTestState = {
    ...getSortState(this.props.location, ITEMS_PER_PAGE)
  };

  componentDidMount() {
    const { activePage, itemsPerPage, sort, order } = this.state;
    if (this.props.match.params.patientId) {
      this.props.getLabTestsByPatientId(this.props.match.params.patientId + '', activePage
- 1, itemsPerPage, null);
    } else {
      this.props.getEntities();
    }
  }

  render() {
    const { labTestList, match, loading } = this.props;

    const isLabStaff = hasLabStaffRole(this.props.authorities);

    return (
      <div>
        <h2 id="lab-test-heading">
          <Translate contentKey="tmcApp.labTest.home.title">Lab Tests</Translate>

          {/*{isLabStaff ? (
          <Link to={`${match.url}/new`} className="btn btn-primary float-right jh-
create-entity" id="jh-create-entity">
```

```
            <FontAwesomeIcon icon="plus" />
            &nbsp;
            <Translate contentKey="tmcApp.labTest.home.createLabel">Create new Lab
Test</Translate>
          </Link>
          ) : null}*/}
        </h2>
        <div className="table-responsive">
          <Table responsive>
            <thead>
              <tr>
                <th>Test ID</th>
                <th>Patient Name</th>
                <th>Test Name</th>
                <th>Prescription Id</th>
                <th />
              </tr>
            </thead>
            <tbody>
              {loading ? (
                <p>Loading...</p>
              ) : labTestList ? (
                labTestList.map((labTest, i) => (
                  <tr key={`entity-${i}`}>
                    <td>
                      <Button tag={Link} to={`${match.url}/${labTest.id}`} color="link"
size="sm">
                    {labTest.id}
                      </Button>
                    </td>
                    <td>{labTest.patient.firstName + ' ' + labTest.patient.lastName}</td>
                    <td>{labTest.reportType ? <Link to={`report-
type/${labTest.reportType.id}`}>{labTest.reportType.type}</Link> : ''}</td>
                    <td>
                      {labTest.prescription ? <Link
to={`prescription/${labTest.prescription.id}`}>{labTest.prescription.id}</Link> : ''}
                    </td>
                    <td className="text-right">
                      <div className="btn-group flex-btn-group-container">
                    <Button tag={Link} to={`${match.url}/${labTest.id}`} color="info"
size="sm">
                      <FontAwesomeIcon icon="eye" />{' '}
                      <span className="d-none d-md-inline">
                        <Translate contentKey="entity.action.view">View</Translate>
                      </span>
                    </Button>

                    {isLabStaff ? (
                      <>
                        &nbsp;&nbsp;&nbsp;
                    <Button tag={Link} to={`${match.url}/${labTest.id}/edit`}
color="primary" size="sm">
                      <FontAwesomeIcon icon="pencil-alt" /> <span className="d-none
d-md-inline">Update</span>
                    </Button>
                      </>
                    ) : null}
                  </div>
                </td>
              </tr>
            ))
          ) : null}
        </tbody>
      </Table>
      {this.props.match.params.patientId ? (
        <Button id="go-back" replace color="info"
onClick={this.props.history.goBack}>
          <FontAwesomeIcon icon="arrow-left" />
          &nbsp;
          <span className="d-none d-md-inline">
            <Translate contentKey="entity.action.back">Back</Translate>
          </span>
        </Button>
      ) : null}
</div>
```

```
      </div>
    );
  }
}

const mapStateToProps = storeState => ({
  labTestList: storeState.labTest.entities,
  authorities: storeState.authentication.account.authorities,
  loading: storeState.labTest.loading
});

const mapDispatchToProps = {
  getEntities,
  getLabTestsByPatientId
};

type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof mapDispatchToProps;

export default connect(
  mapStateToProps,
  mapDispatchToProps
)(LabTest);
```

5.5 Appointment Module

```
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';

// tslint:disable-next-line:no-unused-variable

import { APP_DATE_FORMAT } from 'app/config/constants';

import { hasDoctorRole, hasFrontDeskRole, hasLabStaffRole } from
'app/shared/auth/private-route';

import { currentDate } from 'app/shared/util/date-utils';

import { ITEMS_PER_PAGE } from 'app/shared/util/pagination.constants';

import { AvField, AvForm } from 'availity-reactstrap-validation';

import React from 'react';

// tslint:disable-next-line:no-unused-variable

import { getPaginationItemsNumber, getSortState, IPaginationBaseState, JhiPagination,
TextFormat, Translate } from 'react-jhipster';

import { connect } from 'react-redux';

import { Link, RouteComponentProps } from 'react-router-dom';

import { Button, Row, Table } from 'reactstrap';

import { getAllAppointmentsForDoctor, getAppointmentsForADate, getEntities,
getSelectedEntitiesForFrontDesk } from './appointment.reducer';


export interface IAppointmentProps extends StateProps, DispatchProps,
RouteComponentProps<{ url: string }> {}


export type IAppointmentState = IPaginationBaseState;
export class Appointment extends React.Component<IAppointmentProps,
IAppointmentState> {
  state: IAppointmentState = {
    ...getSortState(this.props.location, ITEMS_PER_PAGE)
  };

  componentDidMount() {
    this.getData();
  }

  sort = prop => () => {
    this.setState(
      {
        order: this.state.order === 'asc' ? 'desc' : 'asc',
        sort: prop
      },
      () => this.sortEntities()
    );
  };

  sortEntities() {
    this.getData();
this.props.history.push(`${this.props.location.pathname}?page=${this.state.activePage}
&sort=${this.state.sort},${this.state.order}`);
  }

  getData() {
    this.state.sort = 'allocatedDate';
    this.state.order = 'desc';

    const { activePage, itemsPerPage, sort, order } = this.state;

    if (this.props.match.url.includes('patient', 0)) {
      this.props.getSelectedEntitiesForFrontDesk(this.props.match.url.replace('/entity',
'api'));
    } else {
      this.props.getEntities(activePage - 1, itemsPerPage, `${sort},${order}`);
    }
  }

  showAllAppointmentForDoctor = () => {
    const { activePage, itemsPerPage, sort, order } = this.state;
    this.props.getAllAppointmentsForDoctor('', activePage - 1, itemsPerPage,
`${sort},${order}`);
  };

  appointmentDataByDate = ({ target }) => {
    this.state.sort = 'allocatedDate';
    this.state.order = 'desc';

    const { activePage, itemsPerPage, sort, order } = this.state;

    if (this.props.match.url.includes('patient', 0)) {
      this.props.getSelectedEntitiesForFrontDesk(this.props.match.url.replace('/entity',
'api') + '?appointmentDate=' + target.value);
    } else {
      this.props.getAppointmentsForADate('appointmentDate=' + target.value, activePage
- 1, itemsPerPage, `${sort},${order}`);
    }
  };

  handlePagination = activePage => this.setState({ activePage }, () =>
this.sortEntities());

  render() {
    const { appointmentList, match, totalItems, authorities, loading } = this.props;

    const isFrontDesk = hasFrontDeskRole(authorities);
    const isDoctor = hasDoctorRole(authorities);
    const isLabStaff = hasLabStaffRole(authorities);
    const isFwd = this.props.match.url.includes('patient', 0);
  return (
    <div>
      <h2 id="appointment-heading">
        <Translate
contentKey="tmcApp.appointment.home.title">Appointments</Translate>
      </h2>

      {isDoctor ? (
        <AvForm>
          <Table>
            <tr>
              <td className="w-50" />
              <td className="w-auto">
                <AvField
                  id="appointment-date"
                  type="date"
                  name="appointment-date"
                  className="form-control float-right border-dark"
                  value={currentDate()}
                  onChange={this.appointmentDataByDate}
                />
              </td>
              <td className="w-25">
                <Button color="info" size="sm" class="float-right"
onClick={this.showAllAppointmentForDoctor}>
```

```
                  <FontAwesomeIcon icon="eye" /> <span className="d-none d-md-
inline">Show All Appointments</span>
                </Button>
              </td>
            </tr>
          </Table>
        </AvForm>
      ) : null}

      <div className="table-responsive">
        <Table responsive>
          <thead>
            <tr>
              <th className="hand" onClick={this.sort('id')}>
                Appointment ID
              </th>

              <th>Date Created</th>

              <th className="hand">Patient Name</th>

              {isFrontDesk ? (
                <th>
                  <Translate contentKey="tmcApp.appointment.doctor">Doctor</Translate>
                </th>
              ) : null}


              <th className="hand" onClick={this.sort('allocatedDate')}>
                Appointment Date/Time
              </th>
              <th />
            </tr>
          </thead>
          <tbody>
            {loading ? (
              <p>Loading...</p>
            ) : (
              <>
                {appointmentList &&
                  appointmentList.map((appointment, i) => (
                    <tr key={`entity-${i}`}>
                      <td>
                        <Button tag={Link} to={`${match.url}/${appointment.id}`}
color="link" size="sm">
                          {appointment.id}
                        </Button>
                      </td>
                      <td>
                        <TextFormat type="date" value={appointment.dateCreated}
format={APP_DATE_FORMAT} />
```

```
</td>
<td>
  {appointment.patient ? (
    <Link to={`/entity/patient/${appointment.patient.id}`}>
      {appointment.patient.firstName + ' ' + appointment.patient.lastName}
    </Link>
  ) : null}
</td>
{isFrontDesk ? (
  <td>
    {appointment.doctor ? (
      <Link to={`/entity/app-user/${appointment.doctor.id}`}>
        {appointment.doctor.user.firstName + ' ' +
appointment.doctor.user.lastName}
      </Link>
    ) : (
      ''
    )}
  </td>
) : null}
<td>
  <TextFormat type="date" value={appointment.allocatedDate}
format={APP_DATE_FORMAT} />
</td>
&nbsp;&nbsp;
```

```
                  {!isFwd ? (
                   <td className="text-right">
                    <div className="btn-group flex-btn-group-container">
                     {!isFrontDesk && !appointment.isClosed &&
!appointment.prescriptionId ? (
                      <>
                       {appointment.preCheckup ? (
                        <Button
                         tag={Link}
                         to={`/entity/pre-checkup/${appointment.preCheckup.id}/edit`}
                         color="primary"
                         size="sm"
                        >
                         <FontAwesomeIcon icon="pencil-alt" /> <span className="d-
none d-md-inline">Pre-Checkup</span>
                        </Button>
                       ) : (
                        <Button
                         tag={Link}
                         to={`/entity/pre-checkup/appointment/${appointment.id}/new`}
                         color="primary"
                         size="sm"
                        >
                         <FontAwesomeIcon icon="plus" /> <span className="d-none
d-md-inline">Pre-Checkup</span>
```

```
                              </Button>
                            )}
                            &nbsp;&nbsp;
                          </>
                        ) : null}
                        {isDoctor && !appointment.isClosed &&
!appointment.prescriptionId ? (
                          <>
                            <Button tag={Link}
to={`${match.url}/${appointment.id}/prescription/new`} color="primary" size="sm">
                              <FontAwesomeIcon icon="plus" /> <span className="d-none d-
md-inline">Prescribe</span>
                            </Button>
                            &nbsp;&nbsp;
                          </>
                        ) : null}
                        <Button tag={Link} to={`${match.url}/${appointment.id}`}
color="info" size="sm">
                          <FontAwesomeIcon icon="eye" />{' '}
                          <span className="d-none d-md-inline">
                            <Translate contentKey="entity.action.view">View</Translate>
                          </span>
                        </Button>
                        &nbsp;&nbsp;
                        {!appointment.isClosed && !appointment.prescriptionId &&
!isLabStaff ? (
                          <>
                            <Button tag={Link}
to={`${match.url}/${appointment.id}/cancel`} color="danger" size="sm">
                              <FontAwesomeIcon icon="pencil-alt" />{' '}
                              <span className="d-none d-md-inline">
                                <Translate
contentKey="entity.action.cancel">Cancel</Translate>
                              </span>
                            </Button>
                            &nbsp;&nbsp;
                          </>
                        ) : null}
                        {isFrontDesk ? (
                          <Button tag={Link}
to={`/entity/invoice/appointment/${appointment.id}/new`} color="primary"
size="sm">
                            <FontAwesomeIcon icon="plus" /> <span className="d-none d-
md-inline">Create Invoice</span>
                          </Button>
                        ) : null}
                      </div>
                    </td>
                  ) : isFrontDesk ? (
                    <td className="text-right">
                      <div className="btn-group flex-btn-group-container">
```

```
                              <Button tag={Link}
to={`/entity/invoice/appointment/${appointment.id}/new`} color="primary"
size="sm">
                                <FontAwesomeIcon icon="plus" /> <span className="d-none d-
md-inline">Create Invoice</span>
                              </Button>
                            </div>
                          </td>
                        ) : null}
                      </tr>
                    ))}
                  </>
                )}
              </tbody>
            </Table>
            {isFwd ? (
              <Button id="go-back" replace color="info"
onClick={this.props.history.goBack}>
                <FontAwesomeIcon icon="arrow-left" />
                &nbsp;
                <span className="d-none d-md-inline">
                  <Translate contentKey="entity.action.back">Back</Translate>
                </span>
              </Button>
            ) : null}
          </div>
          <Row className="justify-content-center">
            <JhiPagination
              items={getPaginationItemsNumber(totalItems, this.state.itemsPerPage)}
              activePage={this.state.activePage}
              onSelect={this.handlePagination}
              maxButtons={5}
            />
          </Row>
        </div>
      );
    }
  }

const mapStateToProps = storeState => ({
  appointmentList: storeState.appointment.entities,
  totalItems: storeState.appointment.totalItems,
  authorities: storeState.authentication.account.authorities,
  account: storeState.authentication.account,
  loading: storeState.appointment.loading
});

const mapDispatchToProps = {
  getEntities,
  getAllAppointmentsForDoctor,
  getAppointmentsForADate,
  getSelectedEntitiesForFrontDesk
};

type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof mapDispatchToProps;

export default connect(
  mapStateToProps,
  mapDispatchToProps
)(Appointment);
```

5.6 Doctor Module

```
import { faDownload } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
// tslint:disable-next-line:no-unused-variable
import { APP_LOCAL_DATE_FORMAT } from 'app/config/constants';
import { hasDoctorRole, hasFrontDeskRole } from 'app/shared/auth/private-route';
import React from 'react';
// tslint:disable-next-line:no-unused-variable
import { TextFormat, Translate } from 'react-jhipster';
import { connect } from 'react-redux';
import { Link, RouteComponentProps } from 'react-router-dom';
import { Button, Table } from 'reactstrap';
import { getEntities, getSelectedEntities } from './prescription.reducer';

export interface IPrescriptionProps extends StateProps, DispatchProps,
RouteComponentProps<{ url: string }> {}

export class Prescription extends React.Component<IPrescriptionProps> {
  prescriptionsUrl = '/entity/prescription';

  componentDidMount() {
    if (this.props.match.url.includes('patient', 0)) {
      this.props.getSelectedEntities(this.props.match.url.replace('/entity', 'api'));
    } else {
      this.props.getEntities();
    }
  }

  render() {
    const { prescriptions, match, authorities, loading } = this.props;
    const isDoctor = hasDoctorRole(authorities);
    const isFrontDesk = hasFrontDeskRole(authorities);
    const isFwd = match.url.includes('patient', 0);
    return (
      <div>
        <h2 id="prescription-heading">
```

```
          <Translate
contentKey="tmcApp.prescription.home.title">Prescriptions</Translate>
        </h2>
        <div className="table-responsive">
          <Table responsive>
            <thead>
              <tr>
                <th>Prescription ID</th>
                <th>Appointment ID</th>
                <th>Patient Name</th>
                <th>Doctor</th>
                <th>Date</th>
                <th />
              </tr>
            </thead>
            <tbody>
              {loading ? (
                <p>Loading...</p>
              ) : (
                <>
                  {prescriptions.map((prescription, i) => (
                    <tr key={`entity-${i}`}>
                      <td>
                        <Button tag={Link} to={`${this.prescriptionsUrl}/${prescription.id}`}
color="link" size="sm">
```

```
                      {prescription.id}
                    </Button>
                  </td>
                  <td>{prescription.appointment.id}</td>
                  <td>{prescription.patient.firstName + ' ' +
prescription.patient.lastName}</td>
                  <td>{prescription.appointment.doctor.user.firstName + ' ' +
prescription.appointment.doctor.user.lastName}</td>
                  <td>
                    <TextFormat type="date" value={prescription.date}
format={APP_LOCAL_DATE_FORMAT} />
                  </td>
                  <td className="text-right">
                    <div className="btn-group flex-btn-group-container">
                      {!isFrontDesk && !prescription.appointment.isClosed &&
prescription.appointment.preCheckup ? (
                        <>
                          <Button
                            tag={Link}
                            to={`/entity/pre-
checkup/${prescription.appointment.preCheckup.id}/edit`}
                            color="primary"
                            size="sm"
                          >
                            <FontAwesomeIcon icon="pencil-alt" /> <span className="d-none
d-md-inline">Pre-Checkup</span>
```

```
                          </Button>
                          &nbsp;&nbsp;
                        </>
                      ) : null}
                      <Button tag={Link} to={`${this.prescriptionsUrl}/${prescription.id}`}
color="info" size="sm">
                        <FontAwesomeIcon icon="eye" />{' '}
                        <span className="d-none d-md-inline">
                          <Translate contentKey="entity.action.view">View</Translate>
                        </span>
                      </Button>
                      &nbsp;&nbsp;
                      {isDoctor ? (
                        <Button tag={Link}
to={`${this.prescriptionsUrl}/${prescription.id}/edit`} color="primary" size="sm">
                          <FontAwesomeIcon icon="pencil-alt" />{' '}
                          <span className="d-none d-md-inline">
                            <Translate contentKey="entity.action.edit">Edit</Translate>
                          </span>
                        </Button>
                      ) : isFrontDesk ? (
                        <>
                          <Button tag={Link}
to={`/entity/invoice/prescription/${prescription.id}/new`} color="primary" size="sm">
                            <FontAwesomeIcon icon="plus" /> <span className="d-none d-
md-inline">Create Invoice</span>
```

```
                    </Button>
                    &nbsp;&nbsp;
                    <Button tag={Link} to={`/entity/consent/${prescription.id}`}
color="primary" size="sm">
                      <FontAwesomeIcon icon={faDownload} /> <span className="d-
none d-md-inline">Consent Form</span>
                    </Button>
                  </>
                ) : null}
              </div>
            </td>
          </tr>
        ))}
      </>
    )}
  </tbody>
</Table>

{isFwd ? (
  <Button id="go-back" replace color="info"
onClick={this.props.history.goBack}>
    <FontAwesomeIcon icon="arrow-left" />
    &nbsp;
    <span className="d-none d-md-inline">
      <Translate contentKey="entity.action.back">Back</Translate>
```

```
              </span>
            </Button>
          ) : null}
        </div>
      </div>
    );
  }
}

const mapStateToProps = storeState => ({
  prescriptions: storeState.prescription.entities,
  authorities: storeState.authentication.account.authorities,
  loading: storeState.prescription.loading
});

const mapDispatchToProps = {
  getEntities,
  getSelectedEntities
};

type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof mapDispatchToProps;

export default connect(
  mapStateToProps,
  mapDispatchToProps
)(Prescription);
```

5.7 Medicine Module

```
import React from 'react';
import { connect } from 'react-redux';
import { Link, RouteComponentProps } from 'react-router-dom';
import { Button, Col, Row, Table } from 'reactstrap';
// tslint:disable-next-line:no-unused-variable
import { Translate, ICrudGetAllAction, getSortState, IPaginationBaseState,
getPaginationItemsNumber, JhiPagination } from 'react-jhipster';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';

import { IRootState } from 'app/shared/reducers';
import { getEntities } from './discharge-medicine.reducer';
import { IDischargeMedicine } from 'app/shared/model/discharge-medicine.model';
// tslint:disable-next-line:no-unused-variable
import { APP_DATE_FORMAT, APP_LOCAL_DATE_FORMAT } from
'app/config/constants';
import { ITEMS_PER_PAGE } from 'app/shared/util/pagination.constants';

export interface IDischargeMedicineProps extends StateProps, DispatchProps,
RouteComponentProps<{ url: string }> {}

export type IDischargeMedicineState = IPaginationBaseState;

export class DischargeMedicine extends React.Component<IDischargeMedicineProps,
IDischargeMedicineState> {
  state: IDischargeMedicineState = {
    ...getSortState(this.props.location, ITEMS_PER_PAGE)
  };

  componentDidMount() {
    this.getEntities();
  }

  sort = prop => () => {
    this.setState(
      {
        order: this.state.order === 'asc' ? 'desc' : 'asc',
        sort: prop
      },
      () => this.sortEntities()
    );
  };

  sortEntities() {
    this.getEntities();

this.props.history.push(`${this.props.location.pathname}?page=${this.state.activePage}
&sort=${this.state.sort},${this.state.order}`);
  }

  handlePagination = activePage => this.setState({ activePage }, () =>
this.sortEntities());

  getEntities = () => {
    const { activePage, itemsPerPage, sort, order } = this.state;
    this.props.getEntities(activePage - 1, itemsPerPage, `${sort},${order}`);
  };

  render() {
    const { dischargeMedicineList, match, totalItems } = this.props;
    return (
    <div>
      <h2 id="discharge-medicine-heading">
        <Translate contentKey="tmcApp.dischargeMedicine.home.title">Discharge
Medicines</Translate>
        <Link to={`${match.url}/new`} className="btn btn-primary float-right jh-
create-entity" id="jh-create-entity">
          <FontAwesomeIcon icon="plus" />
          &nbsp;
          <Translate contentKey="tmcApp.dischargeMedicine.home.createLabel">Create
new Discharge Medicine</Translate>
        </Link>
      </h2>
      <div className="table-responsive">
        <Table responsive>
          <thead>
            <tr>
              <th className="hand" onClick={this.sort('id')}>
                <Translate contentKey="global.field.id">ID</Translate>
<FontAwesomeIcon icon="sort" />
              </th>
              <th className="hand" onClick={this.sort('frequency')}>
                <Translate
contentKey="tmcApp.dischargeMedicine.frequency">Frequency</Translate>
<FontAwesomeIcon icon="sort" />
              </th>
              <th className="hand" onClick={this.sort('duration')}>
                <Translate
contentKey="tmcApp.dischargeMedicine.duration">Duration</Translate>
<FontAwesomeIcon icon="sort" />
              </th>
              <th>
                <Translate
contentKey="tmcApp.dischargeMedicine.medicine">Medicine</Translate>
<FontAwesomeIcon icon="sort" />
```

```
            </th>
            <th>
              <Translate
contentKey="tmcApp.dischargeMedicine.surgicalProcedure">Surgical
Procedure</Translate>{' '}
              <FontAwesomeIcon icon="sort" />
            </th>
            <th />
          </tr>
        </thead>
        <tbody>
          {dischargeMedicineList.map((dischargeMedicine, i) => (
            <tr key={`entity-${i}`}>
              <td>
                <Button tag={Link} to={`${match.url}/${dischargeMedicine.id}`}
color="link" size="sm">
                  {dischargeMedicine.id}
                </Button>
              </td>
              <td>{dischargeMedicine.doseFrequency}</td>
              <td>{dischargeMedicine.doseDuration}</td>
              <td>
                {dischargeMedicine.medicine ? (
                  <Link
to={`medicine/${dischargeMedicine.medicine.id}`}>{dischargeMedicine.medicine.nam
e}</Link>
                ) : (
                  ''
                )}
              </td>
              <td>
                {dischargeMedicine.surgicalProcedure ? (
                  <Link to={`surgical-
procedure/${dischargeMedicine.surgicalProcedure.id}`}>
```

```
                    {dischargeMedicine.surgicalProcedure.id}
                  </Link>
                ) : (
                  ''
                )}
              </td>
              <td className="text-right">
                <div className="btn-group flex-btn-group-container">
                  <Button tag={Link} to={`${match.url}/${dischargeMedicine.id}`}
color="info" size="sm">
                    <FontAwesomeIcon icon="eye" />{' '}
                    <span className="d-none d-md-inline">
                      <Translate contentKey="entity.action.view">View</Translate>
                    </span>
                  </Button>
                  <Button tag={Link} to={`${match.url}/${dischargeMedicine.id}/edit`}
color="primary" size="sm">
                    <FontAwesomeIcon icon="pencil-alt" />{' '}
                    <span className="d-none d-md-inline">
                      <Translate contentKey="entity.action.edit">Edit</Translate>
                    </span>
                  </Button>
                  <Button tag={Link} to={`${match.url}/${dischargeMedicine.id}/delete`}
color="danger" size="sm">
                    <FontAwesomeIcon icon="trash" />{' '}
                    <span className="d-none d-md-inline">
                      <Translate contentKey="entity.action.delete">Delete</Translate>
                    </span>
                  </Button>
                </div>
              </td>
            </tr>
          ))}
        </tbody>
```

```
          </Table>
        </div>
        <Row className="justify-content-center">
          <JhiPagination
            items={getPaginationItemsNumber(totalItems, this.state.itemsPerPage)}
            activePage={this.state.activePage}
            onSelect={this.handlePagination}
            maxButtons={5}
          />
        </Row>
      </div>
    );
  }
}

const mapStateToProps = ({ dischargeMedicine }: IRootState) => ({
  dischargeMedicineList: dischargeMedicine.entities,
  totalItems: dischargeMedicine.totalItems
});

const mapDispatchToProps = {
  getEntities
};

type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof mapDispatchToProps;

export default connect(
  mapStateToProps,
  mapDispatchToProps
)(DischargeMedicine);
```

5.8 Appointment Details

```
import React from 'react';
import { connect } from 'react-redux';
import { Link, RouteComponentProps } from 'react-router-dom';
import { Button, Row, Col, Table } from 'reactstrap';
// tslint:disable-next-line:no-unused-variable
import { Translate, ICrudGetAction, TextFormat } from 'react-jhipster';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { getEntity } from './appointment.reducer';
// tslint:disable-next-line:no-unused-variable
import { APP_LOCAL_DATE_FORMAT, APP_DATE_FORMAT, AUTHORITIES }
from 'app/config/constants';
import { hasDoctorRole, hasDoctorRole as isUserDoctor, hasFrontDeskRole } from
'app/shared/auth/private-route';

export interface IAppointmentDetailProps extends StateProps, DispatchProps,
RouteComponentProps<{ id: string }> {}

export class AppointmentDetail extends React.Component<IAppointmentDetailProps>
{
  componentDidMount() {
    this.props.getEntity(this.props.match.params.id);
  }

  render() {
    const { appointmentEntity, authorities } = this.props;
    const isDoctor = isUserDoctor(authorities);
    const isFrontDesk = hasFrontDeskRole(authorities);

    return (
      <Row>
        <Col md="8">
          <h2>
      <Translate
contentKey="tmcApp.appointment.detail.title">Appointment</Translate>
[<b>{appointmentEntity.id}</b>]
     </h2>
     <Table>
      <tbody>
       <tr>
        <td>
         <dl className="jh-entity-details">
          <Row>
           <Col>
            <dt>Patient Name</dt>
           </Col>
           <Col>
            <dd>
             {appointmentEntity.patient ? appointmentEntity.patient.firstName + ' ' +
appointmentEntity.patient.lastName : ''}
            </dd>
           </Col>
          </Row>
          {!isDoctor ? (
           <Row>
            <Col>
             <dt>Doctor Name</dt>
            </Col>
            <Col>
             <dd>
              {appointmentEntity.doctor
               ? appointmentEntity.doctor.user.firstName + ' ' +
appointmentEntity.doctor.user.lastName
               : ''}
             </dd>
            </Col>
           </Row>
          ) : null}
          <Row>
            <Col>
              <dt>
                <span id="allocatedDate">
                  <Translate
contentKey="tmcApp.appointment.allocatedDate">Allocated Date</Translate>
                </span>
              </dt>
            </Col>
            <Col>
              <dd>
                <TextFormat value={appointmentEntity.allocatedDate} type="date"
format={APP_DATE_FORMAT} />
              </dd>
            </Col>
          </Row>
          <Row>
            <Col>
              <dt>
                <span id="title">
                  <Translate contentKey="tmcApp.appointment.title">Title</Translate>
                </span>
              </dt>
            </Col>
            <Col>
              <dd>{appointmentEntity.title}</dd>
            </Col>
          </Row>
          <Row>
            <Col>
              <dt>
                <span id="isClosed">Cancelled?</span>
              </dt>
            </Col>
            <Col>
              <dd>{appointmentEntity.isClosed ? 'Yes' : 'No'}</dd>
            </Col>
          </Row>
          <Row>
            <Col>
              <dt>
                <span id="dateCreated">
                  <Translate contentKey="tmcApp.appointment.dateCreated">Date
Created</Translate>
                </span>
              </dt>
            </Col>
            <Col>
              <dd>
                <TextFormat value={appointmentEntity.dateCreated} type="date"
format={APP_DATE_FORMAT} />
              </dd>
            </Col>
          </Row>
          <Row>
            <Col>
              <dt>
                <span id="dateCreated">
                  <Translate contentKey="tmcApp.appointment.dateUpdated">Date
Updated</Translate>
                </span>
              </dt>
            </Col>
            <Col>
              <dd>
                <TextFormat value={appointmentEntity.dateUpdated} type="date"
format={APP_DATE_FORMAT} />
```

```
                  </dd>
                </Col>
              </Row>
            </dl>
          </td>
        </tr>
      </tbody>
    </Table>
    <Button tag={Link} to="/entity/appointment" replace color="info">
      <FontAwesomeIcon icon="arrow-left" />{' '}
      <span className="d-none d-md-inline">
        <Translate contentKey="entity.action.back">Back</Translate>
      </span>
    </Button>
    &nbsp;
    {!appointmentEntity.isClosed && !appointmentEntity.prescriptionId &&
(isFrontDesk || isDoctor) ? (
      <>
        <Button tag={Link}
to={`/entity/appointment/${appointmentEntity.id}/cancel`} color="danger" size="sm">
          <FontAwesomeIcon icon="pencil-alt" />{' '}
          <span className="d-none d-md-inline">
            <Translate contentKey="entity.action.cancel">Cancel</Translate>
          </span>
        </Button>
        &nbsp;&nbsp;
        {isFrontDesk ? (
          <Button tag={Link} to={`/entity/appointment/${appointmentEntity.id}/edit`}
replace color="primary">
            <FontAwesomeIcon icon="pencil-alt" />{' '}
            <span className="d-none d-md-inline">
              <Translate contentKey="entity.action.edit">Edit</Translate>
            </span>
          </Button>
```

```
            ) : null}
          </>
        ) : null}
      </Col>
    </Row>
  );
 }
}

const mapStateToProps = storeState => ({
 appointmentEntity: storeState.appointment.entity,
 authorities: storeState.authentication.account.authorities
});

const mapDispatchToProps = { getEntity };

type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof mapDispatchToProps;

export default connect(
 mapStateToProps,
 mapDispatchToProps
)(AppointmentDetail);
```

5.9 Diagnosis

```
import React from 'react';
import InfiniteScroll from 'react-infinite-scroller';
import { connect } from 'react-redux';
import { Link, RouteComponentProps } from 'react-router-dom';
import { Button, Col, Row, Table } from 'reactstrap';
// tslint:disable-next-line:no-unused-variable
import { Translate, ICrudGetAllAction, TextFormat, getSortState, IPaginationBaseState
} from 'react-jhipster';
```

```
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';

import { IRootState } from 'app/shared/reducers';
import { getEntities, reset } from './diagnosis.reducer';
import { IDiagnosis } from 'app/shared/model/diagnosis.model';
// tslint:disable-next-line:no-unused-variable
import { APP_DATE_FORMAT, APP_LOCAL_DATE_FORMAT } from
'app/config/constants';
import { ITEMS_PER_PAGE } from 'app/shared/util/pagination.constants';

export interface IDiagnosisProps extends StateProps, DispatchProps,
RouteComponentProps<{ url: string }> {}

export type IDiagnosisState = IPaginationBaseState;

export class Diagnosis extends React.Component<IDiagnosisProps, IDiagnosisState> {
  state: IDiagnosisState = {
    ...getSortState(this.props.location, ITEMS_PER_PAGE)
  };

  componentDidMount() {
    this.reset();
  }

  componentDidUpdate() {
    if (this.props.updateSuccess) {
      this.reset();
    }
  }

  reset = () => {
```

```
  this.props.reset();
  this.setState({ activePage: 1 }, () => {
    this.getEntities();
  });
};

handleLoadMore = () => {
  if (window.pageYOffset > 0) {
    this.setState({ activePage: this.state.activePage + 1 }, () => this.getEntities());
  }
};

sort = prop => () => {
  this.setState(
    {
      order: this.state.order === 'asc' ? 'desc' : 'asc',
      sort: prop
    },
    () => {
      this.reset();
    }
  );
};

getEntities = () => {
  const { activePage, itemsPerPage, sort, order } = this.state;
  this.props.getEntities(activePage - 1, itemsPerPage, `${sort},${order}`);
};

render() {
    const { diagnosisList, match } = this.props;
    return (
      <div>
        <h2 id="diagnosis-heading">
          <Translate contentKey="tmcApp.diagnosis.home.title">Diagnoses</Translate>
          <Link to={`${match.url}/new`} className="btn btn-primary float-right jh-
create-entity" id="jh-create-entity">
            <FontAwesomeIcon icon="plus" />
            &nbsp;
            <Translate contentKey="tmcApp.diagnosis.home.createLabel">Create new
Diagnosis</Translate>
          </Link>
        </h2>
        <div className="table-responsive">
          <InfiniteScroll
            pageStart={this.state.activePage}
            loadMore={this.handleLoadMore}
            hasMore={this.state.activePage - 1 < this.props.links.next}
            loader={<div className="loader">Loading ...</div>}
            threshold={0}
            initialLoad={false}
          >
            <Table responsive>
              <thead>
                <tr>
                  <th className="hand" onClick={this.sort('id')}>
                    <Translate contentKey="global.field.id">ID</Translate>
<FontAwesomeIcon icon="sort" />
                  </th>
                  <th className="hand" onClick={this.sort('name')}>
                    <Translate contentKey="tmcApp.diagnosis.name">Name</Translate>
<FontAwesomeIcon icon="sort" />
```

```
                </th>
                <th className="hand" onClick={this.sort('dateCreated')}>
                  <Translate contentKey="tmcApp.diagnosis.dateCreated">Date
Created</Translate> <FontAwesomeIcon icon="sort" />
                </th>
                <th>
                  <Translate contentKey="tmcApp.diagnosis.lastModifiedBy">Last
Modified By</Translate> <FontAwesomeIcon icon="sort" />
                </th>
                <th />
              </tr>
            </thead>
            <tbody>
              {diagnosisList.map((diagnosis, i) => (
                <tr key={`entity-${i}`}>
                  <td>
                    <Button tag={Link} to={`${match.url}/${diagnosis.id}`} color="link"
size="sm">
                      {diagnosis.id}
                    </Button>
                  </td>
                  <td>{diagnosis.name}</td>
                  <td>
                    <TextFormat type="date" value={diagnosis.dateCreated}
format={APP_DATE_FORMAT} />
                  </td>
                  <td>
                    {diagnosis.lastModifiedBy ? diagnosis.lastModifiedBy.user.firstName + ' '
+ diagnosis.lastModifiedBy.user.lastName : 'SYSTEM'}
                  </td>
                  <td className="text-right">
                    <div className="btn-group flex-btn-group-container">
                    <Button tag={Link} to={`${match.url}/${diagnosis.id}`} color="info"
size="sm">
                      <FontAwesomeIcon icon="eye" />{' '}
                      <span className="d-none d-md-inline">
                        <Translate contentKey="entity.action.view">View</Translate>
                      </span>
                    </Button>
                    <Button tag={Link} to={`${match.url}/${diagnosis.id}/edit`}
color="primary" size="sm">
                      <FontAwesomeIcon icon="pencil-alt" />{' '}
                      <span className="d-none d-md-inline">
                        <Translate contentKey="entity.action.edit">Edit</Translate>
                      </span>
                    </Button>
                    <Button tag={Link} to={`${match.url}/${diagnosis.id}/delete`}
color="danger" size="sm">
                      <FontAwesomeIcon icon="trash" />{' '}
                      <span className="d-none d-md-inline">
                        <Translate contentKey="entity.action.delete">Delete</Translate>
                      </span>
                    </Button>
                  </div>
                </td>
              </tr>
            ))}
          </tbody>
        </Table>
      </InfiniteScroll>
    </div>
  </div>
 );
}
```

```
}

const mapStateToProps = ({ diagnosis }: IRootState) => ({
  diagnosisList: diagnosis.entities,
  totalItems: diagnosis.totalItems,
  links: diagnosis.links,
  entity: diagnosis.entity,
  updateSuccess: diagnosis.updateSuccess
});

const mapDispatchToProps = {
  getEntities,
  reset
};

type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof mapDispatchToProps;

export default connect(
  mapStateToProps,
  mapDispatchToProps
)(Diagnosis);
```

CHAPTER 6

SOFTWARE TESTING

6.1 Software Testing

Software testing is a critical element of software quality assurance and represents the ultimate reuse of specification. Design and code testing represents interesting anomaly for the software during earlier definition and development phase, it was attempted to build software from an abstract concept to tangible implementation.

The testing phase involves, testing of the development of the system using various techniques such as White Box Testing, Control Structure Testing.

6.2 Software Strategies

A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level against customer requirements.

1) Unit Testing
2) Integration testing
3) System testing.
4) Acceptance testing.

6.3. Testing Techniques:

6.3.1 White Box Testing:

White box testing is a test case design method that uses the control structure of the procedural design to derive test cases.

6.3.2 Control Structure Testing

The following tests were conducted and it was noted that the BCBS is performing them well.

- Basic path Testing
- Condition Testing
- Data Flow Testing
- Loop Testing

Implementation:

There are many type of testing including in white box and black box testing.

Usability Testing:

- Usability testing is nothing but the User-friendliness check.

- In Usability testing, the application flow is tested so that a new user can understand the application easily.

- Basically, system navigation is checked in Usability testing.

Usability Test Cases:

- Web page content should be correct without any spelling or grammatical errors.

- All fonts should be same as per the requirements.

- All the text should be properly aligned.

- All the error messages should be correct without any spelling or grammatical errors and the error message should match with the field label.

- Tool tip text should be there for every field.

- All the fields should be properly aligned.

- Enough space should be provided between field labels, columns, rows, and error messages.

- All the buttons should be in a standard format and size.

- Home link should be there on every single page.

- Disabled fields should be greyed out.

- Check for broken links and images.
- Confirmation message should be displayed for any kind of update and delete operation.
- Check the site on different resolutions (640 x 480, 600x800 etc.?)
- Check the end user can run the system without frustration.
- Check the tab should work properly.
- Scroll bar should appear only if required.
- If there is an error message on submit, the information filled by the user should be there.
- Title should display on each web page
- All fields (Textbox, dropdown, radio button, etc) and buttons should be accessible by keyboard shortcuts and the user should be able to perform all operations by using keyboard.
- Check if the dropdown data is not truncated due to the field size. Also, check whether the data is hardcoded or managed via administrator

Compatibility Testing-

Compatibility testing is used to determine if your software is compatible with other elements of a system with which it should operate, e.g. Browsers, Operating Systems, or hardware.

Compatibility Test Scenarios:

- Test the website in different browsers (IE, Firefox, Chrome, Safari and Opera) and ensure the website is displaying properly.
- Test the HTML version being used is compatible with appropriate browser vers10ns.
- Test the images display correctly in different browsers.
- Test the fonts are usable in different browsers.
- Test the java script code is usable in different browsers.
- Test the Animated GIF's across different browsers.

Database Testing--

In Database testing backend records are tested which have been inserted through the web or desktop applications.

The data which is displaying in the web application should match with the data stored in the Database.

Test Cases for Database Testing:

- Verify the database name: The database name should match with the specifications.
- Verify the Tables, columns, column types and defaults: All things should match with the specifications.
- Verify whether the column allows a null or not.
- Verify the Primary and foreign key of each table.
- Verify the Stored Procedure:
- Test whether the Stored procedure is installed or not.
- Verify the Stored procedure name
- Verify the parameter names, types and number of parameters.
- Test the parameters if they are required or not.
- Test the stored procedure by deleting some parameters
- Test when the output is zero, the zero records should be affected.
- Test the stored procedure by writing simple SQL queries.
- Test whether the stored procedure returns the values
- Test the stored procedure with sample input data.
- Verify the behaviour of each flag in the table.
- Verify the data gets properly saved into the database after each page submission.
- Verify the data if the DML (Update, delete and insert) operations are performed.

- Check the length of every field: The field length in the back end and front end must be same.

- Verify the database names of QA, UAT and production. The names should be umque.

- Verify the encrypted data in the database.

- Verify the database size. Also test the response time of each query executed.

- Verify the data displayed on the front end and make sure it is same in the back end.

- Verify the data validity by inserting the invalid data in the database.

- Verify the Triggers.

What is Security Testing?

Security Testing involves the test to identify any flaws and gaps from a security point of view.

Test Scenarios for Security Testing:

- Verify the web page which contains important data like password, credit card numbers, secret answers for security question etc should be submitted via HTTPS (SSL).

- Verify the important information like password, credit card numbers etc should display in encrypted format.

- Verify password rules are implemented on all authentication pages like Registration, forgot password, change password.

- Verify if the password is changed the user should not be able to login with the old password.

- Verify the error messages should not display any important information.

- Verify if the user is logged out from the system or user session was expired, the user should not be able to navigate the site.

- Verify to access the secured and non-secured web pages directly without login.

- Verify the "View Source code" option is disabled and should not be visible to the user.

CHAPTER 7

MAINTENANCE

7.1 Introduction

Software maintenance refers to the process of modifying and updating a software system after it has been delivered to the customer. This can include fixing bugs, adding new features, improving performance, or updating the software to work with new hardware or software systems.

The goal of software maintenance is to keep the software system working correctly, efficiently, and securely, and to ensure that it continues to meet the needs of the users.

Maintenance is required to maintain a building's initial performance capacity. Without maintenance, performance will not meet the demand and eventually will drop below the limit of acceptance of residents. In practice, both the demand and the limit of acceptance will gradually rise over time as a result of improved technology, rising standards, and growing prosperity.

Improvement and renewal are required to answer the accordingly rising expectations. As a result, the total life cycle costs will generally be a multiple of the initial building costs. Maintenance is a combination of all technical and associated administrative actions during the service life to retain a building or its parts in a state in which it can perform its required functions.

7.2. Administrator

Work for the cookies on patient appointment booking maintenance due to manage restaurant management system for the order to provide you are easily found free of the future.

Weaknesses and if the project patient appointment booking system as per your consent prior to batch mode execution. Efforts of this project on patient appointment booking is totally built using projects published on rails management system project is high as the team captain.

Thesis to that the project report on patient appointment booking due to view menu is not available. Task schedule and the working on maintenance system for booking, customer online role playing area of patient appointment booking is a wide variety of stations.

Possible features that, project report on patient appointment booking online, and may resume the restaurant management system is not have source code of the customer make the details. Materials in the cookies on patient appointment booking due to this project in proper resource management system will view the modern periodic table. Especially zip so, project on patient appointment booking is important to store and reservation table project with monitoring the purpose of these cookies are categorized as the source software.

Printing and the working on patient appointment booking system can occur without the screen and the time. At all is a report on patient appointment booking system allows customer was developed by using the administration. Website and monitor the project on patient appointment booking project of charges customer will need a web application.

7.3. Types of website maintenance services

1) Updating Website Software:

If you are using any kind of content management system like Word Press or any kind of script like PHP, it needs to always be up to date. Word Press usually updates itself automatically but there are instances where it might need to be updated manually.

Additionally, if you are using a website host, a web service maintenance provider will ensure that they are updating their core server software on a regular basis. For instance, they will check if things like FTP service running on the server are up to date. If however, you run your own web server, they will manually test and apply updates.

2) Improving Website Speed:

Did you know that twenty five percent of users navigate out of a website if it takes more than three seconds to load? Page loading is obviously an important part of the user experience. However, we tend to let it slide in order to accommodate new nifty functionality or aesthetic website design.

In reality, web visitors do not care about all the bells and whistles. They care about accessing the information they want as fast as possible. Another important factor to consider is how page load speed affects your search engine rankings. A slow website increases bounce rate and if people are navigating out of your web page as fast as they got there, then Google will think your website isn't important so they will push it further down the rankings.

3) Fixing HTML Errors

HTML Code isn't easy especially if you know nothing about website coding. You need to pay attention to detail to ensure that it's working well. What might seem like a minor error could lead to a number of problems such as funky looking pages, missing multimedia, etc. Reliable website support should be able to inspect,
find and fix issues related to HTML.

4) Backing up Files:

Imagine how frustrating it would be if you were to lose all your website's essential files.

This includes images, pages, blog posts, plugging, etc. If you run your website on Word Press, you might have a few automated backup options. Website maintenance services ensure that file and database backups is automatically performed at least on a weekly basis.

5) Developing New Content:

Your content strategy might be on point but it's always a good idea to improve it by creating new types of content. Your web content also needs to be up to date. If you wrote a post that was published years ago, it might not be relevant right now and some of the information might need to be updated.

6) Search Engine Optimization:

Regularly checking your website for areas where search ability can be improved can go a long way to improving search engine optimization. Your blog posts can be altered to include popular and competitive keywords. Your landing pages can also be updated to include more relevant and timely keywords. Even you're "About Us" page can be updated regularly to take advantage of popular keywords in your industry. Keyword research is also important when creating new content for your website.

As part of the SEO strategy, it's also important to identify structural issues within your website that may affect how search engines view your site.

7) Ensuring Design Consistency across All Pages:

Your website should function properly in all the latest versions of major website browsers like Chrome, Safari, Firefox, etc. These browsers update frequently and if your website doesn't adapt to the updates it might not show up or it might not look the way you want it to. This could negatively affect your business and you're branding. It's also important to ensure that you have a responsive website design that looks good on mobile devices.

Another factor to consider about your website's design is uniformity. If you have a lot of branding elements that represent your company, they should be included uniformly across all your pages.

8) Fixing Broken Links:

Every link on your website should lead exactly to the right place but sometimes links can go bad. A site or resource you linked to earlier might disappear, a page on

your website might become unavailable or you might move a post and forget to update others that link to it.

Broken links can be detrimental to your website. They could give visitors the impression that:

a) Your website is not user friendly

b) You don't regularly update your website

c) You are not credible

9) Reviewing Your Website Analytics:

Your website's performance needs to be monitored to ensure that everything is working as it should. An experienced website support specialist will examine high- level metrics like how many new and returning visitors your website has received for the past week or which blog posts are the most read. They will then make adjustments your website accordingly to ensure all the metrics are at their peak.

This is just a minimum starting point of what website maintenance is all about. Some of these tasks can be overly time consuming. Additionally, if you do not know what you are doing, you could end up doing more damage. The better option is to hire professional website maintenance services.

CHAPTER 8

CONCLUSION

Our project is only a humble venture to satisfy the needs to manage their project work. Several user friendly coding have also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the school. The objective of software planning is to provide a frame work that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

At the end it is concluded that we have made effort on following points...

- A description of the background and context of the project and its relation to work already done in the area.
- Made statement of the aims and objectives of the project.
- The description of Purpose, Scope, and applicability.
- We define the problem on which we are working in the project.
- We describe the requirement Specifications of the system and the actions that can be done on these things.
- We understand the problem domain and produce a model of the system, which describes operations that can be performed on the system.
- We included features and operations in detail, including screen layouts.
- We designed user interface and security issues related to system.
- Finally the system is implemented and tested according to test cases.

CHAPTER 9

FUTURE SCOPE

Future scope of Patient Appointment Booking

In a nutshell, it can be summarized that the future scope of the project circles around maintaining information regarding:

- We can add printer in future.
- We can give more advance software for Patient Appointment Booking including more facilities
- We will host the platform on online servers to make it accessible worldwide
- Integrate multiple load balancers to distribute the loads of the system
- Create the master and slave database structure to reduce the overload of the database queries
- Implement the backup mechanism for taking backup of codebase and database on regular basis on different servers

The above mentioned points are the enhancements which can be done to increase the applicability and usage of this project. Here we can maintain the records of Doctor and Appointment Also, as it can be seen that now-a-days the players are versatile, that is so there is a scope for introducing met maintain the Patient Appointment Booking. Enhancements can be done to maintain all the Doctor, Appointment, Patient, Booking, Doctor Schedule.

We have left all the options open so that if there is any other future requirement in the system by the user for the enhancement of the system then it is possible to implement them .In the last we would like to thanks all the persons involved in the development of the system directly or indirectly. We hope that the project will serve its purpose for which it is develop there by underlining success of process.

CHAPTER 10

REFERENCES

2. 1.Bailey NTJ. A study of queues and appointment systems in hospital out-patient departments, with special reference to waiting times. J Royal Stat Soc 1952;14:185–99
3. Cayirli, T, E. Veral, and H. Rosen. (2006). Designing appointment scheduling systems for ambulatory care services. Health Care Management Science 9, 47–58.
4. Adebayo Peter Idowu., OlajideOlusegunAdeosun., and KehindeOladipo Williams.,“Dependable Online Appointment Booking System for Outpatient in Nigerian Teaching Hospitals" International Journal of Computer Science & Information Technology (IJCSIT) Vol.6(4),pp.109-116,2014.
5. Arthur Hylton III and Suresh Sankaran arayanan “Application of Intelligent Agents in Hospital Appointment Scheduling System”, International Journal of Computer Theory and Engineering, Vol. 4, August 2012, pp. 625-630.
6. Yeo Symey, Suresh Sankaran arayanan, Siti Nurafifah binti Sait “Application of Smart Technologies for Mobile Patient Appointment System”, International Journal of Advanced Trends in Computer Science and Engineering, august 2013
7. Jagannath Aghav, Smita Sonawane, and Himanshu Bhambhlani “Health Track: Health Monitoring and Prognosis System using Wearable Sensors”, IEEE International Conference on Advances in Engineering & Technology Research 2014, pp. 1-5.
8. Nazia S, EktaSarda (2014) "Online Appointment Scheduling System for Hospitals– An Analytical Study",IJIET, Vol. 4:1.
9. S Sri Gowthem, K P Kaliyamurthie (2015) "Smart Appointment
10. Reservation System", IJIRSET, Vol.4:6.

BIBLIOGRAPHY

1. https://mui.com/
2. https://www.w3schools.com/REACT/DEFAULT.ASP
3. https://legacy.reactjs.org/tutorial/tutorial.html
4. https://www.javatpoint.com/reactjs-tutorial
5. https://react-tutorial.app/
6. https://react.dev/learn/react-developer-tools
7. https://www.npmjs.com/package/react-devtools
8. https://reactnative.dev/docs/next/react-devtools