

**Training Report**

**On**

**HOSPITAL MANAGEMENT SYSTEM**

**DECLARATION**

I hereby certify that the training work, in partial fulfillment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING submitted to the **CENTRE OF RAILWAY INFORMATION SYSTEM, ITO, DELHI,** is an authentic record of our work carried out during a period from **MAY**, **2019** to **JULY2019,** under the supervision of **MR. SUNIL KUMAR**, Project Engineer, Department of RFID. The matter presented in this project has been submitted only for experience gaining purposes.

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

Date: **Mr. Sunil Kumar**

Project Guide

**CERTIFICATE**

This is to certify that the training work submitted by Guddy kumari to the Centre for Railway Information System, is a bonafide record of training work carried out by him under my supervision and guidance. The contents of the report or the project, in full or part, have not been shared with any other organization.

(Mr. Sunil Kumar) Date:

(PE)

**ACKNOWLEDGEMENT**

I wish to express my sincere gratitude to my guide **Mr. Sunil Kumar**, for all the advice, encouragement and constant support he has given me throughout the training work.

I thank **Mr. Partha Majumdar** (Project Head of the Department) for providing the necessary facilities to carry out the project.

I am also grateful to **Mr. Tanmay Mehta** (The General Manager) for putting his faith in my abilities and giving me a chance to showcase my skills.

At last I would like to thank all my colleagues for helping me out with the project work.

**ABSTRACT**

1. The purpose of this project is to build a website to help out the patients visiting the **N**orthern **R**ailway **C**entral **H**ospital. The patients prior to this project have had no way to view their reports or make appointments at the hospital from home.
2. The project deals with the collection of information from a patient such as his/her details and provides an efficient way to view reports, book appointments, etc. Traditionally, this was done manually. The main function of the system is to register a patient or a doctor and retrieve those details whenever required.
3. The key idea is to make a website which can work efficiently for a very large scale database management. With many patients checking in and out of the hospital every day, it is a very difficult task to keep updating the records in ledgers.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.no.** |  | **Page no.** |
| **1** | **INTRODUCTION** | **07** |
|  |  |  |
| **2** | **PROBLEM STATEMENT** | **08** |
|  |  |  |
| **4** | **OBJECTIVES** | **08** |
|  |  |  |
| **5** | **METHODOLOGY** | **09** |
|  |  |  |
| **5.1** | **REQUIREMENTS SPECIFICATIONS** | **09** |
|  |  |  |
| **6** | **ANALYSIS** | **10** |
|  |  |  |
| **7** | **SOFTWARE SPECIFICATION** | **11** |
|  |  |  |
| **8** | **TESTING** | **13** |
|  |  |  |
| **9** | **IMPLEMENTATION AND RESULTS** | **15** |
|  |  |  |
| **10** | **REFERENCES** | **41** |

**INTRODUCTION**

1. In the given project, we have carefully analyzed what the requirements of the hospital are from the software and also, how the end user (patient or doctor), can use the software for his/her convenience without having to call or visit the hospital for information such as test reports, fixing appointments, etc.
2. The project has been implemented using technologies such as Node and Angular, which not only provide an efficient way of building up the business logic, but also make it easier for the developers to implement logic that would otherwise take longer duration for implementation as well as testing.
3. The software is easy to use and understand and has been designed to deliver real conceivable benefits to the hospital. The software is designed to improve the quality and management of hospital management in the areas of clinical process analysis and activity-based costing. Hospital Management System enables you to develop your organization and improve its effectiveness and quality of work. Managing the key processes efficiently is critical to the success of the hospital.

**PROBLEM STATEMENT**

1. Lack of immediate retrievals:

The lack of immediate retrievals is one of the biggest problems being faced by hospitals. To view a patient’s history, multiple records have to be checked which makes it very difficult to find and time consuming. This also slows down the entire process.

1. Lack of immediate Information storage:

The newly collected information has to be entered in multiple record books. This makes even the storage of information very tedious. Also, it makes it time consuming and increases the room for errors.

1. Lack of prompt Updating:

To change the information of a particular patient in a haystack of records is yet another challenge faced by the employees of the hospital. This task requires a lot of time and concentration. This again creates room for errors.

1. Tedious Manual Calculations:

Holding the records in multiple places also causes a lot of time wastage when it comes to calculations. For example, to calculate how many medicines are left and how many have to be bought and calculating the amount to buy the medicine requires a lot of labor.

**OBJECTIVES**

1. To record personal details of the patients.
2. Record information related to the diagnosis of a patient.
3. Keeping the records up to date.
4. Patients must be able to view their reports and also check the availability of the doctors.
5. Patients must be able to view their appointments along with the date, time and the name of the doctor (may have to use push messages) as well as make new appointments.
6. To be able to keep a record of the patient’s medical history.

**METHODOLOGY**

**Step 1:** Gathering of requirements

**Step 2:** Creating the front end with Angular cli.

**Step3:** Creating a basic server using Node.js.

**Step4:**  Server side coding for handling the requests (get requests and post requests) using Loopback. Loopback has also been used to create the RESTful API.

**Step5:** Handling form data using Angular directives and pushing it into the database.

**Step 6:** Pulling the data from the database.

**REQUIREMENTS SPECIFICATIONS**

To meet the client requirements, every software must meet the requirements specified by the client. This includes serving both fundamental functionalities as well as additional functionalities.

* Fundamental functionalities:These are the basic functionalities for which the software is being developed.
* Additional functionalities**:** These are the extra requirements that a software may or may not be able to serve.

**Client Requirements**

* Patients must be able to register (Snippets of the screens along with the code are provided in the ‘**Implementation**’ Section).
* Patients should be able to view their reports and his/her past medical history.
* Patient must be able to update his/her personal information.
* Staff should be able to fetch patient data easily.
* Patient should be able to make appointments online.

**Software Requirements**

* **BROWSER SUPPORT :** Chrome/Firefox/IE9 or higher
* **FRONT END :** Angular6, BootStrap4,HTML, CSS
* **BACK END :** Node 8.9, Loopback 3.
* **DATABASE :** MySQL

Efforts have been made to include all the basic functionalities as well as include as much additional functionalities as possible.

**ANALYSIS**

Existing System**:**

NRCH still incorporates old management techniques which includes maintenance of a record book for the appointments made by the patients and also their medical history. This process is not only slow, but also requires extra effort so that the work is done properly.

Proposed System**:**

The website has been designed so that the data can be kept in a regular order and patient records can easily be found without any hassle or time wastage. This makes it easier for both the patients as well as the staff to keep a check of each record without much difficulty. The software is to provide the services in a very efficient and cost-effective manner.

FEASIBILITY STUDY:

Two key things that have been taken into account for this phase are:

* Technical Feasibility:

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

* Operational Feasibility:

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

**SOFTWARE SPECIFICATION**

**Angular 6:**

Angular is a platform that makes it easy to build applications with the web. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop. The JavaScript components complement Apache Cordova, a framework used for developing cross-platform mobile apps. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model–view–controller (MVC) and model–view–view-model (MVVM) architectures, along with components commonly used in rich Internet applications. Angular 6 has many of the APIs inbuilt which makes it all the way easier for the developers to test their applications without any hassle and build robust applications in a very short time.

**Loopback 3.X:**

Loopback is a highly-extensible, open-source Node.js framework that enables us to:

* Create dynamic end-to-end REST APIs with little or no coding.
* Access data from major relational databases such as MySQL, MongoDB.
* Separable components for file storage, and third-party login.

**MySQL:**

MySQL is one of the most commonly used relational databases in the world. The Database. This database makes up the core of the LAMP stack. However, due to its efficiency and simplicity of use, this database is used by many international organizations. Some of the features of MySQL are:

* Written in C and C++.
* Tested with a broad range of different compilers.
* Works on many different platforms.
* Uses multi-layered server design with independent modules.
* Support for large databases. We use MySQL Server with databases that contain 50 million records. We also know of users who use MySQL Server with 200,000 tables and about 5,000,000,000 rows.

**Node 8.9.11:**

Node is one of the most popular server side JavaScript based run-time environments. Initially JavaScript could only be used for front-end (DOM) manipulation purposes but with Node.js, JavaScript has found its way to the server side scripting.

**Commands used to install all these specifications:**

* npm install -g @angular/cli

Install the Angular CLI globally. The Angular CLI installs the necessary npm packages, creates the project files, and populates the project with a simple default app. This can take some time.

* ng new my-app

Generate a new project and default app by running the above command

* cd my-app

ng serve --open

The ng serve command launches the server, watches your files, and rebuilds the app as you make changes to those files. Using the --open (or just -o) option will automatically open your browser on http://localhost:4200/.

* C:\Users\Your Name>npm install MySQL

Once you have MySQL up and running on your computer, you can access it by using Node.js. To access a MySQL database with Node.js, you need a MySQL driver. This tutorial will use the "MySQL" module, downloaded from NPM. To download and install the "MySQL" module, open the Command Terminal and execute the above command.

* ng generate component hero

Run this database connection through this command

* npm install -g loopback-cli

Loopback is a highly-extensible, open-source Node.js framework

* $ npm install -g http://get-studio.strongloop.com/strong-studio.tgz

Run this command to install the strong studio for our project.

* Creating a new loopback project

$ lb

? What's the name of your application? hello-world

? Enter name of the directory to contain the project: hello-world

? Which version of LoopBack would you like to use? 3.x (current)

? What kind of application do you have in mind? hello-world (A project containing a controller,

including a single vanilla Message and a single remote method)

...

I'm all done. Running npm install for you to install the required dependencies.

If this fails, try running the command yourself.

...

* Creating a new model in loopback project

$ lb model

The generator guides you through creating your model. Enter the values highlighted in green. To accept the default, just press Enter.

[?] Enter the model name: person

[?] Select the data-source to attach person to: db (memory)

[?] Select model`s base class (PersistedModel)

[?] Expose person via the REST API? Yes

[?] Custom plural form (used to build REST URL): people

[?] Common model or server only? common

Let's add some person properties now.

**TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a working product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is a process of exercising software with the intent of ensuring that the Software system meets its requirements and the user expectations and does not fail in an unacceptable manner. There are various types of testing techniques. Each test type addresses a specific testing requirement. Some of the testing techniques have been mentioned below.

* **Unit testing:**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

* **Integration testing:**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

* **Functional testing:**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

**Valid Input :**  identified classes of valid input must be accepted.

**Invalid Input :** identified classes of invalid input must be rejected.

**Functions :** identified functions must be exercised.

**Output :** identified classes of outputs must be exercised.

**Systems/Procedures:** interfacing systems or procedures must be invoked.

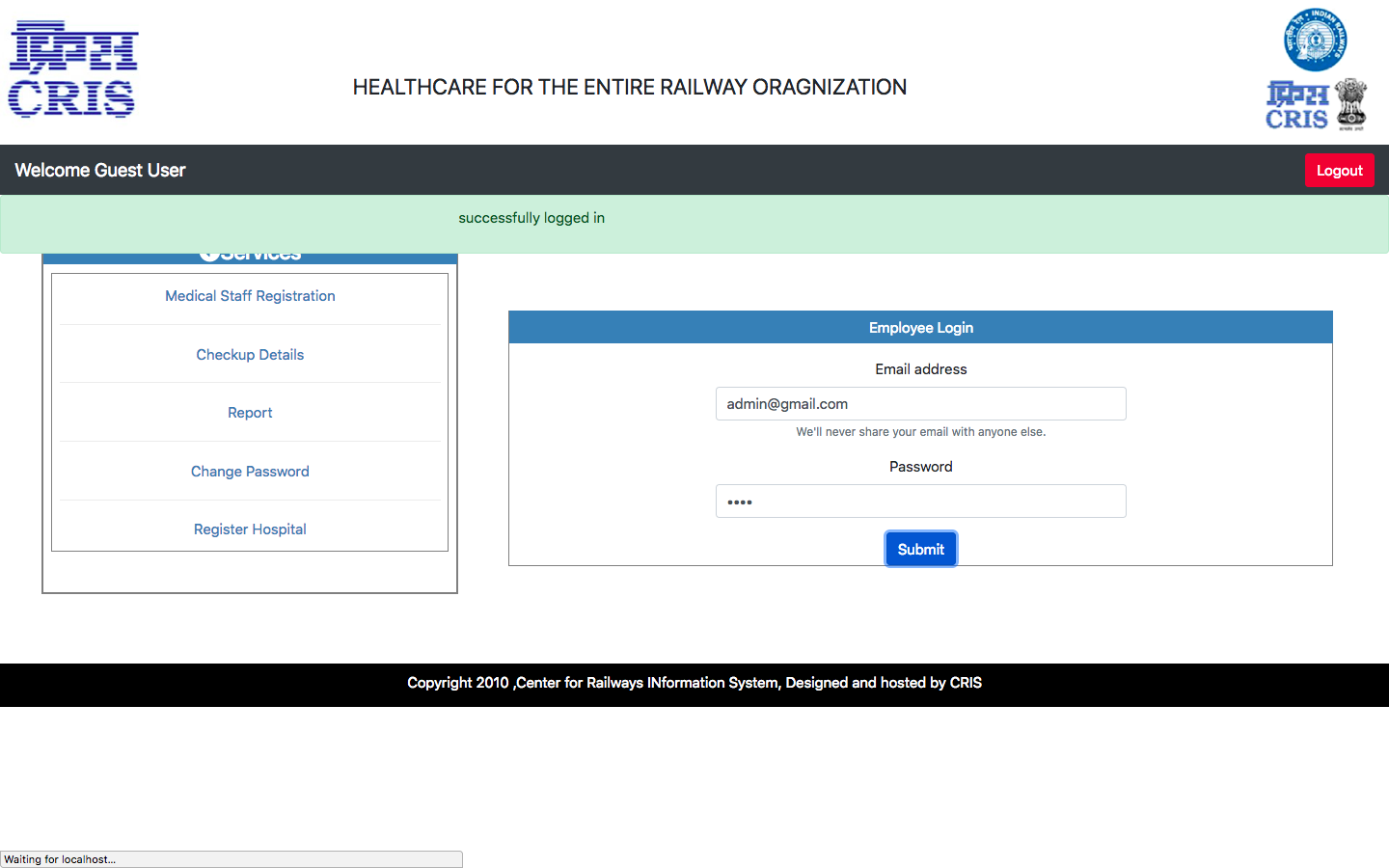
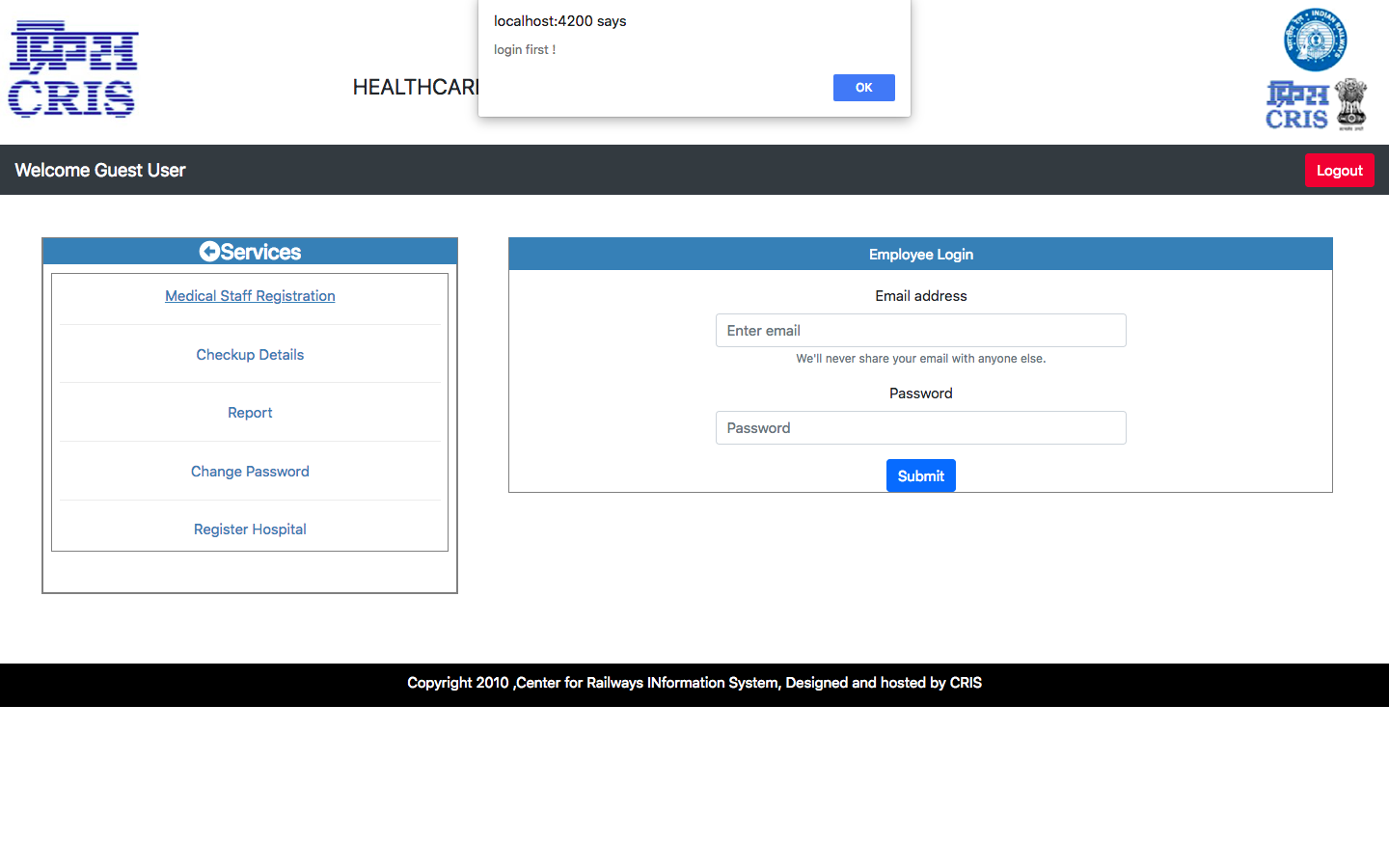
Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

Test Objectives:

* All field entries must work properly.
* Pages must be activated from the respective given link.
* Check the validation of form data
* Check the efficiency of the software under construction.

**IMPLEMENTATION AND RESULTS**

The website covers not only the patients section but also the doctors section. All the parts made by me have been tested and any issues that were found were fixed.

****

**HTML CODE**

<form style="width: 50%;margin-left: 25%;margin-right:25% ">

<div clss="form-group">

<label for="exampleInputEmail1">Email address</label>

<input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter email" [(ngModel)]="emaillog" name="emaillog">

<small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone else.</small>

</div>

<div class="form-group">

<label for="exampleInputPassword1">Password</label>

<input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password" [(ngModel)]="passlog" name="passwlog">

</div>

<button type="submit" (click)="onClickLogin()" class="btn btn-primary">Submit</button>

</form>

**TYPESCRIPT CODE**

ngOnInit() { }

onClickLogin(){

if(this.passlog==undefined || this.emaillog==undefined){

this.flashMessage.show('Please fill all the "Fields"',{cssClass:'alert-danger',timeout:3000});

return false; }

if(!this.validate.validateEmail(this.emaillog)){

this.flashMessage.show('Please fill email in proper email format',{cssClass:'alert-danger',timeout:3000});

return false;

}

var url="http://localhost:3000/api/Medicalstaffs?filter=%7B%22where%22%3A%7B%22email%22%3A%20%22"+this.emaillog+"%22%7D%7D";

this.getData.getemployee(url).then(data=>{

this.employee=data;

if(this.employee[0]==undefined){

this.flashMessage.show('Email not found',{cssClass:'alert-danger',timeout:3000});

}else{

if(this.employee[0].password==this.passlog){

this.flashMessage.show('successfully logged in',{cssClass:'alert-success',timeout:3000});

this.cookieService.removeAll();

localStorage.setItem('emailid',this.emaillog);

localStorage.setItem('status','true');

localStorage.setItem('role',data[0].role);

console.log(localStorage.getItem('status'))

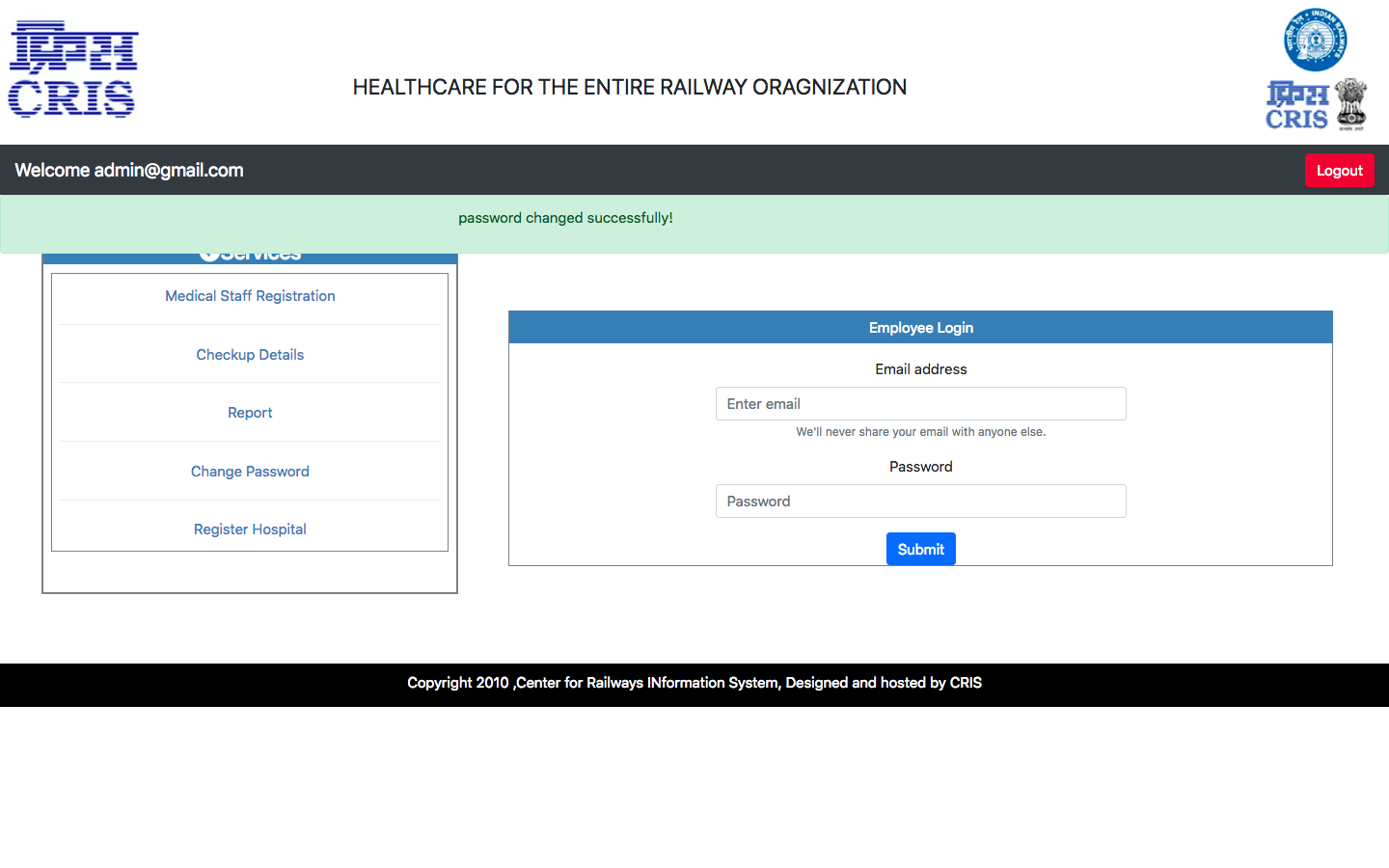
window.location.href="http://localhost:4200/reports";

}else if(this.employee[0].password!=this.passlog){

this.flashMessage.show('Email and password dont match',{cssClass:'alert-danger',timeout:3000}); } }

},err=>{

console.log(err); }) }

****

**HTML CODE**

<form>

<div class="form-group" style="margin-left:10%;margin-right:10%;margin-top:5%;width: 80%;">

<input type="password" class="form-control" aria-describedby="emailHelp" placeholder="Old Password" [(ngModel)]="oldpass" name="oldpass">

</div>

<div class="form-group" style="width: 80%;margin-left:10%;margin-right:10%">

<input type="password" class="form-control" placeholder="New Password" [(ngModel)]="newpass" name="newpass">

</div>

<div class="form-group" style="width: 80%;margin-left:10%;margin-right:10%">

<input type="password" class="form-control" placeholder="Confirm New Password" [(ngModel)]="confirmpass" name="confirmpass">

</div>

<button type="submit" style="width:80%;margin-left: 10%;margin-right: 10%;margin-bottom: 20%" class="btn btn-primary" (click)="onclicksubmit()">Change!</button>

</form>

**TYPESCRIPT CODE**

ngOnInit() {

if(localStorage.getItem('status')=='false'){

alert('login first !');

this.router.navigate(['/']);

}

}

onclicksubmit(){

var url="http://localhost:3000/api/Medicalstaffs?filter=%7B%22where%22%3A%7B%22email%22%3A%20%22"+localStorage.getItem('emailid')+"%22%7D%7D";

this.getdata.getAllData(url).then(data=>{

this.staff=data[0];

console.log(this.staff);

if(this.staff.password==this.oldpass){

if(this.oldpass==undefined || this.newpass==undefined || this.confirmpass==undefined){

this.flash.show('Fill all the fields',{cssClass:'alert-danger',timeout:3000});

return;

}

if(this.newpass==this.confirmpass){

this.staff.password=this.newpass;

url="http://localhost:3000/api/Medicalstaffs";

this.sendData.modifyData(url,this.staff).then(data=>{

console.log(data);

},err=>{

console.log(err);

})

this.flash.show('password changed successfully!',{cssClass:'alert-success',timeout:3000});

}else{

this.flash.show('Confirm password does not match',{cssClass:'alert-danger',timeout:3000});

}

}else{

this.flash.show('Entered wrong old password!',{cssClass:'alert-danger',timeout:3000});

}

localStorage.setItem('status','false');

localStorage.setItem('emailid','');

localStorage.setItem('role','');

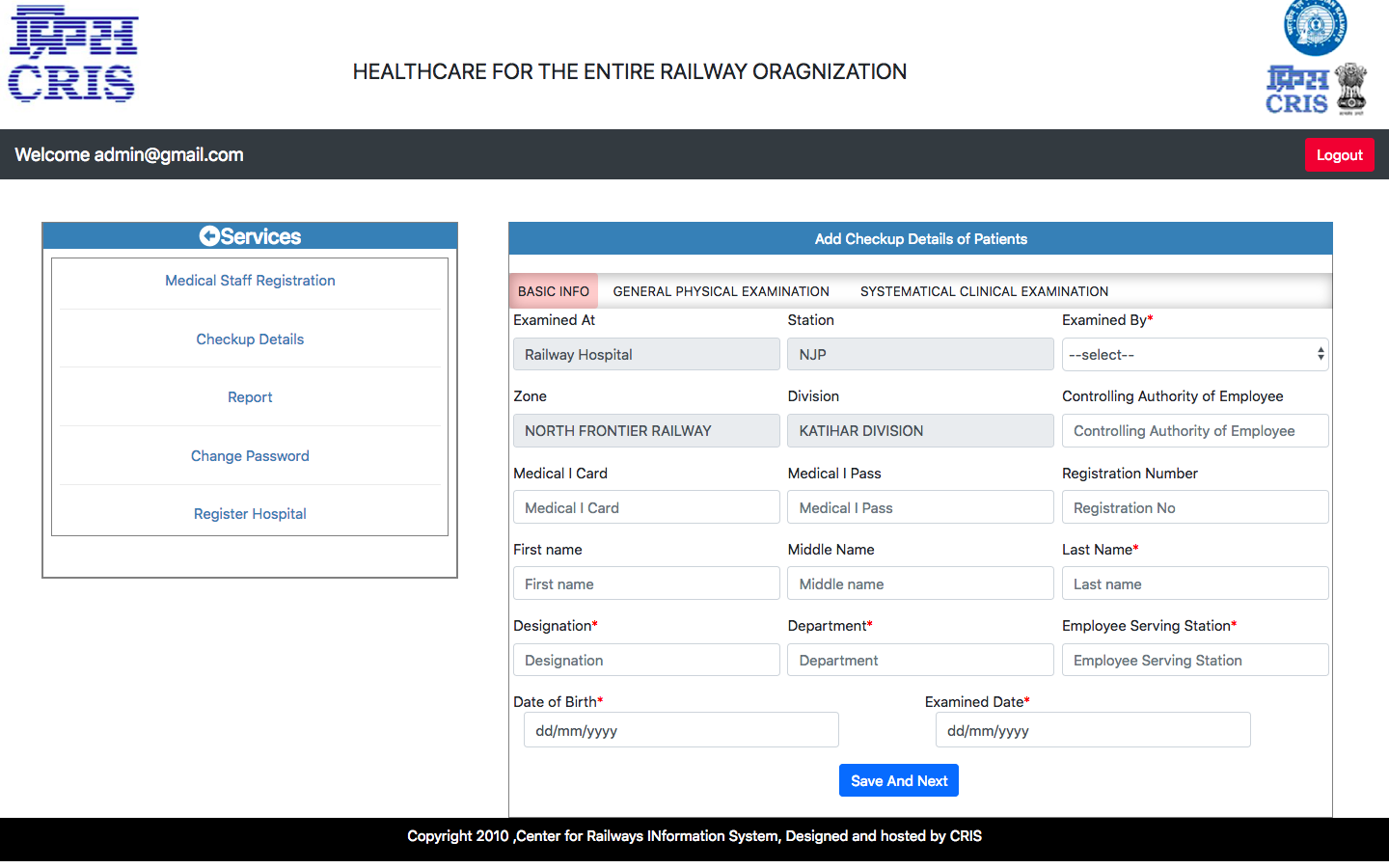
this.router.navigate(['/']);

},err=>{

console.log(err);

})

}



**HTML CODE**

<form style="padding: 5px" >

<div id="box" class="form-row">

<span style="background-color: rgb(255,0,0,0.2);margin:0;border-radius: 5px">BASIC INFO</span>

<span>GENERAL PHYSICAL EXAMINATION</span>

<span>SYSTEMATICAL CLINICAL EXAMINATION</span>

<!-- <span>INVESTIGATION</span>

<span>FINAL SUBMIT</span> -->

</div>

<div class="form-row" >

<div class="col-md-4 mb-3" >

<label for="validationDefault01">Examined At</label>

<input type="text" class="form-control" readonly id="validationDefault01" value="Railway Hospital" required>

</div>

<div class="col-md-4 mb-3" >

<label for="validationDefault02">Station</label>

<input type="text" class="form-control" readonly id="validationDefault02" value="NJP" required>

</div>

<div class="col-lg-4 col-md-4">

<label for="inputState" class="label">Examined By</label>

<select id="inputState" class="form-control" [(ngModel)]="examinedby" name="examinedby">

<option selected>--select--</option>

<option \*ngFor="let doctor of doctors">{{doctor.firstname}} {{doctor.middlename}} {{doctor.lastname}}</option>

</select>

</div>

</div>

<div class="form-row">

<div class="col-md-4 mb-3">

<label for="validationDefault03">Zone</label>

<input type="text" class="form-control" readonly id="validationDefault03" value="NORTH FRONTIER RAILWAY" required>

</div>

<div class="col-md-4 mb-3">

<label for="validationDefault04">Division</label>

<input type="text" class="form-control " readonly id="validationDefault04" value="KATIHAR DIVISION" required>

</div>

<div class="col-md-4 mb-3">

<label for="validationDefault04">Controlling Authority of Employee</label>

<input type="text" class="form-control " id="validationDefault04" placeholder="Controlling Authority of Employee" required [(ngModel)]="controllingauthority" name="controllingauthority">

</div>

</div>

<div class="form-row">

<div class="col-md-4 mb-3" style="width:0">

<label for="validationDefault03">Medical I Card</label>

<input type="text" class="form-control" id="validationDefault03" placeholder="Medical I Card" required [(ngModel)]="medicalicard" name="medicalicard">

</div>

<div class="col-md-4 mb-3">

<label for="validationDefault04">Medical I Pass</label>

<input type="text" class="form-control " id="validationDefault04" placeholder="Medical I Pass" required [(ngModel)]="medicalpass" name="medicalpass">

</div>

<div class="col-md-4 mb-3">

<label for="validationDefault04">Registration Number</label>

<input type="text" class="form-control " id="validationDefault04" placeholder="Registration No" required [(ngModel)]="registrationno" name="registrationno">

</div>

</div>

<div class="form-row">

<div class="col-md-4 mb-3" >

<label for="validationDefault01">First name</label>

<input type="text" class="form-control" id="validationDefault01" placeholder="First name" required [(ngModel)]="firstname" name="firstname">

</div>

<div class="col-md-4 mb-3" >

<label for="validationDefault02">Middle Name</label>

<input type="text" class="form-control" id="validationDefault02" placeholder="Middle name" required [(ngModel)]="middlename" name="middlename">

</div>

<div class="col-md-4 mb-3" >

<label for="validationDefault02" class="label">Last Name</label>

<input type="text" class="form-control" id="validationDefault02" placeholder="Last name" required [(ngModel)]="lastname" name="lastname">

</div>

</div>

<div class="form-row">

<div class="col-md-4 mb-3" >

<label for="validationDefault01" class="label">Designation</label>

<input type="text" class="form-control" id="validationDefault01" placeholder="Designation" required [(ngModel)]="designation" name="designation">

</div>

<div class="col-md-4 mb-3" >

<label for="validationDefault02" class="label">Department</label>

<input type="text" class="form-control" id="validationDefault02" placeholder="Department" required [(ngModel)]="department" name="department">

</div>

<div class="col-md-4 mb-3" >

<label for="validationDefault02" class="label">Employee Serving Station</label>

<input type="text" class="form-control" id="validationDefault02" placeholder="Employee Serving Station" required [(ngModel)]="servingstation" name="servingstation">

</div>

</div>

<div class="form-row" >

<div class="col-lg-6 col-md-6">

<div class="label">Date of Birth</div>

<div class="col-10">

<input class="form-control" type="date" id="example-date-input" [(ngModel)]="DOB" name="DOB">

</div>

</div>

<div class="col-lg-6 col-md-6" >

<div class="label" class="label">Examined Date</div>

<div class="col-10">

<input class="form-control" type="date" id="example-date-input" [(ngModel)]="examinationdate" name="examinationdate">

</div>

</div>

</div>

<button class="btn btn-primary" type="submit" style="margin-left: 40%;margin-right:20% ;margin-top: 2%;margin-bottom: 2%" (click)="onclicksubmit()">Save And Next</button>

</form>

**TYPESCRIPT CODE**

ngOnInit()

if(localStorage.getItem('status')=='false'){

alert('login first !');

this.router.navigate(['/']);

}else if(localStorage.getItem('status')=='true' && localStorage.getItem('role')=='Nurse'){

alert('You have insufficient rights to access this content');

this.router.navigate(['/']);

}

this.getData.getAllData("http://localhost:3000/api/Medicalstaffs?filter=%7B%22where%22%3A%7B%20%22role%22%3A%20%22Doctor%22%7D%7D").then(data=>{

this.doctors=data;

},err=>{

console.log(err);

})

}

onclicksubmit(){

var url="http://localhost:3000/api/Employeebasicinfos";

let employee={

"controllingauthority": this.controllingauthority,

"dataofbirth": this.DOB,

"datenow": new Date().getTime(),

"department": this.department,

"designation": this.designation,

"division": "KATIHAR DIVISION",

"eid": "E"+Date.now(),

"employeeservicestation": this.servingstation,

"examinedat": "Railway Hospital",

"examinedby": this.examinedby,

"examineddate": this.examinationdate,

"firstname": this.firstname,

"lastname": this.lastname,

"medicalcard":this.medicalicard ,

"medicalpass": this.medicalpass,

"middlename": this.middlename,

"registrationno": this.registrationno,

"station": "NJP",

"zone": "NORTH FRONTIER RAILWAY"

}

this.validate.eidInstance(employee.eid);

this.sendData.sendData(url,employee).then(data=>{

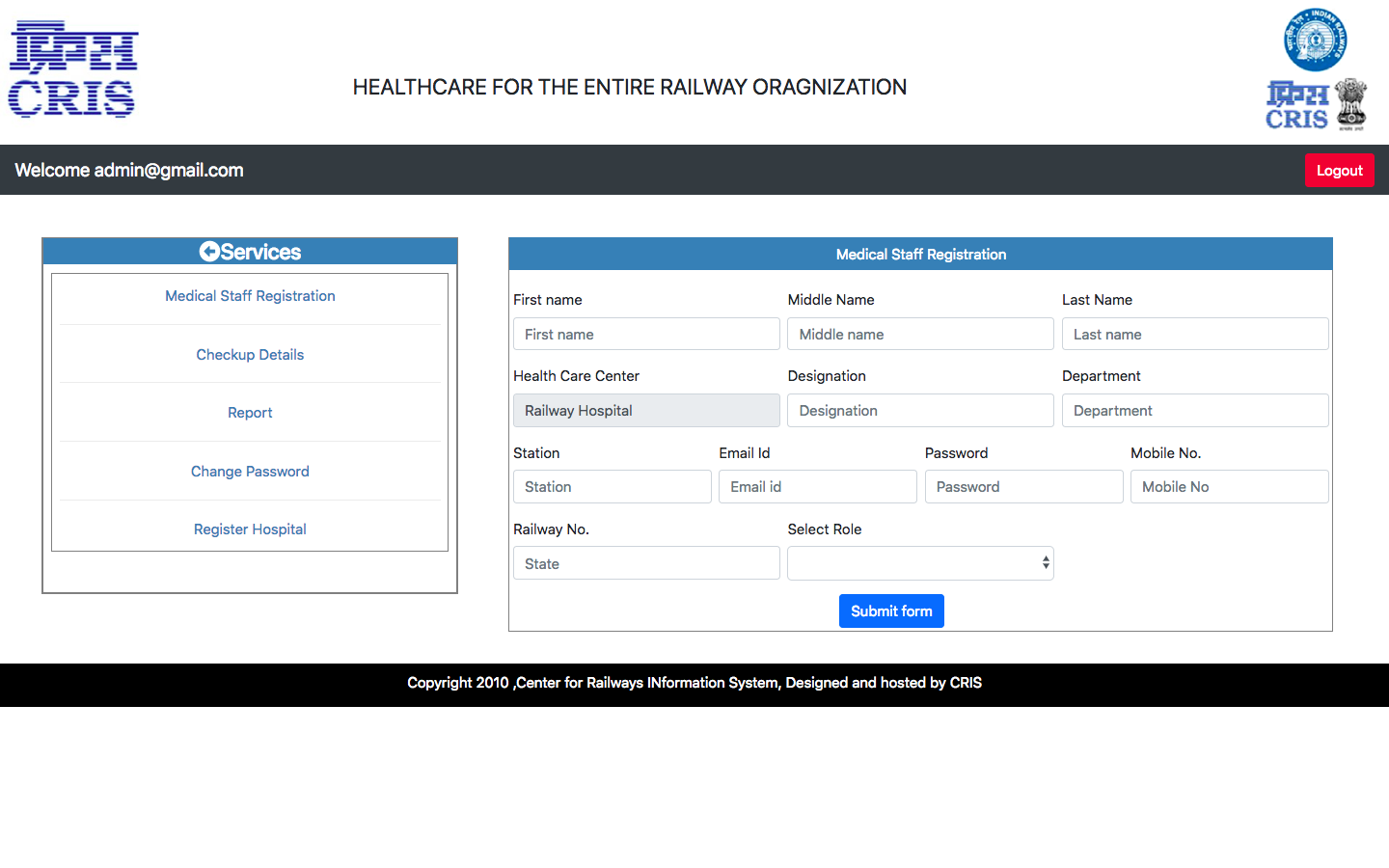
console.log(data);

},err=>{

console.log(err);

})

}



**HTML CODE**

<form style="padding: 5px" >

<div class="form-row" >

<div class="col-md-4 mb-3" >

<label for="validationDefault01">First name</label>

<input type="text" class="form-control" id="validationDefault01" placeholder="First name" required [(ngModel)]="firstname" name="firstname">

</div>

<div class="col-md-4 mb-3" >

<label for="validationDefault02">Middle Name</label>

<input type="text" class="form-control" id="validationDefault02" placeholder="Middle name" required [(ngModel)]="middlename" name="middlename">

</div>

<div class="col-md-4 mb-3" >

<label for="validationDefault02">Last Name</label>

<input type="text" class="form-control" id="validationDefault02" placeholder="Last name" required [(ngModel)]="lastname" name="lastname">

</div>

</div>

<div class="form-row">

<div class="col-md-4 mb-3">

<label for="validationDefault03">Health Care Center</label>

<input type="text" class="form-control" readonly id="validationDefault03" value="Railway Hospital" required >

</div>

<div class="col-md-4 mb-3">

<label for="validationDefault04">Designation</label>

<input type="text" class="form-control " id="validationDefault04" placeholder="Designation" required [(ngModel)]="designation" name="designation">

</div>

<div class="col-md-4 mb-3">

<label for="validationDefault04">Department</label>

<input type="text" class="form-control " id="validationDefault04" placeholder="Department" required [(ngModel)]="department" name="department">

</div>

</div>

<div class="form-row">

<div class="col-md-3 mb-3" style="width:0">

<label for="validationDefault03">Station</label>

<input type="text" class="form-control" id="validationDefault03" placeholder="Station" required [(ngModel)]="station" name="station">

</div>

<div class="col-md-3 mb-3">

<label for="validationDefault04">Email Id</label>

<input type="text" class="form-control " id="validationDefault04" placeholder="Email id" required [(ngModel)]="emailid" name="emailid">

</div>

<div class="col-md-3 mb-3">

<label for="validationDefault04">Password</label>

<input type="text" class="form-control " id="validationDefault05" placeholder="Password" required [(ngModel)]="password" name="password">

</div>

<div class="col-md-3 mb-3">

<label for="validationDefault04">Mobile No.</label>

<input type="number" class="form-control " id="validationDefault04" placeholder="Mobile No" required [(ngModel)]="mobileno" name="mobileno">

</div>

</div>

<div class="form-row">

<div class="col-lg-4 col-md-3 mb-3">

<label for="validationDefault04">Railway No.</label>

<input type="number" class="form-control " id="validationDefault04" placeholder="State" required [(ngModel)]="railwayno" name="railwayno">

</div>

<div class="col-lg-4 col-md-4">

<label for="inputState">Select Role</label>

<select id="inputState" class="form-control" [(ngModel)]="role" name="role">

<option selected>Choose...</option>

<option>Doctor</option>

<option>Nurse</option>

</select>

</div>

</div>

<button class="btn btn-primary" type="submit" style="margin-left: 40%;margin-right:20% " (click)="onClickSubmit()">Submit form</button>

</form>

**TYPESCRIPT CODE**

ngOnInit() {

if(localStorage.getItem('status')=='false'){

alert('login first !');

this.router.navigate(['/']);

}else if(localStorage.getItem('status')=='true' && localStorage.getItem('role')=='Doctor'){

alert('You have insufficient rights to access this content');

this.router.navigate(['/']);

}else if(localStorage.getItem('status')=='true' && localStorage.getItem('role')=='Nurse'){

alert('You have insufficient rights to access this content');

this.router.navigate(['/']);

}

}

onClickSubmit(){

var url="http://localhost:3000/api/Medicalstaffs";

let employee={

"department": this.department,

"designation": this.designation,

"email": this.emailid,

"firstname": this.firstname,

"healthcenter": "Railways Hospital",

"lastname": this.lastname,

"mId": "M"+Date.now(),

"middlename": this.middlename,

"mobile":this.mobileno,

"password": this.password,

"railwayno": this.railwayno,

"role": this.role,

"station": this.station

}

this.sendData.sendData(url,employee).then(data=>{

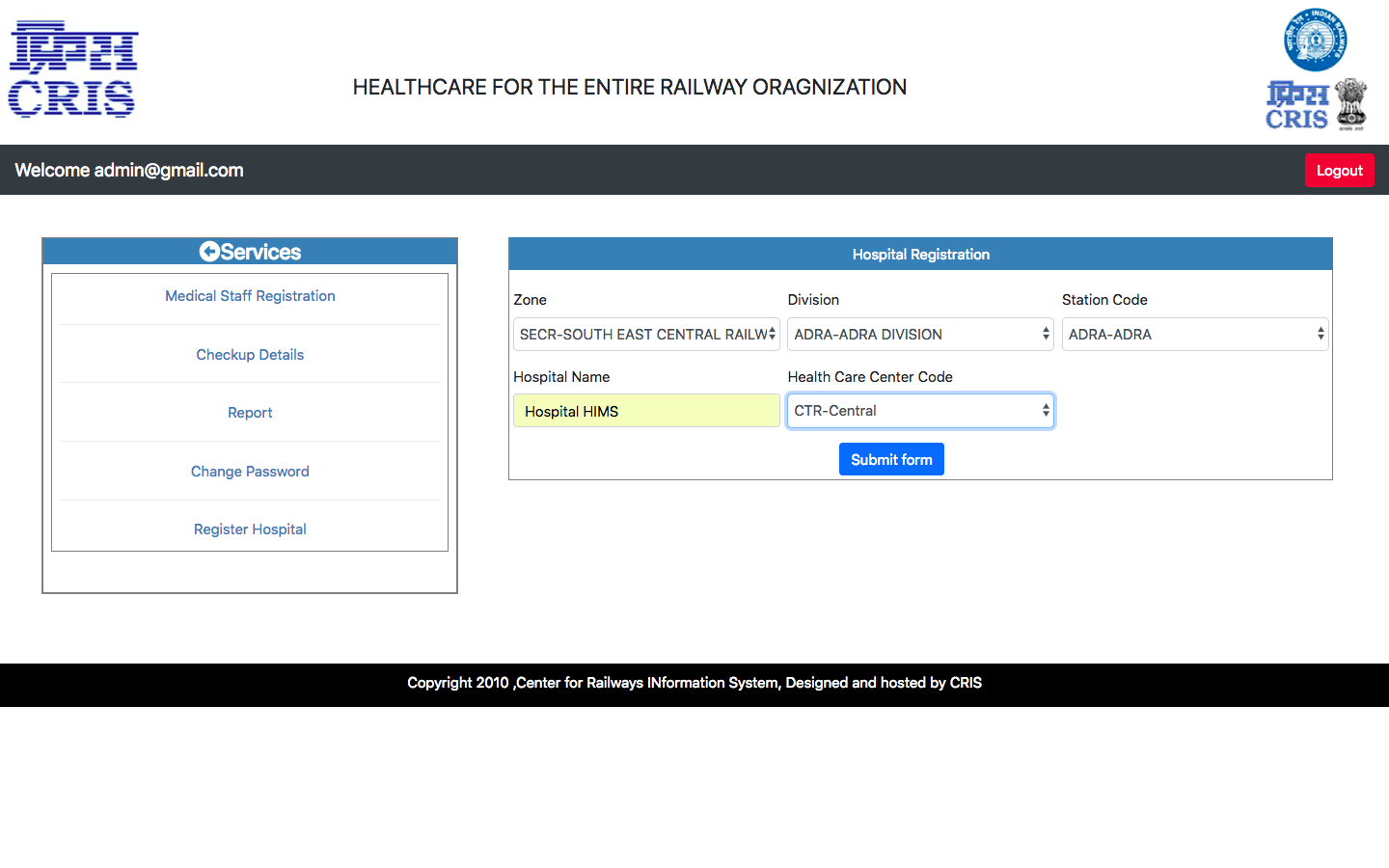
this.flash.show("Registered Successfull",{cssClass:'alert-success',timeout:3000});

},err=>{

// this.flash.show(err.message,{cssClass:'alert-danger',timeout:3000})

})

}

****

**HTML CODE**

<form style="padding: 5px" >

<div class="form-row" >

<div class="col-lg-4 col-md-4">

<label for="inputState">Zone</label>

<select name="zone" [(ngModel)]="zone" id="inputState" (change)="onChangezone($event.target.value)" class="form-control">

<option selected>--select--</option>

<option \*ngFor="let zone of zones" >{{zone.znCd}}-{{zone.znNm}}</option>

</select>

</div>

<div class="col-md-4 col-md-4" >

<label for="validationDefault02">Division</label>

<select name="division" id="inputState2" class="form-control" (change)="onChangedivision($event.target.value)" >

<option selected>--select--</option>

<option \*ngFor="let division of availableDivision">{{division.divCd}}-{{division.divNm}}</option>

</select>

<!-- <input type="text" class="form-control" id="validationDefault02" placeholder="Middle name" required> -->

</div>

<div class="col-md-4 mb-3" >

<label for="validationDefault02">Station Code</label>

<select [(ngModel)]="stationCode" name="station" class="form-control" (change)="onchangestation($event.target.value)" >

<option selected>--select--</option>

<option \*ngFor="let station of availableStation" >{{station.stationCode}}-{{station.stationName}}</option>

</select>

</div>

</div>

<div class="form-row">

<div class="col-md-4 mb-3">

<label for="validationDefault03">Hospital Name</label>

<input name="name" [(ngModel)]="name" type="text" class="form-control" id="validationDefault03" placeholder="Hospital Name" required >

</div>

<div class="col-md-4 mb-3">

<label for="validationDefault04">Health Care Center Code </label>

<select name="center" class="form-control" name="">

<option selected>--select--</option>

<option \*ngFor="let center of healthcarecode">{{center.hospital\_type}}-{{center.hospital\_desc}}</option>

</select>

</div>

</div>

<button (click)="onsubmit()" class="btn btn-primary" type="submit" style="margin-left: 40%;margin-right:20% ">Submit form</button>

</form>

**TYPESCRIPT CODE**

ngOnInit() {

if(localStorage.getItem('status')=='false'){

alert('login first !');

this.router.navigate(['/']);

}else if(localStorage.getItem('status')=='true' && localStorage.getItem('role')=='Doctor'){

alert('You have insufficient rights to access this content');

this.router.navigate(['/']);

}else if(localStorage.getItem('status')=='true' && localStorage.getItem('role')=='Nurse'){

alert('You have insufficient rights to access this content');

this.router.navigate(['/']);

}

this.uniqueid="H"+Date.now();

console.log(this.uniqueid);

this.zoneService.getAllZones().then(zones=>{

this.zones=zones;

})

this.zoneService.getAllDivisions().then(divisions=>{

this.divisions=divisions;

})

this.healthcarecode=[

{hospital\_type:"CH",hospital\_desc:"Central Hospital"},

{hospital\_type:"CTR",hospital\_desc:"Central"},

{hospital\_type:"DH",hospital\_desc:"Division Unit"},

{hospital\_type:"SD",hospital\_desc:"Sub Divisional"},

{hospital\_type:"POL",hospital\_desc:"Polycline"},

{hospital\_type:"HU",hospital\_desc:"Health Unit"},

{hospital\_type:"SUS",hospital\_desc:"Super Speciality"}

]

}

onChangezone(deviceValue) {

this.selectedZone=deviceValue;

this.availableDivision=[];

this.zoneCd=this.selectedZone.substring(0,2);

this.zoneService.getAvailableDivisions(this.selectedZone.substring(0,2)).then(availableDivision=>{

this.availableDivision=availableDivision;

})

}

onChangedivision(deviceValue){

console.log(deviceValue);

var availablecode=deviceValue;

this.availableStation=[];

this.divisionCodeSelected="";

var i:any;

var ctr:any;

ctr=0;

for(i=0;i<availablecode.length;i++){

if(availablecode.charAt(i)=='-'){

break;

}

this.divisionCodeSelected+=availablecode.charAt(i);

this.divCd=this.divisionCodeSelected;

}

this.zoneService.getAvailableStations(this.divisionCodeSelected).then(availableStation=>{

this.availableStation=availableStation;

})

}

onchangestation(value){

console.log(this.stationCode);

this.stationCd="";

var ctr:any;

ctr=0;

console.log(value);

console.log('helloo'+value);

for(var i=0;i<value.length;i++){

if(value.charAt(i)=='-'){

console.log('hello on break');

break;

}

this.stationCd+=value.charAt(i);

console.log(this.stationCd);

}

}

onsubmit(){

let hospital={

name:this.name,

center:this.center,

zoneCode:this.zoneCd,

divisionCode:this.divCd,

stationCd:this.stationCd,

hospitalid:this.uniqueid

}

this.sendData.registerHospital(hospital).then(data=>{

console.log(data);

},err=>{

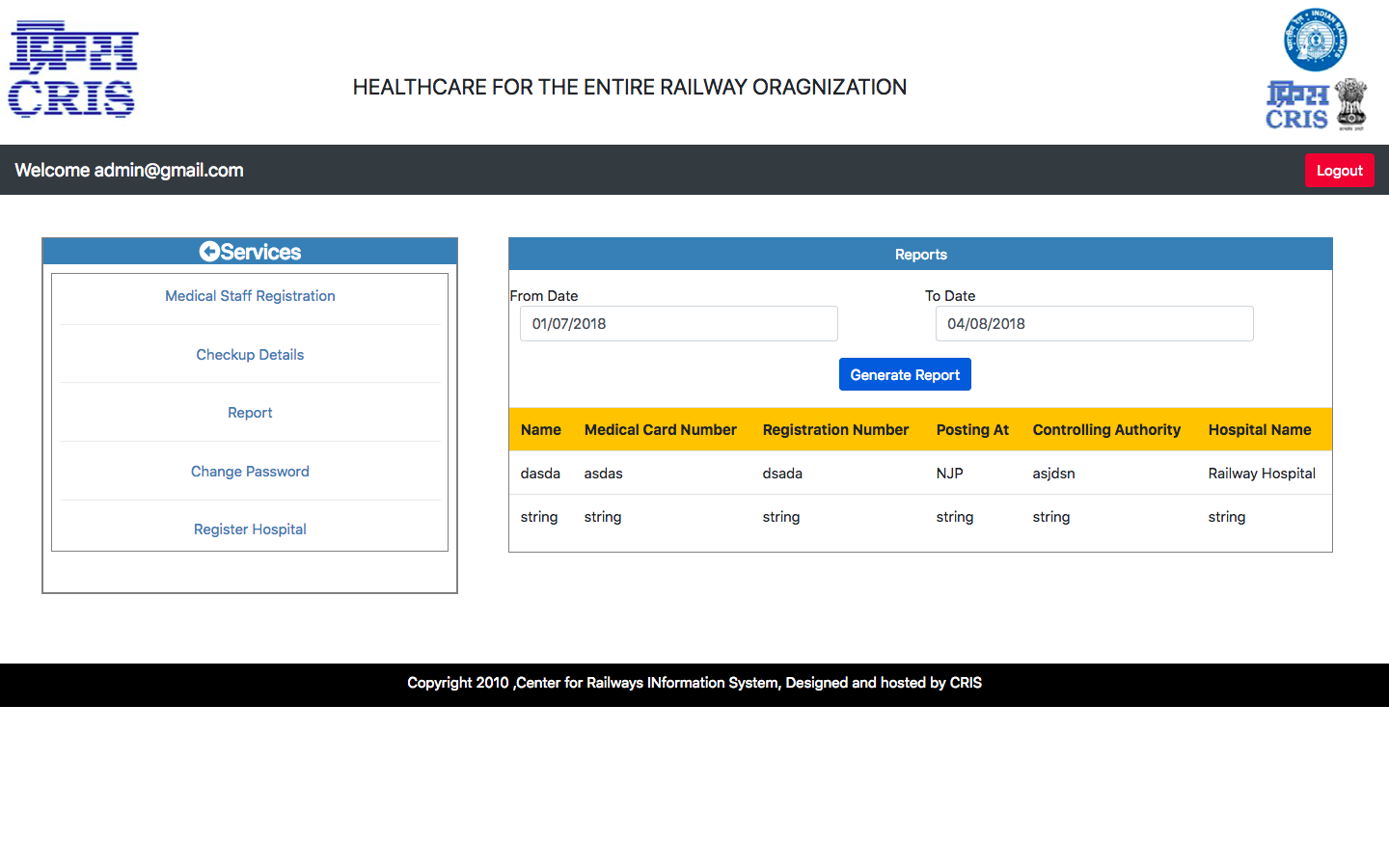
console.log(err);

})

this.flashMessage.show('registered',{cssClass:'alert-success',timeout:3000});

// this.router.navigate(['/']);

}

****

**HTML CODE**

<form >

<div class="form-row" >

<div class="col-lg-6 col-md-6">

<div class="label">From Date</div>

<div class="col-10">

<input class="form-control" type="date" id="example-date-input" [(ngModel)]="fromdate" name="fromdate">

</div>

</div>

<div class="col-lg-6 col-md-6" >

<div class="label">To Date</div>

<div class="col-10">

<input class="form-control" type="date" id="example-date-input" [(ngModel)]="todate" name="todate">

</div>

</div>

</div>

<button class="btn btn-primary" type="submit" style="margin:2%;margin-left:40% " (click)="onsubmit()" >Generate Report</button>

<table class="table">

<thead class="bg-warning">

<tr>

<th scope="col">Name</th>

<th scope="col">Medical Card Number</th>

<th scope="col">Registration Number</th>

<th scope="col">Posting At</th>

<th scope="col">Controlling Authority</th>

<th scope="col">Hospital Name</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let employee of availbleemployee">

<td>{{employee.firstname}}</td>

<td>{{employee.medicalcard}}</td>

<td>{{employee.registrationno}}</td>

<td>{{employee.station}}</td>

<td>{{employee.controllingauthority}}</td>

<td>{{employee.examinedat}}</td>

</tr>

</tbody>

</table>

</form>

**TYPESCRIPT CODE**

ngOnInit() {

if(localStorage.getItem('status')=='false'){

alert('login first !');

this.router.navigate(['/']);

}

}

onsubmit(){

var fdate=new Date(this.fromdate);

var tdate=new Date(this.todate);

var url="http://localhost:3000/api/Employeebasicinfos";

console.log(fdate);

console.log(tdate);

this.getData.getAllData(url).then(data=>{

this.employees=data;

this.ctr=0;

for(var i=0;i<this.employees.length;i++){

var date=this.employees[i].datenow;

var intdate=parseInt(date);

console.log(new Date(intdate));

if(fdate<new Date(intdate) && new Date(intdate)<tdate){

this.ctr++;

this.availbleemployee.push(this.employees[i]);

}

}

console.log(this.availbleemployee);

},err=>{

console.log(err);

})

}

**GETTING DATA SERVICE TYPESCRIPT CODE**

export class GetdataService {

constructor(private http:HttpClient) {}

getAllData(url){

return new Promise(resolve=>{

this.http.get(url).subscribe(data=>{

resolve(data);

},err=>{

console.log(err);

})

})

}

getemployee(url){

return new Promise(resolve=>{

this.http.get(url).subscribe(data=>{

resolve(data);

},err=>{

console.log(err);

})

})

}

}

**POSTING DATA SERVICE TYPESCRIPT CODE**

export class SendDataService {

authToken:any;

staff:any;

constructor(private http:Http,

private flash:FlashMessagesService) { }

registerStaff(staff){

console.log(staff);

return new Promise(resolve=>{

this.http.post('http://localhost:3000/api/Medicalstaffs',

{

"firstname": staff.name,

"middlename": null,

"lastname": null,

"healthcenter": "Railways Hospital",

"designation": staff.designation,

"department": staff.department,

"station": staff.station,

"email": staff.email,

"mobile": staff.mobile,

"railwayno": staff.railno,

"role": staff.role

}).subscribe(data=>{

resolve(data);

},err=>{

console.log(err);

})

})

}

registerHospital(hospital){

return new Promise(resolve=>{

this.http.post('http://localhost:3000/api/MedicalExamCentreMsts',

{

"hospitalName":hospital.name,

"healthCareCenterCode":hospital.center,

"zoneCd":hospital.zoneCode,

"divCd":hospital.divisionCode,

"stationCode":hospital.stationCd,

"hospitalId":hospital.hospitalid

}).subscribe(data=>{

resolve(data);

},err=>{

console.log(err);

})

})

}

sendData(url,data){

return new Promise(resolve=>{

this.http.post(url,data).subscribe(data=>{

resolve(data);

},err=>{

var error=JSON.parse(err.\_body);

console.log(error.error.message);

this.flash.show(error.error.message,{cssClass:'alert-danger',timeout:3000})

})

})

}

modifyData(url,data){

return new Promise(resolve=>{

this.http.put(url,data).subscribe(data=>{

resolve(data);

},err=>{

var error=JSON.parse(err.\_body);

console.log(error.error.message);

this.flash.show(error.error.message,{cssClass:'alert-danger',timeout:3000});

})

}) } }

**VALIDATING SERVICE TYPESCRIPT CODE**

export class ValidationServiceService {

eid:any;

constructor() { }

validateEmail(email){

const re = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)\*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

return re.test(email);

}

eidInstance(id){

this.eid=id;

}

geteidinstance(){

return this.eid;

}

}

**MySQL DATABASE SCHEMA**

-- MySQL dump 10.13 Distrib 5.7.17, for macos10.12 (x86\_64)

--

-- Host: localhost Database: hmis

-- ------------------------------------------------------

-- Server version 8.0.11

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `division\_mst`

--

DROP TABLE IF EXISTS `division\_mst`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `division\_mst` (

`DIV\_CD` varchar(255) NOT NULL,

`DIV\_NM` varchar(255) DEFAULT NULL,

`ZN\_CD` varchar(255) DEFAULT NULL,

`statusCode` int(11) DEFAULT '1',

PRIMARY KEY (`DIV\_CD`),

UNIQUE KEY `DIV\_CD` (`DIV\_CD`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `division\_mst`

--

--

-- Table structure for table `Employee`

--

DROP TABLE IF EXISTS `Employee`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `Employee` (

`emaiid` varchar(255) NOT NULL,

`password` varchar(45) NOT NULL,

PRIMARY KEY (`emaiid`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `Employee`

--

--

-- Table structure for table `EmployeeBasicInfo`

--

DROP TABLE IF EXISTS `EmployeeBasicInfo`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `EmployeeBasicInfo` (

`EID` varchar(45) NOT NULL,

`ExaminedAt` varchar(45) DEFAULT NULL,

`Station` varchar(45) DEFAULT NULL,

`ExaminedBy` varchar(45) DEFAULT NULL,

`Zone` varchar(45) DEFAULT NULL,

`Division` varchar(45) DEFAULT NULL,

`ControllingAuthority` varchar(45) NOT NULL,

`MedicalCard` varchar(45) DEFAULT NULL,

`MedicalPass` varchar(45) DEFAULT NULL,

`RegistrationNo` varchar(45) DEFAULT NULL,

`firstname` varchar(45) NOT NULL,

`middlename` varchar(45) DEFAULT NULL,

`lastname` varchar(45) DEFAULT NULL,

`designation` varchar(45) DEFAULT NULL,

`department` varchar(45) DEFAULT NULL,

`employeeservicestation` varchar(45) DEFAULT NULL,

`dataofbirth` varchar(45) DEFAULT NULL,

`examineddate` varchar(45) DEFAULT NULL,

`datenow` varchar(45) NOT NULL,

PRIMARY KEY (`EID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `EmployeeBasicInfo`

--

--

-- Table structure for table `medical\_exam\_centre\_mst`

--

DROP TABLE IF EXISTS `medical\_exam\_centre\_mst`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `medical\_exam\_centre\_mst` (

`hospital\_id` varchar(96) DEFAULT NULL,

`hospital\_name` varchar(300) DEFAULT NULL,

`health\_care\_centre\_code` char(9) DEFAULT NULL,

`station\_code` char(30) DEFAULT NULL,

`zone\_cd` varchar(75) DEFAULT NULL,

`div\_cd` varchar(75) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `medical\_exam\_centre\_mst`

--

--

-- Table structure for table `medical\_staff\_qbe`

--

DROP TABLE IF EXISTS `medical\_staff\_qbe`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `medical\_staff\_qbe` (

`EmpID` smallint(6) NOT NULL,

`Designation` varchar(10) DEFAULT NULL,

`Name` varchar(100) DEFAULT NULL,

`Speciality` varchar(10) DEFAULT NULL,

`statusCode` int(11) DEFAULT '1',

PRIMARY KEY (`EmpID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `medical\_staff\_qbe`

--

--

-- Table structure for table `medicalStaff`

--

DROP TABLE IF EXISTS `medicalStaff`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `medicalStaff` (

`firstname` char(255) NOT NULL,

`middlename` varchar(255) DEFAULT NULL,

`lastname` varchar(45) DEFAULT NULL,

`healthcenter` varchar(45) DEFAULT NULL,

`designation` varchar(45) DEFAULT NULL,

`department` varchar(45) DEFAULT NULL,

`station` varchar(45) DEFAULT NULL,

`email` varchar(45) NOT NULL,

`mobile` varchar(45) DEFAULT NULL,

`railwayno` varchar(45) DEFAULT NULL,

`role` varchar(45) DEFAULT NULL,

`m\_id` varchar(45) NOT NULL,

`password` varchar(45) NOT NULL,

PRIMARY KEY (`m\_id`),

UNIQUE KEY `email\_UNIQUE` (`email`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `medicalStaff`

--

--

-- Table structure for table `station\_mst`

--

DROP TABLE IF EXISTS `station\_mst`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `station\_mst` (

`STATION\_CODE` char(18) NOT NULL,

`STATION\_NAME` varchar(120) DEFAULT NULL,

`DIV\_CD` char(12) DEFAULT NULL,

`STATION\_STATUS` tinyint(1) DEFAULT NULL,

PRIMARY KEY (`STATION\_CODE`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `station\_mst`

--

DROP TABLE IF EXISTS `zone\_mst`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `zone\_mst` (

`ZN\_CD` char(10) NOT NULL,

`ZN\_NM` varchar(40) NOT NULL,

`statusCode` int(11) DEFAULT '1',

PRIMARY KEY (`ZN\_CD`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `zone\_mst`

--

-- Dump completed on 2018-07-16 13:32:23

**REFERENCES**

**[1]** Angular Documentation**“**Angular Tutorial”

https://angular.io/docs

**[2]** Loopback**“**Loopback Doucmentation”

https://loopback.io/doc/

**[3]** Bootstrap”Bootstrap Documentation”

https://getbootstrap.com/docs/4.1/getting-started/introduction/