**CONTENTS**

**Introduction 3**

**Survey of Technologies 4**

**Requirements and Analysis 5**

**System Design 12**

**Implementation and testing 23**

**Results and discussion 120**

**Conclusion 121**

**Reference 122**

1. **INTRODUCTION**
   1. **Background**

A bank maintains different types of safe lockers at some of its safe sites. A locker can be held jointly by two or more customers also. Bank also has a process for allocation and cancellation of a locker. This online locker management system provides facilities for allotment of lockers to new account holders (individual or joint) as well as cancellation of lockers, management of existing lockers, monitoring of locker operations and rentals, and scheduling appointments for operating the lockers. In general as the number of lockers is limited, therefore, it maintains a queue of customers who have requested for the locker service.

* 1. **Objectives**

1. To reduce the communication gap between the customer and the bank locker service as well as the manual effort and hassle faced by the bank to carry out locker management operations.
2. Centralized system to manage multiple lockers and multiple requests made by customers.
3. It makes the overall procedure faster and secure.
4. Comprehensive and convenient to use, both for the bank as well as the customer.
   1. **Purpose**

It is time-saving, as it allows multiple customers to request a locker online, thereby avoiding lengthy queues. All of the data in this system is secure and safe from unauthorized access. The process becomes more accurate and efficient.

* 1. **Scope**

Since the system has a centralized structure and workflow, it can be used by almost any bank in our country to carry out the locker management, which will make it very easy and accessible for all bank account holders in our country.

1. **SURVEY OF TECHNOLOGIES**
   1. **Hardware Used**

|  |  |
| --- | --- |
| **Processor** | **:** Intel Core i3 7th generation, 3.9GHz |
| **RAM** | **:** 4GB |
| **Hard Disk** | **:** 1TB |

* 1. **Software Used**

|  |  |
| --- | --- |
| **Front-end** | **:** HTML, CSS, JavaScript, Bootstrap |
| **Back-end** | **:** JSP, MySQL Server 5.5, Apache Tomcat Server |
| **Programming language** | **:** Java (JDK 1.8.0\_181) |
| **Text editor/IDE** | **:** Net Beans IDE 8.2 and Visual Studio Code |
| **Web browser** | **:** Google Chrome |
| **Operating System** | **:** Windows 8.1 Pro |

1. **REQUIREMENTS AND ANALYSIS**
   1. **Problem Definition**

For a better understanding, the system is divided into specific domains – Administrator, User, Locker, and the System. The admin has the privileges to allot as well as cancel the locker on the customer’s request. It keeps a record of the yearly locker rental to be deducted from the customer’s account, and it also keeps a record of the number of locker operations performed by the customer(s). It maintains a queue of the requests made by the customers (processing and pending). It can check the status of locker availability and accordingly schedule the locker operation on the stipulated date and time as requested by the customer. A record of the users and lockers is maintained and can be manipulated by the administrator of the system.

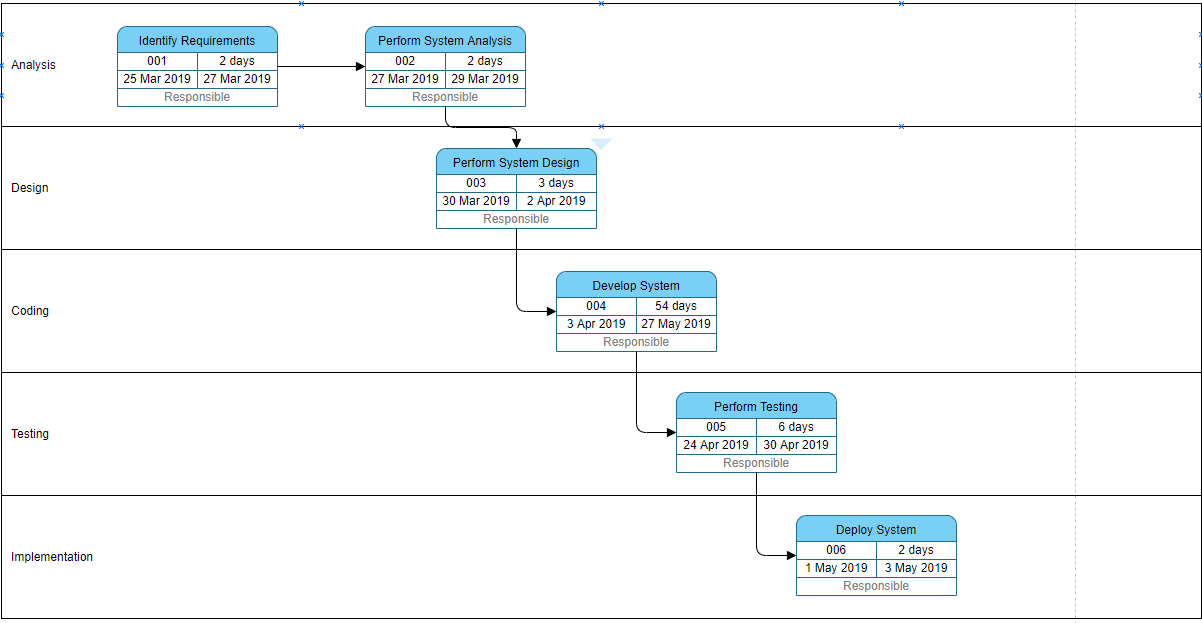
* 1. **Requirement Specifications**
     1. **Functional Requirements**

**Administrator Level:**

1. Creation or cancellation of a locker
2. Validate and verify all input values
3. Check locker availability
4. Track number of locker operations, and deduct charges if the number of operations has exceeded the limit
5. Schedule appointment for locker operation
6. Maintain queue of requests made by waiting customers
7. Notify the end user whenever required
8. Deduct yearly locker rental from the user’s account

**User Level:**

1. Login to the system
2. View dashboard
3. Request for locker creation
4. Request for locker operation
5. Request for cancellation of locker
6. Reset/Forgot password
   * 1. **Non-functional requirements**
7. It should be able to handle multiple requests at a time
8. Response time should be minimum
9. It should occupy less memory on the system
10. It should be portable, i.e. it should be able to work in any environment
11. All transactions should occur in a secure manner, and the data should be stored securely
12. Authorization and authentication is necessary
13. The user interface has to be interactive and self-explanatory
    1. **Planning and Scheduling**

****

* 1. **Software and Hardware Requirements**
     1. **Hardware**

Processor - Intel Pentium, Core i3 or above

Hard Disk - Minimum 50GB free space

RAM - 3GB or more

* + 1. **Software**

Operating System - Windows 7 and above, LINUX,

Web Browser - Microsoft Edge, Internet

Explorer 5.0 and above,

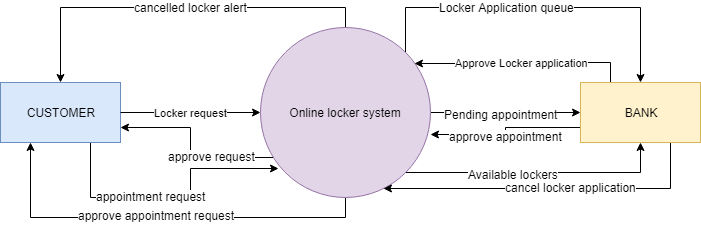
Chrome, Firefox

Or any other JavaScript-enabled

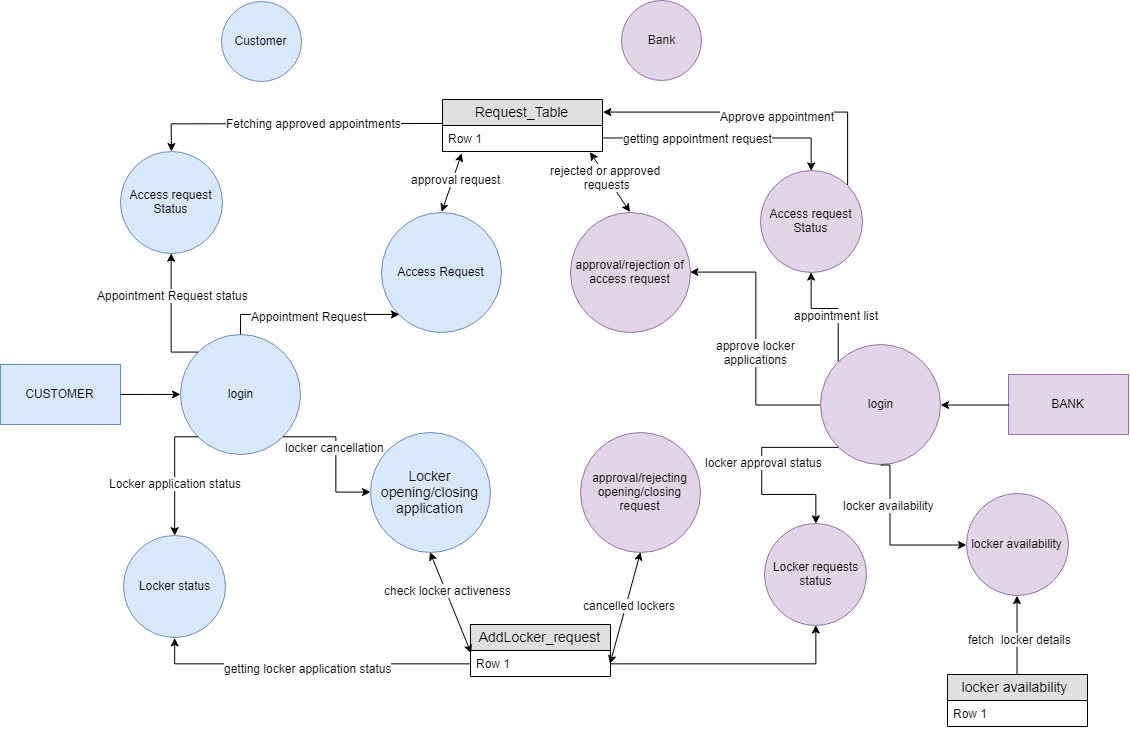
Browser

Database - MySQL Server 5.5

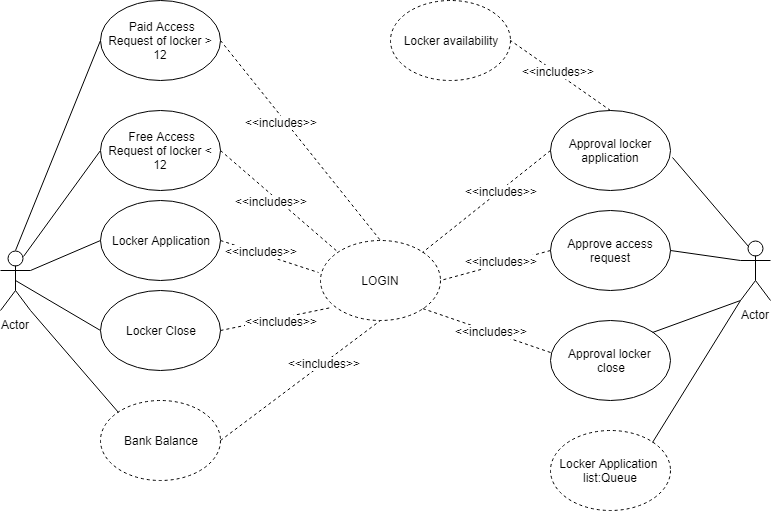
Server - Glassfish Tomcat Server

* 1. **Conceptual Models**
     1. **Data Flow Diagram**

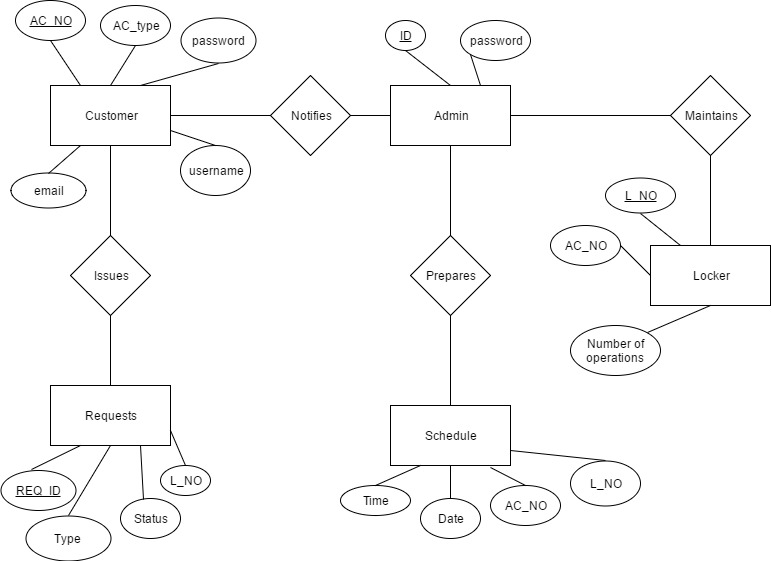
**Data Flow Diagram Level 0**

****

**Data Flow Diagram Level 1**

****

* + 1. **Use Case Diagram**
    2. **Entity Relationship Diagram**

****

1. **SYSTEM DESIGN**
   1. **Basic Modules**
      1. **Customer**

* + - 1. **Login**

This module is responsible for allowing user/customer to use the locker management systems. Login is essential segment of the project for many reasons such as validation, identity proof and preventing unauthorized accesses.

* + - 1. **Logout**

This module is responsible to clear/close all the open resources such as, database connection, network connections, etc. This module also clear the cache from client’s computer in the case of local storage attack.

* + - 1. **Locker Application**

This module works under the locker application. Customer applies for locker application which is required to be approved from bank side. This module is responsible for maintaining the status such as pending locker application request, cancellation locker application request.

* + - 1. **Locker Cancellation Notification**

This module is completely depended on bank operation towards customer locker application request. This module takes care of cancellation status, for instance, if customer request is approved by bank then in the middle of working duration bank had to cancel then it will reflect on customer screen.

* + - 1. **Balance Deduction Process**

This module bank balance deduction process only does specific task, such as checking current balance, overflow, underflow balance and deductions. Whenever locker gets renewed or appointment reaches 12 then it gets activated to deduct money from particular customer account.

* + - 1. **Locker Renewal Process**

Locker renewal process is the process to renew the account after one year or stipulated years decided by bank. Locker renewal process revokes the balance deduction module also.

* + - 1. **Locker Appointment**

Locker appointment is core section of this project and it actually handles, customer’s locker access request on specific data and time. It ultimately deals with bank directly to get the approved.

* + - 1. **Appointment History**

This module is responsible for maintaining the past appointment of particular user to monitor the free appointments and paid appointment.

* + - 1. **Bank Balance, Locker details**

Bank balance, Locker details module request for data such as current balance, locker issued and expiry date. This module displays only current balance and locker details on customer dashboard.

* + - 1. **Cross site route validation**

This module basically concern security of the application. For instance, if any unauthorized user tries to access our routes then it will redirect to login page.

It also maintains the requirement resources such as database connections.

* + - 1. **Account opening process**

This module is only responsible for opening the account with some particular opening account. After applying online, customer has to visit bank branch to complete other formalities.

* + 1. **Bank**
       1. **Login**

This module is responsible for allowing user/customer to use the locker management systems. Login is essential segment of the project for many reasons such as validation, identity proof and preventing unauthorized accesses. Bank login differs usual login because it is authenticate employee id with bank and its association.

* + - 1. **Logout**

This module is responsible to clear/close all the open resources such as, database connection, network connections, etc. This module also clear the cache from client’s computer in the case of local storage attack.

* + - 1. **Locker Application Queue**

This module is responsible for displaying and maintaining the queue of customer’s request for new locker. This is module is also responsible to refresh updated request if any new requests arrives in the room. It also refresh on tab for updated data.

* + - 1. **Locker Application Cancellations**

Locker application cancellations cancel the approved request. It deals with only those applications which are approved already, if bank finds any suspicious activities against particular customer then bank has authority to cancel any customer’s locker applications.

* + - 1. **Locker Availability**

This module is responsible for maintaining the locker availabilities. It also provides cancel button to cancel according to bank authority. It will reflect on customer dashboard.

* + - 1. **Locker Appointment Approvals**

Locker appointment approvals are responsible for approvals and providing confirmation of the locker account. Approvals button are available in the bank dashboard. Lockers approvals are available on certain rules which will be conducted by bank only.

* + - 1. **Locker Appointment Rejections**

This module ‘Locker appointment rejections’ rejects the request appointments for specific date and time for accessing the locker. Rejection alert will be displayed on the customer dashboard.

* + - 1. **Cross site route validation**

This module basically concern security of the application. For instance, if any unauthorized user tries to access our routes then it will redirect to login page. It also maintains the requirement resources such as database connections.

* + - 1. **Account Information**

It is a simple active account dashboard which displays available account bank currently has. The list of active account contains customer’s following information, such as name, account number, current balance, etc.

* 1. **Data Design**
     1. **Schema Design**

Usually, a collection of interrelated data is referred to as database. The database contains information about one particular enterprise. The management of data involves both definitions of structures of the stage of information. In addition, the database system must

1. **users**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Account\_number | int(30) |  | NO | PRI | *NULL* | auto\_increment |
| username | varchar(255) |  | NO | UNI | *NULL* |  |
| password | varchar(255) |  | NO |  | *NULL* |  |
| nature | varchar(255) |  | NO |  | *NULL* |  |
| balance | int(11) |  | NO |  | *NULL* |  |
| name | varchar(255) |  | NO |  | *NULL* |  |
| contact | varchar(18) |  | NO |  | *NULL* |  |
| address | text |  | NO |  | *NULL* |  |
| accounttype | varchar(255) |  | NO |  | *NULL* |  |
| is\_active | int(11) |  | NO |  | *NULL* |  |

1. **Locker**

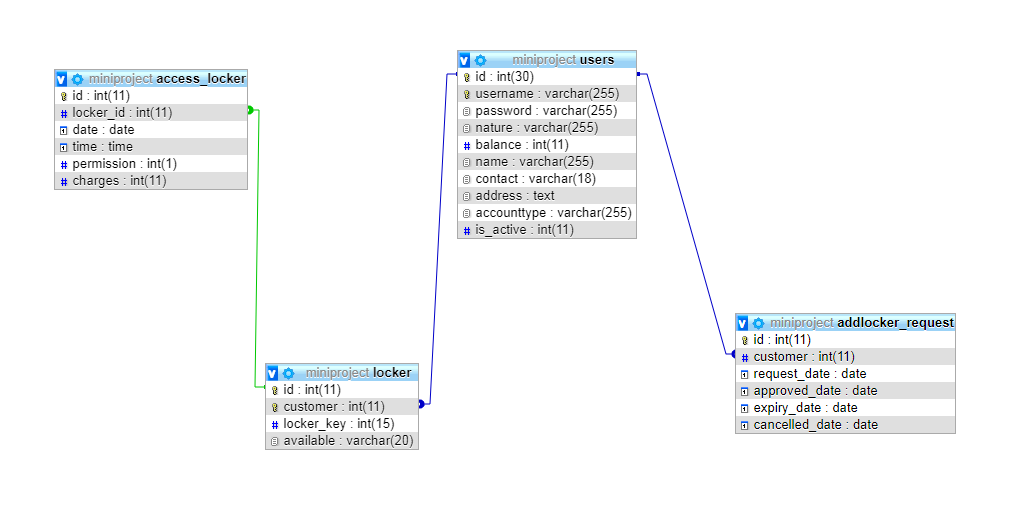
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | int(11) | NO | PRI | *NULL* | auto\_increment |
| customer | int(11) | YES | UNI | *NULL* |  |
| locker\_key | int(15) | NO |  | *NULL* |  |
| available | varchar(20) | NO |  | yes |  |

1. **AddLocker\_Request**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | int(11) | NO | PRI | *NULL* | auto\_increment |
| customer | int(11) | NO | MUL | *NULL* |  |
| request\_date | date | YES |  | *NULL* |  |
| approved\_date | date | YES |  | *NULL* |  |
| expiry\_date | date | YES |  | *NULL* |  |
| cancelled\_date | date | YES |  | *NULL* |  |

1. **Appointment**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | int(11) | NO | PRI | *NULL* | auto\_increment |  |
| locker\_id | int(11) | NO | MUL | *NULL* |  |  |
| date | date | NO |  | *NULL* |  |  |
| time | time | NO |  | *NULL* |  |  |
| permission | int(1) | NO |  | *NULL* |  |  |
| charges | int(11) | NO |  | *NULL* |  |  |

* + 1. **Data Integrity and Constraints**
  1. **Procedural Design** 
     1. **Logic Diagrams**

Logical System Design involves developing general specification for how the basic information system activities of input, processing, output, storage and control can meet end user requirements.

* + 1. **Data Structures**

Data structures played very crucial role in this project. I have implemented and used following data structure such as hash table, array list and tree. Different type of data structure played different roles such hash table is used to parse json data and improving searching techniques. Arrays are used to travel data between server to server/ process to process.

* 1. **User Interface Design**

The first step in System Design is the User Interface Design. The User Interface Design activity focuses on the preparation of input and design of output reports in a form acceptable to the users.

User Interface Design consists of two steps – Input Design and Output design.

• **Input Design:** Input Design is the process of converting user originated inputs to a computer based format. The goal is a make the data entry as easy, logical and free from errors as possible.

• **Output Design:** Computer output is the most important and direct source of information to the user. Efficient, intelligible output design helps the user in decision making. Provide for safely information storage in the database, system crash or unauthorized access

* 1. **Test Cases Design**

**Appointment form:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Date** | **Time** | **Expected Output** |
| 1 | [BLANK] | [BLANK] | Do not leave blank any field |
| 2 | Date | [BLANK] | Do not leave blank any field |
| 3 | [BLANK] | Time | Do not leave blank any field |
| 4 | Date | Time | SUCCESS |

**Login form**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. no** | **User ID** | **Password** | **Expected Output** |
| **1.** | **[Blank]** | **[Blank]** | **[Label] - Enter your credentials** |
| **2.** | **[Blank]** | **Sample\_Password** | **[Label] - Enter your credentials** |
| **3.** | **Sample\_Username** | **[Blank]** | **[Label] - Enter your credentials** |
| **4.** | **[Incorrect username]** | **[Incorrect password]** | **[Label] – Invalid credentials** |
| **5.** | **[Incorrect username]** | **[correct password]** | **[Label] – Invalid credentials** |
| **6.** | **[correct username]** | **[Incorrect password]** | **[Label] – Invalid credentials** |
| **7.** | **[correct username]** | **[correct password]** | **Logged in** |

**Account Opening Form :**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S. No** | **Contact Number** | **Residential Address** | **Account Type** | **Opening Amount** | **Password** | **Expected Output** |
| 1 | [BLANK] | [BLANK] | [BLANK] | [BLANK] | [BLANK] | Do not leave blank any field |
| 2 | ABC | sample\_address | Saving | 15000 | IGNOU123 | Invalid values |
| 3 | 9877824140 | [BLANK] | Saving | 15000 | IGNOU123 | Do not leave blank any field |
| 4 | 9877824140 | sample\_address | [BLANK] | 15000 | IGNOU123 | Do not leave blank any field |
| 5 | 9877824140 | sample\_address | Current | ABC | IGNOU123 | Invalid values |
| 6 | 9877824140 | sample\_address | Current | 1500 | [BLANK] | Do not leave blank any field |
| 7 | 9877824140 | sample\_address | Current | 15000 | IGNOU123 | SUCCESS |

1. **IMPLEMENTATION AND TESTING**
   1. **Implementation Approaches**

Implementation actually varies according to project requirement and size of project functionality. In my project, I applied/implemented three tier architecture:

Three-tier (layer) is a client-server architecture in which the user interface, business process (business rules) and data storage and data access are developed and maintained as independent modules or most often on separate platforms

• Presentation tier –HTML & Javascript Web forms,

• Middle tier – JAVA & library such as JERSEY, etc.

• Data tier- MYSQL Database

**Flexibility:**

• Management of data is independent from the physical storage support,

• Maintenance of the business logic is easier,

• Migration to new graphical environments is faster.

• If there is a minor change in the business logic, we don’t have to install the entire system in individual user’s PCs.

**Reusability:**

Reusability of business logic is greater for the presentation layer. As this component is developed and tested, we can use it in any other project and would be helpful for future use.

* 1. **Coding Details and Code Efficiency**
     1. **Code Efficiency**
        1. **Frontend code :**

App.modul.ts :

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { HttpClientModule } from '@angular/common/http';

import { LoginComponent } from './login/login.component';

import { RouterModule, Routes } from '@angular/router';

import { FormsModule } from '@angular/forms';

import { DashboardComponent } from './dashboard/dashboard.component';

import { MainComponent } from './main/main.component';

import { BankloginComponent } from './banklogin/banklogin.component';

import { BankdashboardComponent } from './bankdashboard/bankdashboard.component';

import { OpenAccountComponent } from './open-account/open-account.component';

const appRoutes : Routes = [

{

path : '', component : LoginComponent

},

{

path : '\dashboard', component : DashboardComponent

},

{

path : '\main', component : MainComponent

},

{

path : 'bank', component : BankloginComponent

},

{

path : 'bankdashboard', component : BankdashboardComponent

},

{

path : 'account', component : OpenAccountComponent

}

];

@NgModule({

declarations: [

AppComponent,

LoginComponent,

DashboardComponent,

MainComponent,

BankloginComponent,

BankdashboardComponent,

OpenAccountComponent

],

imports: [

BrowserModule,

AppRoutingModule,

HttpClientModule,

FormsModule,

RouterModule.forRoot(appRoutes)

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**App.component.html :**

<div id="divLoading" style="margin: 0px; padding: 0px; position: fixed; right: 0px; top: 0px; width: 100%; height: 100%; background-color: rgb(15, 15, 17); z-index: 30001; opacity: 0.3;display: none;">

<p style="position: absolute; color: rgb(19, 18, 18); font-weight: bold; top: 45%; left: 45%;">

Loading, please wait...

<img src="https://timesheet.teri.res.in/images/ajax-loading.gif">

</p>

</div>

<div style="text-align:center">

<h2>Online Locker Management System</h2>

</div>

<div class="container-fluid">

<router-outlet></router-outlet>

</div>

**App.component.ts :**

import { Component } from '@angular/core';

import { HttpClient } from '@angular/common/http';

declare var $: any;

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'mini-project';

kajal\_data : any;

constructor(private http: HttpClient) { }

ngOnInit(){

this.http.get('http://localhost:8080/api/miniproject/AuthorService/test')

.subscribe((data)=>{

console.log(data);

this.kajal\_data = data;

});

}

public static onShowLoader(loader){

if(loader == 1){

$('#divLoading').css("display", "block");

}else{

$('#divLoading').css("display", "none");

}

}

}

Login.component.css:

/\* BASIC \*/

html {

background-color: #56baed;

}

body {

font-family: "Poppins", sans-serif;

height: 100vh;

}

a {

color: #92badd;

display:inline-block;

text-decoration: none;

font-weight: 400;

}

h2 {

text-align: center;

font-size: 16px;

font-weight: 600;

text-transform: uppercase;

display:inline-block;

margin: 40px 8px 10px 8px;

color: #cccccc;

}

/\* STRUCTURE \*/

.wrapper {

display: flex;

align-items: center;

flex-direction: column;

justify-content: center;

width: 100%;

min-height: 100%;

padding: 20px;

}

#formContent {

-webkit-border-radius: 10px 10px 10px 10px;

border-radius: 10px 10px 10px 10px;

background: #fff;

padding: 30px;

width: 90%;

max-width: 450px;

position: relative;

padding: 0px;

-webkit-box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);

box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);

text-align: center;

}

#formFooter {

background-color: #f6f6f6;

border-top: 1px solid #dce8f1;

padding: 25px;

text-align: center;

-webkit-border-radius: 0 0 10px 10px;

border-radius: 0 0 10px 10px;

}

/\* TABS \*/

h2.inactive {

color: #cccccc;

}

h2.active {

color: #0d0d0d;

border-bottom: 2px solid #5fbae9;

}

/\* FORM TYPOGRAPHY\*/

input[type=button], input[type=submit], input[type=reset] {

background-color: #56baed;

border: none;

color: white;

padding: 15px 80px;

text-align: center;

text-decoration: none;

display: inline-block;

text-transform: uppercase;

font-size: 13px;

-webkit-box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);

box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);

-webkit-border-radius: 5px 5px 5px 5px;

border-radius: 5px 5px 5px 5px;

margin: 5px 20px 40px 20px;

-webkit-transition: all 0.3s ease-in-out;

-moz-transition: all 0.3s ease-in-out;

-ms-transition: all 0.3s ease-in-out;

-o-transition: all 0.3s ease-in-out;

transition: all 0.3s ease-in-out;

}

input[type=button]:hover, input[type=submit]:hover, input[type=reset]:hover {

background-color: #39ace7;

}

input[type=button]:active, input[type=submit]:active, input[type=reset]:active {

-moz-transform: scale(0.95);

-webkit-transform: scale(0.95);

-o-transform: scale(0.95);

-ms-transform: scale(0.95);

transform: scale(0.95);

}

input[type=text] {

background-color: #f6f6f6;

border: none;

color: #0d0d0d;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 5px;

width: 85%;

border: 2px solid #f6f6f6;

-webkit-transition: all 0.5s ease-in-out;

-moz-transition: all 0.5s ease-in-out;

-ms-transition: all 0.5s ease-in-out;

-o-transition: all 0.5s ease-in-out;

transition: all 0.5s ease-in-out;

-webkit-border-radius: 5px 5px 5px 5px;

border-radius: 5px 5px 5px 5px;

}

input[type=text]:focus {

background-color: #fff;

border-bottom: 2px solid #5fbae9;

}

input[type=text]:placeholder {

color: #cccccc;

}

input[type=password] {

background-color: #f6f6f6;

border: none;

color: #0d0d0d;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 5px;

width: 85%;

border: 2px solid #f6f6f6;

-webkit-transition: all 0.5s ease-in-out;

-moz-transition: all 0.5s ease-in-out;

-ms-transition: all 0.5s ease-in-out;

-o-transition: all 0.5s ease-in-out;

transition: all 0.5s ease-in-out;

-webkit-border-radius: 5px 5px 5px 5px;

border-radius: 5px 5px 5px 5px;

}

input[type=password]:focus {

background-color: #fff;

border-bottom: 2px solid #5fbae9;

}

input[type=password]:placeholder {

color: #cccccc;

}

/\* ANIMATIONS \*/

/\* Simple CSS3 Fade-in-down Animation \*/

.fadeInDown {

-webkit-animation-name: fadeInDown;

animation-name: fadeInDown;

-webkit-animation-duration: 1s;

animation-duration: 1s;

-webkit-animation-fill-mode: both;

animation-fill-mode: both;

}

@-webkit-keyframes fadeInDown {

0% {

opacity: 0;

-webkit-transform: translate3d(0, -100%, 0);

transform: translate3d(0, -100%, 0);

}

100% {

opacity: 1;

-webkit-transform: none;

transform: none;

}

}

@keyframes fadeInDown {

0% {

opacity: 0;

-webkit-transform: translate3d(0, -100%, 0);

transform: translate3d(0, -100%, 0);

}

100% {

opacity: 1;

-webkit-transform: none;

transform: none;

}

}

/\* Simple CSS3 Fade-in Animation \*/

@-webkit-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

@-moz-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

@keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

.fadeIn {

opacity:0;

-webkit-animation:fadeIn ease-in 1;

-moz-animation:fadeIn ease-in 1;

animation:fadeIn ease-in 1;

-webkit-animation-fill-mode:forwards;

-moz-animation-fill-mode:forwards;

animation-fill-mode:forwards;

-webkit-animation-duration:1s;

-moz-animation-duration:1s;

animation-duration:1s;

}

.fadeIn.first {

-webkit-animation-delay: 0.4s;

-moz-animation-delay: 0.4s;

animation-delay: 0.4s;

}

.fadeIn.second {

-webkit-animation-delay: 0.6s;

-moz-animation-delay: 0.6s;

animation-delay: 0.6s;

}

.fadeIn.third {

-webkit-animation-delay: 0.8s;

-moz-animation-delay: 0.8s;

animation-delay: 0.8s;

}

.fadeIn.fourth {

-webkit-animation-delay: 1s;

-moz-animation-delay: 1s;

animation-delay: 1s;

}

/\* Simple CSS3 Fade-in Animation \*/

.underlineHover:after {

display: block;

left: 0;

bottom: -10px;

width: 0;

height: 2px;

background-color: #56baed;

content: "";

transition: width 0.2s;

}

.underlineHover:hover {

color: #0d0d0d;

}

.underlineHover:hover:after{

width: 100%;

}

/\* OTHERS \*/

\*:focus {

outline: none;

}

#icon {

width:60%;

}

login.component.html :

<div class="wrapper fadeInDown">

<div id="formContent">

<!-- Tabs Titles -->

<!-- Icon -->

<div class="fadeIn first">

<h4>Customer Login</h4>

<!-- <img src="https://www.consumer-voice.org/wp-content/uploads/2017/09/bank-locker-insurance.jpg" id="icon" alt="User Icon" /> -->

</div>

<!-- Login Form -->

<form (ngSubmit)="onLogin()" #login="ngForm" name="loginForm" id="loginForm" method="post" role="form">

<div ngModelGroup="loginFormData" #loginFormData = "ngModelGroup">

<input type="text" id="login" class="fadeIn second" name="username" placeholder="login" ngModel required>

<input type="password" id="password" class="fadeIn third" name="password" placeholder="password" ngModel required>

<input type="submit" class="fadeIn fourth" value="Log In">

</div>

</form>

<button class="btn btn-link" (click)="createAccount()">Don't have an account? Create one</button>

<!-- Remind Passowrd -->

<!-- <div id="formFooter">

<a class="underlineHover" href="#">Forgot Password?</a>

</div> -->

</div>

</div>

Login.component.ts :

import { Component, OnInit, ViewChild } from '@angular/core';

import { NgForm } from '@angular/forms';

import { HttpClient } from '@angular/common/http';

import { Router } from '@angular/router';

import { AppComponent } from '../app.component';

@Component({

selector: 'app-login',

templateUrl: './login.component.html',

styleUrls: ['./login.component.css']

})

export class LoginComponent implements OnInit {

login\_d = {

"username":'',

"password":''

};

user : any = {

"username" : "",

"password" : ""

}

@ViewChild('login') loginData : NgForm;

constructor(private http: HttpClient, private route:Router) { }

ngOnInit() {

this.user.username = localStorage.getItem("user");

this.user.password = localStorage.getItem("key");

var role = localStorage.getItem("banker");

if(this.user.username==null){

this.route.navigate(['']);

}else{

if(role == 'banker'){

this.route.navigate(['/bankdashboard']);

}else{

console.log("not a banker but already logged in ");

this.route.navigate(['dashboard']);

}

}

}

onLogin(){

console.log("login hit");

let loginData=this.loginData.value.loginFormData;

console.log(loginData);

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/login/auth",loginData)

.subscribe((data)=>{

AppComponent.onShowLoader(0);

console.log(data);

if(data['auth']==true){

localStorage.setItem("user",loginData.username);

localStorage.setItem("key",loginData.password);

this.route.navigate(['\dashboard']);

}else{

console.log("Invalid credentials");

}

},

error=>{

console.log(error['error']['message']);

});

}

createAccount(){

this.route.navigate(['account']);

}

}

Main.component.css:

.information{

background:lightgreen;

}

Main.component.html :

<script>

</script>

<nav class="navbar navbar-default">

<div class="container-fluid">

<div class="navbar-header">

<a class="navbar-brand" href="dashboard">Welcome {{user.username}}</a>

</div>

<!-- <ul class="nav navbar-nav">

<li class="active"><a href="#">Home</a></li>

<li><a href="#">Page 1</a></li>

<li><a href="#">Page 2</a></li>

</ul> -->

<ul class="nav navbar-nav navbar-right">

<!-- <li><a href="#"><span class="glyphicon glyphicon-user"></span> Sign Up</a></li> -->

<li><a href="#" (click)="logout()"><span class="glyphicon glyphicon-log-in"></span> Logout</a></li>

</ul>

</div>

</nav>

<div class="row information">

<div class="col-xs-12 col-sm-12 col-md-12 col-lg-12 text-center">

<label>locker ID - {{locker\_id}}</label><br>

<label>Locker Issue Date : {{approved\_date}}</label><br>

<label>Locker Expiry Date : {{expiry\_date}}</label><br>

<label>Bank Balance : {{balance}} rupees</label><br>

(Locker charges will be deducted on {{expiry\_date}})

</div>

</div>

<br>

<div class="row">

<div class="col-xs-12 col-sm-12 col-md-12 col-lg-12 text-center">

<table class="table table-hover">

<tr>

<th class="text-center" style="font-size:25px">

Locker access logs:

</th>

</tr>

<tr [ngStyle]="{'display':hide\_blank}">

<!-- <td> -->

No records

<!-- </td> -->

</tr>

<tr \*ngFor="let data of records; let i=index;">

<td>

{{i+1}}. You had accessed your locker on {{data.date}}. Your access id was {{data.access\_id+2019000}}. Access charge is {{data.charges}}

</td>

</tr>

</table>

<label [ngStyle]="{'display': lockeraccesspending\_message}">Your locker access appointment is pending.<br> Your access id is {{access\_id+2019000}}</label>

<form (ngSubmit)="accessrequest()" class="form-inline" #request="ngForm" name="request" id="request" method="post" role="form" [ngStyle] = "{'display' : formHide}">

<div ngModelGroup="requestData" #requestData="ngModelGroup">

<h3><label>Locker access appointment </label></h3>

<div class="form-group">

<label>Date : &nbsp;</label>

<input id="date" name="date" type="date" class="form-control" ngModel required>&nbsp;

</div>

<div class="form-group">

<label>Time : &nbsp;</label>

<input id="time" name="time" type="time" class="form-control" ngModel required>&nbsp;

</div>

<!-- <div class="form-group"> -->

<span style="color:red">{{warning}}</span>

<!-- </div> -->

<div class="form-group">

<input id="submit" name="submit" type="submit" value="Request" class="form-control">

</div>

</div>

</form>

</div>

</div>

Main.component.ts :

import { Component, OnInit, ViewChild } from '@angular/core';

import { Router } from '@angular/router';

import { HttpClient } from '@angular/common/http';

import { AppComponent } from '../app.component';

import { NgForm } from '@angular/forms';

import { DatePipe } from '@angular/common';

declare var $: any;

@Component({

selector: 'app-main',

templateUrl: './main.component.html',

styleUrls: ['./main.component.css']

})

export class MainComponent implements OnInit {

user : any = {

"username" : "",

"password" : ""

}

request\_data = {

"Date" : "",

"Time" : "",

"username" : "",

"password" : ""

}

approved\_date = "2019/03/14";

expiry\_date = "2020/03/14";

balance = 150;

serverError: string;

warning: string;

formHide: string;

lockeraccesspending\_message: string = "none";

access\_id : any;

records : any;

hide\_blank : string;

locker\_id: any;

free : any;

constructor(private route:Router, private http:HttpClient) { }

@ViewChild('request') requestdata : NgForm;

ngOnInit() {

this.user.username = localStorage.getItem("user");

this.user.password = localStorage.getItem("key");

if(this.user.username==null){

this.route.navigate(['']);

}else{

}

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/customer2bank/check\_locker\_request",this.user)

.subscribe(response=>{

console.log(response);

AppComponent.onShowLoader(0);

console.log(response['status']);

if(response['status'] == 4){

this.locker\_id = response['locker\_id'];

this.balance = response['balance'];

this.approved\_date = response['issued\_on'];

this.expiry\_date = response['expired\_on'];

// this.route.navigate(['\main']);

}else if (response['status'] == 3){

history.back();

}

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

this.serverError = "Server Unavailable";

});

//check the locker status and show old records of locker access request(if any)

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/customer2bank/approvedlockerrequests",this.user)

.subscribe(response=>{

console.log(response);

if(response['approvals']!=""){

this.records = response['approvals'];

this.hide\_blank = "none";

}else{

this.hide\_blank = "block";

}

AppComponent.onShowLoader(0);

},

error=>{

AppComponent.onShowLoader(0);

});

//check the locker status and show the pending message (if any request is pending) and hide locker acceess request form

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/customer2bank/checkpendinglockeraccessrequest",this.user)

.subscribe(

response=>{

AppComponent.onShowLoader(0);

console.log(response);

//hide the form and show message of locker access apointment if LockerAccessId is not a 0

if(response['LockerAccessId']!=0){

this.access\_id = response['LockerAccessId'];

this.formHide = "none";

this.lockeraccesspending\_message ="block";

}else{

if(response['is\_rejected']==1){

alert("Your last locker access request is rejected");

this.formHide = "block";

this.lockeraccesspending\_message ="none";

}else{

this.formHide = "block";

this.lockeraccesspending\_message ="none";

}

}

},

error=>{

AppComponent.onShowLoader(0);

});

//fetch bank balance and locker issued and expiry date

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/customer2bank/balance",this.user)

.subscribe(response=>{

AppComponent.onShowLoader(0);

// console.log(response);

this.balance = response['balance'];

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

this.serverError = "Server Unavailable";

});

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/customer2bank/locker\_issued\_expiry\_date",this.user)

.subscribe(response=>{

AppComponent.onShowLoader(0);

// console.log(response);

this.approved\_date = response['issued'];

this.expiry\_date = response['expiry'];

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

this.serverError = "Server Unavailable";

});

/\*

compare and expiry date with current date if current date is exceeds the expiry then call api to deduct money from user account and refresh current page.

\*/

} //end ngInit

logout(){

localStorage.clear();

// history.go(-2);

this.route.navigate(['']);

}

// locker appointment

accessrequest(){

console.log(this.requestdata.value.requestData);

this.request\_data.Date = this.requestdata.value.requestData.date;

this.request\_data.Time = this.requestdata.value.requestData.time;

this.request\_data.username = this.user.username;

this.request\_data.password = this.user.password;

this.warning = "";

var datePipe = new DatePipe("en-US");

var currentDate = datePipe.transform(new Date(),'dd/MM/yyyy');

var givenDate = datePipe.transform(this.request\_data.Date, "dd/MM/yyyy");

// console.log(val2>val);

if(this.request\_data.Date == "" || this.request\_data.Time == ""){

this.warning = "Schedule is invalid";

}else if (currentDate > givenDate){

this.warning = "Schedule is underflowed";

}else{

this.warning = "";

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/customer2bank/lockeraccessrequest",this.request\_data)

.subscribe(

response=>{

AppComponent.onShowLoader(0);

console.log(response);

//hide the form and show message of locker access apointment

this.formHide = "none";

this.lockeraccesspending\_message ="block";

},

error=>{

AppComponent.onShowLoader(0);

});

}

history.go(0);

}

}

Open-account.component.html:

<div class="container">

<div class="row">

<h3 class="text-center">Account Opening Form</h3>

<hr>

<div class="col-xs-12 col-sm-12 col-md-12 col-lg-12">

<!-- Login Form -->

<form (ngSubmit)="onSubmit()" #account\_open\_details="ngForm" name="account\_open\_details" id="account\_open\_details" method="post" role="form" class="">

<div ngModelGroup="account\_open\_data" #loginFormData = "ngModelGroup" class="form-group">

<label for="account\_type">Customer Name : </label>

<input type="text" id="name" class="form-control" name="name" placeholder="Name" ngModel required>

<label for="account\_type">Contact Number</label>

<input type="text" id="contact" class="form-control" name="contact" placeholder="Contact" ngModel required>

<label for="account\_type">Residential Address</label>

<!-- <input type="textarea" rows="5" id="address" class="form-control" > -->

<textarea class="form-control" rows="3" id="address" name="address" placeholder="Address" ngModel required></textarea>

<label for="account\_type">Account Type:</label>

<select class="form-control" id="accounttype" name="accounttype" ngModel required>

<option>Saving</option>

<option>Current</option>

</select>

<label for="account\_type">Opening Amount :</label>

<input type="number" id="openingamount" class="form-control" name="openingamount" placeholder="Opening Amount" ngModel required>

<label for="account\_type">Password</label>

<input type="password" id="password" class="form-control" name="password" placeholder="password" ngModel required>

<span style="color:red">{{warning}}</span>

<br>

<input style="width:200px" type="submit" class="btn btn-success" value="Submit">

</div>

</form>

</div>

</div>

</div>

Open-account.component.ts:

import { Component, OnInit, ViewChild } from '@angular/core';

import { AppComponent } from '../app.component';

import { HttpClient } from '@angular/common/http';

import { Router } from '@angular/router';

import { NgForm } from '@angular/forms';

@Component({

selector: 'app-open-account',

templateUrl: './open-account.component.html',

styleUrls: ['./open-account.component.css']

})

export class OpenAccountComponent implements OnInit {

@ViewChild('account\_open\_details') account\_open\_details : NgForm;

warning: string;

constructor(private http:HttpClient, private route:Router) { }

ngOnInit() {

}

onSubmit(){

// let account\_open\_details;

let account\_open\_details=this.account\_open\_details.value.account\_open\_data;

console.log(account\_open\_details);

const name = account\_open\_details.name;

const contact = account\_open\_details.contact;

const address = account\_open\_details.address;

const openingamount = account\_open\_details.openingamount;

const accounttype = account\_open\_details.accounttype;

const password = account\_open\_details.password;

if(name=="" || contact=="" || address=="" || openingamount=="" || accounttype=="" || password==""){

this.warning = "Do not leave blank any field";

}else{

if(!isNan(contact) || !isNan(openingamount) || length(contact) >= 10){

this.warning = "Invalid values";

}

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/customer2bank/open\_account",account\_open\_details)

.subscribe((data)=>{

AppComponent.onShowLoader(0);

console.log(data);

const account\_number = data['account\_number'];

const info = "Your account number and login id is "+account\_number+". You may login with same password you entered before";

alert(info);

this.route.navigate(['']);

},

error=>{

AppComponent.onShowLoader(0);

console.log(error['error']['message']);

});

}

}

}

Dashboard.component.css :

.commonButtonDesign{

background-color: #56baed;

border: none;

color: white;

padding: 15px 80px;

text-align: center;

text-decoration: none;

display: inline-block;

text-transform: uppercase;

font-size: 13px;

-webkit-box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);

box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);

-webkit-border-radius: 5px 5px 5px 5px;

border-radius: 5px 5px 5px 5px;

margin: 5px 20px 40px 20px;

-webkit-transition: all 0.3s ease-in-out;

-moz-transition: all 0.3s ease-in-out;

-ms-transition: all 0.3s ease-in-out;

-o-transition: all 0.3s ease-in-out;

transition: all 0.3s ease-in-out;

}

Dashboard.component.html :

<nav class="navbar navbar-default">

<div class="container-fluid">

<div class="navbar-header">

<a class="navbar-brand" href="dashboard">Welcome {{user.username}}</a>

</div>

<!-- <ul class="nav navbar-nav">

<li class="active"><a href="#">Home</a></li>

<li><a href="#">Page 1</a></li>

<li><a href="#">Page 2</a></li>

</ul> -->

<ul class="nav navbar-nav navbar-right">

<!-- <li><a href="#"><span class="glyphicon glyphicon-user"></span> Sign Up</a></li> -->

<li><a href="#" (click)="logout()"><span class="glyphicon glyphicon-log-in"></span> Logout</a></li>

</ul>

</div>

</nav>

<div class="row">

<div class="col-xs-12 col-sm-12 col-md-12 col-lg-12 text-center">

<button class="commonButtonDesign text-center" (click)="request\_locker()" [ngStyle] = "{'display' : hideButton}" >Apply For Locker - Single</button>

<button class="commonButtonDesign text-center" (click)="request\_locker()" [ngStyle] = "{'display' : hideButton}" >Apply For Locker - Joint Account</button>

</div>

</div>

<div class="row">

<div class="col-xs-12 col-sm-12 col-md-12 col-lg-12 text-center">

<label [ngStyle] = "{'display' : messageVisibility}">You have requested for locker. Please check and try to login after 24 hours. you will be redirected to the panel once bank apporves your locker application</label>

</div>

</div>

Dashboard.component.ts :

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { HttpClient } from '@angular/common/http';

import { AppComponent } from '../app.component';

@Component({

selector: 'app-dashboard',

templateUrl: './dashboard.component.html',

styleUrls: ['./dashboard.component.css']

})

export class DashboardComponent implements OnInit {

user : any = {

"username" : "",

"password" : ""

}

username : any;

hideButton: string;

messageVisibility: string = "none";

serverError: string;

constructor(private route:Router, private http:HttpClient) { }

ngOnInit() {

this.user.username = localStorage.getItem("user");

this.user.password = localStorage.getItem("key");

var role = localStorage.getItem("banker");

if(this.user.username==null){

this.route.navigate(['']);

}else{

if(role == 'banker'){

this.route.navigate(['/bankdashboard']);

}

}

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/customer2bank/check\_locker\_approval",this.user)

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response['status']);

if(response['status'] == 4){

this.route.navigate(['\main']);

}else if (response['status'] == 3){

this.hideButton = "none";

this.messageVisibility = "block";

}else if(response['status']==5){

localStorage.clear();

alert("your locker account is cancelled, please contact your bank");

this.route.navigate(['']);

}else{

// this.hideButton="block";

// this.messageVisibility = "none";

}

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

this.serverError = "Server Unavailable";

});

}

logout(){

localStorage.clear();

history.back();

}

request\_locker(){

console.log("requested");

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/customer2bank/locker\_request",this.user)

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response);

this.hideButton = "none";

this.messageVisibility = "block";

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

this.serverError = "Server Unavailable";

});

}

}

Banklogin.component.css:

/\* BASIC \*/

html {

background-color: #56baed;

}

body {

font-family: "Poppins", sans-serif;

height: 100vh;

}

a {

color: #92badd;

display:inline-block;

text-decoration: none;

font-weight: 400;

}

h2 {

text-align: center;

font-size: 16px;

font-weight: 600;

text-transform: uppercase;

display:inline-block;

margin: 40px 8px 10px 8px;

color: #cccccc;

}

/\* STRUCTURE \*/

.wrapper {

display: flex;

align-items: center;

flex-direction: column;

justify-content: center;

width: 100%;

min-height: 100%;

padding: 20px;

}

#formContent {

-webkit-border-radius: 10px 10px 10px 10px;

border-radius: 10px 10px 10px 10px;

background: #fff;

padding: 30px;

width: 90%;

max-width: 450px;

position: relative;

padding: 0px;

-webkit-box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);

box-shadow: 0 30px 60px 0 rgba(0,0,0,0.3);

text-align: center;

}

#formFooter {

background-color: #f6f6f6;

border-top: 1px solid #dce8f1;

padding: 25px;

text-align: center;

-webkit-border-radius: 0 0 10px 10px;

border-radius: 0 0 10px 10px;

}

/\* TABS \*/

h2.inactive {

color: #cccccc;

}

h2.active {

color: #0d0d0d;

border-bottom: 2px solid #5fbae9;

}

/\* FORM TYPOGRAPHY\*/

input[type=button], input[type=submit], input[type=reset] {

background-color: #56baed;

border: none;

color: white;

padding: 15px 80px;

text-align: center;

text-decoration: none;

display: inline-block;

text-transform: uppercase;

font-size: 13px;

-webkit-box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);

box-shadow: 0 10px 30px 0 rgba(95,186,233,0.4);

-webkit-border-radius: 5px 5px 5px 5px;

border-radius: 5px 5px 5px 5px;

margin: 5px 20px 40px 20px;

-webkit-transition: all 0.3s ease-in-out;

-moz-transition: all 0.3s ease-in-out;

-ms-transition: all 0.3s ease-in-out;

-o-transition: all 0.3s ease-in-out;

transition: all 0.3s ease-in-out;

}

input[type=button]:hover, input[type=submit]:hover, input[type=reset]:hover {

background-color: #39ace7;

}

input[type=button]:active, input[type=submit]:active, input[type=reset]:active {

-moz-transform: scale(0.95);

-webkit-transform: scale(0.95);

-o-transform: scale(0.95);

-ms-transform: scale(0.95);

transform: scale(0.95);

}

input[type=text] {

background-color: #f6f6f6;

border: none;

color: #0d0d0d;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 5px;

width: 85%;

border: 2px solid #f6f6f6;

-webkit-transition: all 0.5s ease-in-out;

-moz-transition: all 0.5s ease-in-out;

-ms-transition: all 0.5s ease-in-out;

-o-transition: all 0.5s ease-in-out;

transition: all 0.5s ease-in-out;

-webkit-border-radius: 5px 5px 5px 5px;

border-radius: 5px 5px 5px 5px;

}

input[type=text]:focus {

background-color: #fff;

border-bottom: 2px solid #5fbae9;

}

input[type=text]:placeholder {

color: #cccccc;

}

input[type=password] {

background-color: #f6f6f6;

border: none;

color: #0d0d0d;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 5px;

width: 85%;

border: 2px solid #f6f6f6;

-webkit-transition: all 0.5s ease-in-out;

-moz-transition: all 0.5s ease-in-out;

-ms-transition: all 0.5s ease-in-out;

-o-transition: all 0.5s ease-in-out;

transition: all 0.5s ease-in-out;

-webkit-border-radius: 5px 5px 5px 5px;

border-radius: 5px 5px 5px 5px;

}

input[type=password]:focus {

background-color: #fff;

border-bottom: 2px solid #5fbae9;

}

input[type=password]:placeholder {

color: #cccccc;

}

/\* ANIMATIONS \*/

/\* Simple CSS3 Fade-in-down Animation \*/

.fadeInDown {

-webkit-animation-name: fadeInDown;

animation-name: fadeInDown;

-webkit-animation-duration: 1s;

animation-duration: 1s;

-webkit-animation-fill-mode: both;

animation-fill-mode: both;

}

@-webkit-keyframes fadeInDown {

0% {

opacity: 0;

-webkit-transform: translate3d(0, -100%, 0);

transform: translate3d(0, -100%, 0);

}

100% {

opacity: 1;

-webkit-transform: none;

transform: none;

}

}

@keyframes fadeInDown {

0% {

opacity: 0;

-webkit-transform: translate3d(0, -100%, 0);

transform: translate3d(0, -100%, 0);

}

100% {

opacity: 1;

-webkit-transform: none;

transform: none;

}

}

/\* Simple CSS3 Fade-in Animation \*/

@-webkit-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

@-moz-keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

@keyframes fadeIn { from { opacity:0; } to { opacity:1; } }

.fadeIn {

opacity:0;

-webkit-animation:fadeIn ease-in 1;

-moz-animation:fadeIn ease-in 1;

animation:fadeIn ease-in 1;

-webkit-animation-fill-mode:forwards;

-moz-animation-fill-mode:forwards;

animation-fill-mode:forwards;

-webkit-animation-duration:1s;

-moz-animation-duration:1s;

animation-duration:1s;

}

.fadeIn.first {

-webkit-animation-delay: 0.4s;

-moz-animation-delay: 0.4s;

animation-delay: 0.4s;

}

.fadeIn.second {

-webkit-animation-delay: 0.6s;

-moz-animation-delay: 0.6s;

animation-delay: 0.6s;

}

.fadeIn.third {

-webkit-animation-delay: 0.8s;

-moz-animation-delay: 0.8s;

animation-delay: 0.8s;

}

.fadeIn.fourth {

-webkit-animation-delay: 1s;

-moz-animation-delay: 1s;

animation-delay: 1s;

}

/\* Simple CSS3 Fade-in Animation \*/

.underlineHover:after {

display: block;

left: 0;

bottom: -10px;

width: 0;

height: 2px;

background-color: #56baed;

content: "";

transition: width 0.2s;

}

.underlineHover:hover {

color: #0d0d0d;

}

.underlineHover:hover:after{

width: 100%;

}

/\* OTHERS \*/

\*:focus {

outline: none;

}

#icon {

width:60%;

}

Banklogin.component.html :

<div class="wrapper fadeInDown">

<div id="formContent">

<!-- Tabs Titles -->

<!-- Icon -->

<div class="fadeIn first">

<h4>Bank Login</h4>

<span style="color:red">{{warning}}</span>

<!-- <img src="https://www.consumer-voice.org/wp-content/uploads/2017/09/bank-locker-insurance.jpg" id="icon" alt="User Icon" /> -->

</div>

<!-- Login Form -->

<form (ngSubmit)="onLogin()" #login="ngForm" name="loginForm" id="loginForm" method="post" role="form">

<div ngModelGroup="loginFormData" #loginFormData = "ngModelGroup">

<input type="text" id="login" class="fadeIn second" name="username" placeholder="login" ngModel required>

<input type="password" id="password" class="fadeIn third" name="password" placeholder="password" ngModel required>

<input type="submit" class="fadeIn fourth" value="Log In">

</div>

</form>

</div>

</div>

Banklogin.component.ts :

import { Component, OnInit, ViewChild } from '@angular/core';

import { AppComponent } from '../app.component';

import { HttpClient } from '@angular/common/http'

import { Router } from '@angular/router';

import { NgForm } from '@angular/forms';

@Component({

selector: 'app-banklogin',

templateUrl: './banklogin.component.html',

styleUrls: ['./banklogin.component.css']

})

export class BankloginComponent implements OnInit {

@ViewChild('login') loginData : NgForm;

warning : string ;

constructor(private http: HttpClient, private route:Router) { }

ngOnInit() {

}

onLogin(){

console.log("login hit");

let loginData=this.loginData.value.loginFormData;

console.log(loginData);

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/login/bankauth",loginData)

.subscribe((data)=>{

AppComponent.onShowLoader(0);

// console.log(data);

if(data['role']=="banker"){

localStorage.setItem("user",loginData.username);

localStorage.setItem("key",loginData.password);

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

this.route.navigate(['/bankdashboard']);

}else{

this.warning = "Invalid credentials";

console.log("Invalid credentials");

}

},

error=>{

console.log(error['error']['message']);

});

}

}

Bankdashboard.component.css :

th{

text-align:center;

}

Bankdashboard.component.html:

<div class="container text-center">

<h4>Welcome Bank</h4>

</div>

<nav class="navbar navbar-default">

<div class="container-fluid">

<div class="navbar-header">

<a class="navbar-brand" href="bankdashboard">{{user.username}}</a>

</div>

<ul class="nav navbar-nav navbar-right">

<!-- <li><a href="#"><span class="glyphicon glyphicon-user"></span> Sign Up</a></li> -->

<li><button class="btn btn-link" (click)="logout()"><span class="glyphicon glyphicon-log-in"></span> Logout</button></li>

</ul>

</div>

</nav>

<div class="container text-center">

<h5>You can see all the requests here and keep the track of available locker being used by customer</h5>

<hr><br>

<ul class="nav nav-tabs">

<li class="active"><a data-toggle="tab" href="#home" (click)="refreshQueue()">Locker Application Queue</a></li>

<li><a data-toggle="tab" href="#menu1" (click)="refreshLocker()">Locker Availability</a></li>

<li><a data-toggle="tab" href="#menu2" (click)="refreshAppointment()">Locker Appointments</a></li>

<li><a data-toggle="tab" href="#menu3" (click)="refreshAccounts()">Active Accounts</a></li>

</ul>

<div class="tab-content">

<div id="home" class="tab-pane fade in active">

<h3>Locker Application Queue</h3>

<p>Locker application queue : these people are waiting for free locker</p>

<!-- {{test}} -->

<table class="table table-responsive table-bordered table-hover ">

<thead>

<tr>

<th>S.no</th>

<th>Locker Application Id</th>

<th>Account Number</th>

<th>Requested on</th>

<th>Operations</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let data of records; let i=index;">

<td>{{i+1}}</td>

<td>{{data.locker\_application\_id}}</td>

<td>{{data.customer\_id}}</td>

<td>{{data.requested\_date}}</td>

<td>

<button class="btn btn-success" (click)="approve(data.customer\_id)">Approve</button>

</td>

</tr>

</tbody>

</table>

</div>

<div id="menu1" class="tab-pane fade">

<h3>Available Locker</h3>

<p>Locker are being used by customers</p>

<table class="table table-responsive table-bordered table-hover ">

<thead>

<tr>

<th>S.no</th>

<th>Locker ID</th>

<th>Assigned to (Account Number)</th>

<th>Availability</th>

<th>Operations</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let data of availableLocker; let i=index;">

<td \*ngIf="!data.account\_number == false; else s">{{i+1}}</td>

<ng-template #s><td style="background-color:lightgreen">{{i+1}}</td></ng-template>

<td \*ngIf="!data.account\_number == false; else id">{{data.locker\_id}}</td>

<ng-template #id><td style="background-color:lightgreen">{{data.locker\_id}}</td></ng-template>

<td \*ngIf="!data.account\_number == false; else elseBlock">{{data.account\_number}}

</td>

<ng-template #elseBlock><td style="background-color:lightgreen">Not Assigned Yet!</td></ng-template>

<td \*ngIf="!data.account\_number == false; else availability">{{data.availability}}</td>

<ng-template #availability><td style="background-color:lightgreen">{{data.availability}}</td></ng-template>

<td \*ngIf="!data.account\_number == false else elsecancel">

<button class="btn btn-danger" (click)="cancel(data.account\_number)">Cancel</button>

</td>

<ng-template #elsecancel><td style="background-color:lightgreen"></td></ng-template>

</tr>

</tbody>

</table>

</div>

<div id="menu2" class="tab-pane fade">

<h3>Locker Appointments </h3>

<p>You can approve or reject </p>

<table class="table table-responsive table-bordered table-hover ">

<thead>

<tr>

<th>S.no</th>

<th>Locker ID</th>

<th>Access ID</th>

<th>Account</th>

<th>Date</th>

<th>Time</th>

<th>Charges</th>

<th>Operations</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let data of appointments; let i=index;">

<td>{{i+1}}</td>

<td>{{data.locker\_id}}</td>

<td>{{data.access\_id}}</td>

<td>{{data.account}}</td>

<td>{{data.date}}</td>

<td>{{data.time}}</td>

<td>{{data.charges}}</td>

<td>

<div class="btn-group">

<button class="btn btn-success" (click)="approveAppointment(data.access\_id)">Approve</button>

<button class="btn btn-danger" (click)="rejectAppointment(data.access\_id)">Reject</button>

</div>

</td>

</tr>

</tbody>

</table>

</div>

<div id="menu3" class="tab-pane fade">

<h3>All Active Accounts </h3>

<table class="table table-responsive table-bordered table-hover ">

<thead>

<tr>

<th>S.no</th>

<th>Account Number</th>

<th>Account Name</th>

<th>Current Balance</th>

<th>Contact</th>

<th>Address</th>

<th>Account Type</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let data of accounts; let i=index;">

<td>{{i+1}}</td>

<td>{{data.username}}</td>

<td>{{data.name}}</td>

<td>{{data.balance}}</td>

<td>{{data.contact}}</td>

<td>{{data.address}}</td>

<td>{{data.accounttype}}</td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

Bankdashboard.component.ts :

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { AppComponent } from '../app.component';

import { HttpClient } from '@angular/common/http';

@Component({

selector: 'app-bankdashboard',

templateUrl: './bankdashboard.component.html',

styleUrls: ['./bankdashboard.component.css']

})

export class BankdashboardComponent implements OnInit {

records: any;

test: string;

availableLocker: any;

appointments: any;

accounts : any;

constructor(private route:Router, private http:HttpClient) { }

user : any = {

"username" : "",

"password" : ""

}

ngOnInit() {

this.test = "hey hey ngOnInit";

this.user.username = localStorage.getItem("user");

this.user.password = localStorage.getItem("key");

if(this.user.username==null){

this.route.navigate(['']);

}else{

//locker appointments

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/bank/appointments",this.user)

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response);

this.appointments = response['appointments'];

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

});

// locker application queue

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/bank/queue",this.user)

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response);

this.records = response['approvals'];

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

});

//Locker Availability

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/bank/overalllocker",{

"username" : this.user.username,

"password" : this.user.password,

})

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response);

this.availableLocker = response['locker\_info'];

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

});

//currently active accounts

AppComponent.onShowLoader(1);

this.http.get("http://localhost:8080/api/miniproject/bank/active\_accounts")

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response);

this.accounts = response['accounts'];

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

});

}

}

logout(){

localStorage.clear();

this.route.navigate(['bank']);

}

approve(data:any){

console.log(data);

// this.route.navigateByUrl('/bankdashboard', {skipLocationChange: true}).then(()=>this.route.navigate(["/bankdashboard"]));

this.test="I am approved";

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/bank/approve\_locker\_application",{

"username" : this.user.username,

"password" : this.user.password,

"customer\_id" : data

})

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response);

if(response['locker']=="NA"){

alert("Lockers are not available");

}else{

this.records = response['approvals'];

}

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

});

}

cancel(data:any){

console.log(data);

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/bank/cancel\_locker\_application",{

"username" : this.user.username,

"password" : this.user.password,

"customer\_id" : data

})

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response);

this.availableLocker = response['locker\_info'];

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

});

}

refreshQueue(){

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/bank/queue",this.user)

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response);

this.records = response['approvals'];

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

});

}

refreshLocker(){

//Locker Availability

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/bank/overalllocker",{

"username" : this.user.username,

"password" : this.user.password,

})

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response);

this.availableLocker = response['locker\_info'];

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

});

}

refreshAppointment(){

console.log("appointment section is refreshed");

//locker appointments

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/bank/appointments",this.user)

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response);

this.appointments = response['appointments'];

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

});

}

approveAppointment(access\_id : any){

//approve locker appointments

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/bank/approveAppointment",{

"username":this.user.username,

"password":this.user.password,

"access\_id":access\_id

})

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response);

this.appointments = response['appointments'];

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

});

}

rejectAppointment(access\_id : any ){

//reject locker appointments

AppComponent.onShowLoader(1);

this.http.post("http://localhost:8080/api/miniproject/bank/rejectAppointment",{

"username":this.user.username,

"password":this.user.password,

"access\_id":access\_id

})

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response);

this.appointments = response['appointments'];

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

});

}

refreshAccounts(){

//reject locker appointments

AppComponent.onShowLoader(1);

this.http.get("http://localhost:8080/api/miniproject/bank/active\_accounts")

.subscribe(response=>{

AppComponent.onShowLoader(0);

console.log(response);

this.accounts = response['accounts'];

},

error=>{

AppComponent.onShowLoader(0);

console.log(error);

});

}

}

* + - 1. **Backend code :**

Backend code is divided into two sections that are models and services:

**Services:**

**LoginService.java**

package ignou.miniproject;

import ignou.miniproject.model.Users;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import javax.ws.rs.Consumes;

import javax.ws.rs.FormParam;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

//import static javax.ws.rs.HttpMethod.POST;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

import org.json.simple.JSONObject;

import org.json.simple.parser.JSONParser;

import org.json.simple.parser.ParseException;

@Path("/login")

public class LoginService {

private Users users= null;

@POST

@Path("/send")

@Consumes(MediaType.APPLICATION\_JSON)

public Response auth(

@PathParam("username") String username

){

return Response.status(200)

.entity("got"+username)

.build();

}

@POST

@Path("/auth")

@Consumes(MediaType.APPLICATION\_JSON)

public Response consumeJSON( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

// pass the username and password for authentication from database

this.users = new Users();

Boolean result = this.users.auth(username, password);

JSONObject result\_json = new JSONObject();

result\_json.put("auth", result);

return Response.status(200).entity(result\_json.toString()).build();

}

@POST

@Path("/bankauth")

@Consumes(MediaType.APPLICATION\_JSON)

public Response bankLogin( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

// pass the username and password for authentication from database

this.users = new Users();

Boolean result = this.users.bankauth(username, password);

JSONObject result\_json = new JSONObject();

if(result){

result\_json.put("role", "banker");

return Response.status(200).entity(result\_json.toString()).build();

}else{

result\_json.put("role", "customer");

return Response.status(200).entity(result\_json.toString()).build();

}

}

/\*

@GET

@Path("/test")

@Produces(MediaType.TEXT\_PLAIN)

public Response getJSONData(

@FormParam("name") String name

){

JSONObject json = new JSONObject();

json.put("name",name);

json.put("code","1010011");

json.put("age",21);

// return String.valueOf(json);

return Response.status(200)

.entity("got"+name)

.build();

}

\*/

}

**CustomerService.java:**

package ignou.miniproject;

import ignou.miniproject.model.Users;

import ignou.miniproject.model.Bank;

import java.sql.SQLException;

import java.util.ArrayList;

import javax.ws.rs.Consumes;

import javax.ws.rs.POST;

//import static javax.ws.rs.HttpMethod.POST;

import javax.ws.rs.Path;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

import org.json.simple.JSONObject;

import org.json.simple.parser.JSONParser;

import org.json.simple.parser.ParseException;

@Path("/customer2bank")

public class CustomerService {

private Users users=null;

private Bank bank=null;

@POST

@Path("/open\_account")

@Consumes(MediaType.APPLICATION\_JSON)

public Response open\_account( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String name = (String) json.get("name");

String contact = (String) json.get("contact");

String address = (String) json.get("address");

String openingamount = Long.toString((Long)json.get("openingamount"));

String accounttype = (String) json.get("accounttype");

String password = (String) json.get("password");

JSONObject result\_json = new JSONObject();

this.bank = new Bank();

String account\_number = this.bank.openAccount(name, contact, address, openingamount, accounttype, password);

result\_json.put("account\_number", account\_number);

return Response.status(200).entity(result\_json.toString()).build();

}

@POST

@Path("/locker\_request")

@Consumes(MediaType.APPLICATION\_JSON)

public Response locker\_Request( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.auth(username, password);

JSONObject result\_json = new JSONObject();

if(result){

//insert user into addlocker\_request with dates

int is\_requested;

is\_requested = this.bank.locker\_request(username);

result\_json.put("auth", is\_requested);

return Response.status(200).entity(result\_json.toString()).build();

}else{

result\_json.put("status", result);

return Response.status(200).entity(result\_json.toString()).build();

}

}

@POST

@Path("/check\_locker\_request")

@Consumes(MediaType.APPLICATION\_JSON)

public Response check\_locker\_Request( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

//

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.auth(username, password);

JSONObject result\_json = new JSONObject();

if(result){

//insert user into addlocker\_request with dates

int is\_requested;

is\_requested = this.bank.check\_locker\_request(username);

ArrayList list = this.bank.getLockerDetails(username);

result\_json.put("locker\_id",list.get(1));

result\_json.put("issued\_on",list.get(2));

result\_json.put("expired\_on",list.get(3));

result\_json.put("balance",list.get(0));

result\_json.put("status", is\_requested);

return Response.status(200).entity(result\_json.toString()).build();

}else{

result\_json.put("status", result);

return Response.status(200).entity(result\_json.toString()).build();

}

// return Response.status(200)

// .entity((login\_json))

// .build();

}

@POST

@Path("/check\_locker\_approval")

@Consumes(MediaType.APPLICATION\_JSON)

public Response check\_locker\_Approval( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

//

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.auth(username, password);

JSONObject result\_json = new JSONObject();

if(result){

//insert user into addlocker\_request with dates

int is\_requested;

is\_requested = this.bank.check\_locker\_request(username);

result\_json.put("status", is\_requested);

return Response.status(200).entity(result\_json.toString()).build();

}else{

result\_json.put("status", result);

return Response.status(200).entity(result\_json.toString()).build();

}

// return Response.status(200)

// .entity((login\_json))

// .build();

}

@POST

@Path("/balance")

@Consumes(MediaType.APPLICATION\_JSON)

public Response getBalance( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

//

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.auth(username, password);

JSONObject result\_json = new JSONObject();

if(result){

//insert user into addlocker\_request with dates

int balance;

balance = this.bank.getBalance(username);

result\_json.put("balance", balance);

return Response.status(200).entity(result\_json.toString()).build();

}else{

result\_json.put("status", result);

return Response.status(200).entity(result\_json.toString()).build();

}

//

// return Response.status(200)

// .entity((login\_json))

// .build();

}

@POST

@Path("/locker\_issued\_expiry\_date")

@Consumes(MediaType.APPLICATION\_JSON)

public Response locker\_issued\_expiry\_date( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

//

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.auth(username, password);

JSONObject result\_json = new JSONObject();

if(result){

//insert user into addlocker\_request with dates

ArrayList dates;

dates = this.bank.locker\_issued\_expiry\_date(username);

result\_json.put("issued", dates.get(0));

result\_json.put("expiry", dates.get(1));

return Response.status(200).entity(result\_json.toString()).build();

}else{

result\_json.put("status", result);

return Response.status(200).entity(result\_json.toString()).build();

}

// return Response.status(200)

// .entity((login\_json))

// .build();

}

@POST

@Path("/lockeraccessrequest")

@Consumes(MediaType.APPLICATION\_JSON)

public Response lockerAccessRequest( String request\_data\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(request\_data\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

String date = (String)json.get("Date");

String time = (String)json.get("Time");

//

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.auth(username, password);

JSONObject result\_json = new JSONObject();

if(result){

//insert user into addlocker\_request with dates

int requested;

requested = this.bank.addLockerAccessRequest(username, date, time);

result\_json.put("requested", requested);

return Response.status(200).entity(result\_json.toString()).build();

}else{

result\_json.put("status", result);

return Response.status(200).entity(result\_json.toString()).build();

}

// return Response.status(200)

// .entity((login\_json))

// .build();

}

@POST

@Path("/checkpendinglockeraccessrequest")

@Consumes(MediaType.APPLICATION\_JSON)

public Response checkPendingLockerAccessRequest( String request\_data\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(request\_data\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

// pass the username and password for authen-tication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.auth(username, password);

JSONObject result\_json = new JSONObject();

if(result){

//insert user into addlocker\_request with dates

int locker\_access\_id;

locker\_access\_id = this.bank.checkPendingLockerAccessRequest(username);

int is\_rejected;

is\_rejected = this.bank.checkRejectedLockerAccessRequest(username);

// if locker\_access\_id is 0 then no pending request is there

result\_json.put("LockerAccessId", locker\_access\_id);

result\_json.put("is\_rejected", is\_rejected);

return Response.status(200).entity(result\_json.toString()).build();

}else{

result\_json.put("status", result);

return Response.status(200).entity(result\_json.toString()).build();

}

}

@POST

@Path("/approvedlockerrequests")

@Consumes(MediaType.APPLICATION\_JSON)

public Response approvedlockerrequests( String request\_data\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(request\_data\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.auth(username, password);

JSONObject result\_json = new JSONObject();

if(result){

//insert user into addlocker\_request with dates

JSONObject approvals;

approvals = this.bank.approvedLockerRequests(username);

// if locker\_access\_id is 0 then no pending request is there

// result\_json.put("checking", "checked");

return Response.status(200).entity(approvals.toString()).build();

}else{

result\_json.put("status", result);

return Response.status(200).entity(result\_json.toString()).build();

}

}

}

**BankService.java:**

package ignou.miniproject;

import ignou.miniproject.model.Bank;

import ignou.miniproject.model.Users;

import java.sql.SQLException;

import javax.ws.rs.Consumes;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

//import static javax.ws.rs.HttpMethod.POST;

import javax.ws.rs.Path;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

import org.json.simple.JSONObject;

import org.json.simple.parser.JSONParser;

import org.json.simple.parser.ParseException;

@Path("/bank")

public class BankService {

private Users users= null;

private Bank bank = null;

@GET

@Path("/active\_accounts")

public Response activeAccounts() throws SQLException{

this.bank = new Bank();

JSONObject locker\_details = this.bank.allActiveAccounts();

return Response.status(200).entity(locker\_details.toJSONString()).build();

}

// rejectAppointment

@POST

@Path("/rejectAppointment")

@Consumes(MediaType.APPLICATION\_JSON)

public Response rejectAppointment( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

int access\_id = Integer.parseInt(json.get("access\_id").toString()) ;

JSONObject result\_json = new JSONObject();

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.bankauth(username, password);

if(result){

Boolean rejectAppointment;

rejectAppointment = this.bank.rejectAppointment(access\_id);

if(rejectAppointment){

result\_json.put("locker", "approved");

JSONObject locker\_details;

locker\_details = this.bank.allAppointments();

return Response.status(200).entity(locker\_details.toString()).build();

}else{

result\_json.put("locker", "NA");

return Response.status(200).entity(result\_json.toString()).build();

}

}else{

result\_json.put("status", "false\_l");

return Response.status(200).entity(result\_json.toString()).build();

}

// JSONObject result\_json = new JSONObject();

// result\_json.put("auth", result);

// return Response.status(200).entity(result\_json.toString()).build();

}

// approveAppointment

@POST

@Path("/approveAppointment")

@Consumes(MediaType.APPLICATION\_JSON)

public Response approveAppointment( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

int access\_id = Integer.parseInt(json.get("access\_id").toString()) ;

JSONObject result\_json = new JSONObject();

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.bankauth(username, password);

if(result){

Boolean apporveAppointment;

apporveAppointment = this.bank.setPermission2One(access\_id);

if(apporveAppointment){

result\_json.put("locker", "approved");

JSONObject locker\_details;

locker\_details = this.bank.allAppointments();

return Response.status(200).entity(locker\_details.toString()).build();

}else{

result\_json.put("locker", "NA");

return Response.status(200).entity(result\_json.toString()).build();

}

}else{

result\_json.put("status", "false\_l");

return Response.status(200).entity(result\_json.toString()).build();

}

// JSONObject result\_json = new JSONObject();

// result\_json.put("auth", result);

// return Response.status(200).entity(result\_json.toString()).build();

}

// appointments

@POST

@Path("/appointments")

@Consumes(MediaType.APPLICATION\_JSON)

public Response appoointments( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

// int customer\_id = Integer.parseInt(json.get("customer\_id").toString()) ;

JSONObject result\_json = new JSONObject();

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.bankauth(username, password);

if(result){

JSONObject locker\_details;

locker\_details = this.bank.allAppointments();

System.out.println("-------------------------"+locker\_details+"--------------------------------------------------------");

return Response.status(200).entity(locker\_details.toString()).build();

}else{

result\_json.put("status", "login error");

return Response.status(200).entity(result\_json.toString()).build();

}

}

//cancel\_locker\_application

@POST

@Path("/cancel\_locker\_application")

@Consumes(MediaType.APPLICATION\_JSON)

public Response cancelLocker( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

int customer\_id = Integer.parseInt(json.get("customer\_id").toString()) ;

JSONObject result\_json = new JSONObject();

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.bankauth(username, password);

if(result){

Boolean removeCustomerId = this.bank.removeCustomerId(customer\_id);

Boolean insertCancellationDataIntoAddLocker\_requestTable = this.bank.insertCancellationDataIntoAddLocker\_requestTable(customer\_id);

if(removeCustomerId && insertCancellationDataIntoAddLocker\_requestTable){

JSONObject locker\_details;

locker\_details = this.bank.allLockerTableData();

System.out.println("-------------------------"+locker\_details+"--------------------------------------------------------");

return Response.status(200).entity(locker\_details.toString()).build();

}else{

result\_json.put("status","cannot cancel");

return Response.status(200).entity(result\_json.toString()).build();

}

}else{

result\_json.put("status", "login error");

return Response.status(200).entity(result\_json.toString()).build();

}

// JSONObject result\_json = new JSONObject();

// result\_json.put("auth", result);

// return Response.status(200).entity(result\_json.toString()).build();

}

// overalllocker

@POST

@Path("/overalllocker")

@Consumes(MediaType.APPLICATION\_JSON)

public Response overalllocker( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

// int customer\_id = Integer.parseInt(json.get("customer\_id").toString()) ;

JSONObject result\_json = new JSONObject();

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.bankauth(username, password);

if(result){

JSONObject approvals;

approvals = this.bank.allLockerTableData();

return Response.status(200).entity(approvals.toString()).build();

}else{

result\_json.put("status", "login error");

return Response.status(200).entity(result\_json.toString()).build();

}

// JSONObject result\_json = new JSONObject();

// result\_json.put("auth", result);

// return Response.status(200).entity(result\_json.toString()).build();

}

// approve\_locker\_application

@POST

@Path("/approve\_locker\_application")

@Consumes(MediaType.APPLICATION\_JSON)

public Response approveLocker( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

int customer\_id = Integer.parseInt(json.get("customer\_id").toString()) ;

JSONObject result\_json = new JSONObject();

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.bankauth(username, password);

if(result){

//update the locker table with customer id

// Boolean updatedLockerTable=true;

Boolean updatedLockerTable;

updatedLockerTable = this.bank.updateLockerTable(customer\_id);

if(updatedLockerTable){

result\_json.put("locker", "approved");

Boolean updatedAddLockerRequest = this.bank.updatedAddLockerRequestTable(customer\_id);

JSONObject approvals;

approvals = this.bank.allTheLockerRequest();

return Response.status(200).entity(approvals.toString()).build();

}else{

result\_json.put("locker", "NA");

return Response.status(200).entity(result\_json.toString()).build();

}

}else{

result\_json.put("status", "false\_l");

return Response.status(200).entity(result\_json.toString()).build();

}

// JSONObject result\_json = new JSONObject();

// result\_json.put("auth", result);

// return Response.status(200).entity(result\_json.toString()).build();

}

@POST

@Path("/queue")

@Consumes(MediaType.APPLICATION\_JSON)

public Response consumeJSON( String login\_json ) throws ParseException, SQLException {

JSONParser parser = new JSONParser();

JSONObject json = (JSONObject) parser.parse(login\_json);

String username = (String) json.get("username");

String password = (String) json.get("password");

JSONObject result\_json = new JSONObject();

// pass the username and password for authentication from database

this.users = new Users();

this.bank = new Bank();

Boolean result = this.users.bankauth(username, password);

if(result){

//get all the data from addlocker\_request

JSONObject approvals;

approvals = this.bank.allTheLockerRequest();

return Response.status(200).entity(approvals.toString()).build();

}else{

result\_json.put("status", result);

return Response.status(200).entity(result\_json.toString()).build();

}

// JSONObject result\_json = new JSONObject();

// result\_json.put("auth", result);

// return Response.status(200).entity(result\_json.toString()).build();

}

}

**Models:**

**Conn.java –** this is a connection file to provide mysql connection :

package ignou.miniproject.model;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class Conn {

private static Connection con = null;

public static Connection getMysqlConnection(){

try{

Class.forName("com.mysql.jdbc.Driver");

Conn.con=DriverManager.getConnection("jdbc:mysql://localhost:3306/miniproject","root","");

return Conn.con;

}catch(ClassNotFoundException | SQLException e){

System.out.println(e);

return null;

}

}

}

**Users.java:**

package ignou.miniproject.model;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class Users {

private String username;

private String password;

private Connection conn;

/\*\*

\* @param username

\* @param password

\* @return the username

\* @throws java.sql.SQLException

\*/

public Boolean auth(String username, String password) throws SQLException{

this.username = username;

this.password = password;

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("select \* from users where nature=? and username=? and password=?");

stmt.setString(1,"customer");

stmt.setString(2,username);

stmt.setString(3,password);

ResultSet rs = stmt.executeQuery();

return rs.next();

}

public Boolean bankauth(String username, String password) throws SQLException{

this.username = username;

this.password = password;

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("select \* from users where nature=? and username=? and password=?");

stmt.setString(1,"banker");

stmt.setString(2,username);

stmt.setString(3,password);

ResultSet rs = stmt.executeQuery();

return rs.next();

}

public String getUsername() {

return username;

}

/\*\*

\* @param username the username to set

\*/

public void setUsername(String username) {

this.username = username;

}

/\*\*

\* @return the password

\*/

public String getPassword() {

return password;

}

/\*\*

\* @param password the password to set

\*/

public void setPassword(String password) {

this.password = password;

}

}

**Bank.java :**

package ignou.miniproject.model;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.Date;

import java.util.HashMap;

import org.json.simple.JSONArray;

import org.json.simple.JSONObject;

public class Bank {

private int id;

// store the name of the customer who is applying for locker

private String customer;

private Date request\_date;

private Date approved\_date;

private Date cancellation\_date;

private Connection conn;

public int locker\_request(String username) throws SQLException {

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("insert into addlocker\_request (`customer`, `request\_date`) values((SELECT id FROM users where username=?),?) ");

stmt.setString(1,username);

DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");

stmt.setString(2,dateFormat.format(new Date()));

// Calendar cal = Calendar.getInstance();

// Date today = cal.getTime();

// System.out.println("================================="+today+"================================================================");

// cal.add(Calendar.YEAR, 1); // to get previous year add 1

// cal.add(Calendar.DAY\_OF\_MONTH, -1); // to get previous day add -1

// Date expiryDate = cal.getTime();

// System.out.println("================================="+expiryDate+"================================================================");

//

//

// stmt.setString(3,dateFormat.format(expiryDate));

int rs = stmt.executeUpdate();

return rs;

}

public int check\_locker\_request(String username) throws SQLException {

this.conn = Conn.getMysqlConnection();

// PreparedStatement stmt=this.conn.prepareStatement("insert into addlocker\_request (`customer`, `request\_date`) values((SELECT id FROM users where username=?),?) ");

PreparedStatement stmt=this.conn.prepareStatement("SELECT \* FROM addlocker\_request where customer = (SELECT id FROM users where username=?)");

stmt.setString(1,username);

ResultSet rs = stmt.executeQuery();

if(rs.next()){

Date request\_date = rs.getDate("request\_date");

if(request\_date!=null){

Date approved\_date = rs.getDate("approved\_date");

Date cancelled\_date = rs.getDate("cancelled\_date");

if(cancelled\_date!=null){

return 5;

}

else if(approved\_date!=null){

return 4;

}else{

return 3;

}

}

}

return 0;

}

public int getBalance(String username) throws SQLException {

this.conn = Conn.getMysqlConnection();

// PreparedStatement stmt=this.conn.prepareStatement("insert into addlocker\_request (`customer`, `request\_date`) values((SELECT id FROM users where username=?),?) ");

PreparedStatement stmt=this.conn.prepareStatement("SELECT \* FROM users where username=?");

stmt.setString(1,username);

ResultSet rs = stmt.executeQuery();

if(rs.next()){

int balance = rs.getInt("balance");

return balance;

}

return 0;

}

public ArrayList locker\_issued\_expiry\_date(String username) throws SQLException {

this.conn = Conn.getMysqlConnection();

// PreparedStatement stmt=this.conn.prepareStatement("insert into addlocker\_request (`customer`, `request\_date`) values((SELECT id FROM users where username=?),?) ");

PreparedStatement stmt=this.conn.prepareStatement("SELECT \* FROM addlocker\_request where customer = (SELECT id FROM users where username=?)");

stmt.setString(1,username);

ResultSet rs = stmt.executeQuery();

if(rs.next()){

Date approved\_date = rs.getDate("approved\_date");

Date expiry\_date = rs.getDate("expiry\_date");

String approved = new SimpleDateFormat("yyyy/MM/dd").format(approved\_date);

String expiry = new SimpleDateFormat("yyyy/MM/dd").format(expiry\_date);

ArrayList datelist = new ArrayList();

datelist.add(approved);

datelist.add(expiry);

return datelist;

}

return null;

}

public int addLockerAccessRequest(String username, String date, String time) throws SQLException {

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("insert into access\_locker (`locker\_id`,`date`,`time`,`permission`,`charges`) values((SELECT locker.id FROM locker INNER JOIN users ON users.id=locker.customer where users.username=?),?,?,?,?) ");

stmt.setString(1,username);

stmt.setString(2,date);

stmt.setString(3,time);

stmt.setInt(4,0);

PreparedStatement countappointment=this.conn.prepareStatement("SELECT count(\*) FROM access\_locker INNER JOIN locker ON locker.id=access\_locker.locker\_id inner join users on users.id=locker.customer WHERE access\_locker.permission=1 and users.username = ? ");

countappointment.setString(1, username);

// ResultSet rs = stmt.executeQuery();

ResultSet countResult = countappointment.executeQuery();

if(countResult.next()){

int free\_access = countResult.getInt(1);

if(free\_access>12){

stmt.setInt(5,50);

}else{

stmt.setInt(5,0);

}

}

// DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");

// stmt.setString(2,dateFormat.format(new Date()));

int rs = stmt.executeUpdate();

if(rs == 1){

// deduct money from account

PreparedStatement fetchCurrentBalance=this.conn.prepareStatement("Select balance from users where username=?");

fetchCurrentBalance.setString(1, username);

ResultSet currentBalance = fetchCurrentBalance.executeQuery();

int c\_balance = 0;

if(currentBalance.next()){

c\_balance = currentBalance.getInt("balance");

PreparedStatement deductLockerAccessFee=this.conn.prepareStatement("UPDATE users set balance=? where username=? ");

//charges = 200 per visit after 12 free access

deductLockerAccessFee.setInt(1,(c\_balance-200));

deductLockerAccessFee.setString(2, username);

int balance\_deducted = deductLockerAccessFee.executeUpdate();

return balance\_deducted;

}

}else{

}

return 0;

}

/\*\*

\* checkPendingLockerAccessRequest : this method checks for pending request and return 'access id' of that request

\*/

public int checkPendingLockerAccessRequest(String username) throws SQLException {

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("SELECT access\_locker.id FROM access\_locker INNER JOIN locker ON locker.id=access\_locker.locker\_id inner join users on users.id=locker.customer WHERE access\_locker.permission=0 and users.username=?");

stmt.setString(1,username);

ResultSet rs = stmt.executeQuery();

if(rs.next()){

int access\_id = rs.getInt("id");

return access\_id;

}

return 0;

}

public int checkRejectedLockerAccessRequest(String username) throws SQLException {

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("SELECT access\_locker.id FROM access\_locker INNER JOIN locker ON locker.id=access\_locker.locker\_id inner join users on users.id=locker.customer WHERE access\_locker.permission=2 and users.username=?");

stmt.setString(1,username);

ResultSet rs = stmt.executeQuery();

if(rs.next()){

int access\_id = rs.getInt("id");

// return access\_id;

return 1;

}

return 0;

}

/\*\*

\* checkPendingLockerAccessRequest : this method checks for pending request and return 'access id' of that request

\*/

public JSONObject approvedLockerRequests(String username) throws SQLException {

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("SELECT access\_locker.\* FROM access\_locker INNER JOIN locker ON locker.id=access\_locker.locker\_id inner join users on users.id=locker.customer WHERE access\_locker.permission=1 and users.username=?");

stmt.setString(1,username);

// stmt.setString(2,username);

ResultSet rs = stmt.executeQuery();

// if(rs.next()){

// approvals.put("access\_id", rs.getInt("locker\_id"));

// approvals.put("date", rs.getString("date"));

// }

JSONArray jsonList = new JSONArray();

while(rs.next()){

JSONObject approvals = new JSONObject();

approvals.put("access\_id", rs.getInt("id"));

approvals.put("date", rs.getString("date"));

approvals.put("charges",rs.getInt("charges"));

// jsonList.put(approvals);

jsonList.add(approvals);

}

System.out.println("======================================================"+jsonList);

JSONObject data=new JSONObject();

data.put("approvals", jsonList);

System.out.println("======================================================"+data);

return data;

// return (JSONObject) new JSONObject().put("approvals", jsonList);

}

public JSONObject allTheLockerRequest() throws SQLException {

System.out.println("======================================================check");

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("SELECT users.id, users.username, addlocker\_request.\* from addlocker\_request INNER JOIN users on users.id=addlocker\_request.customer where approved\_date IS NULL");

ResultSet rs = stmt.executeQuery();

JSONArray jsonList = new JSONArray();

while(rs.next()){

JSONObject approvals = new JSONObject();

approvals.put("locker\_application\_id", rs.getInt("addlocker\_request.id"));

approvals.put("username", rs.getString("username"));

approvals.put("requested\_date",rs.getString("request\_date"));

approvals.put("customer\_id",rs.getInt("users.id"));

// jsonList.put(approvals);

jsonList.add(approvals);

}

System.out.println("======================================================"+jsonList);

JSONObject data=new JSONObject();

data.put("approvals", jsonList);

System.out.println("======================================================"+data);

return data;

}

public Boolean updateLockerTable(int customer\_id) throws SQLException {

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* "+customer\_id);

this.conn = Conn.getMysqlConnection();

int l\_id = 0;

//get the id of unused locker

PreparedStatement free\_locker\_id=this.conn.prepareStatement("SELECT id FROM `locker` WHERE customer is null limit 1");

ResultSet rs = free\_locker\_id.executeQuery();

if(rs.next()){

l\_id = rs.getInt("id");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Deducting the money from user account\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ");

PreparedStatement deductMoney = this.conn.prepareStatement("UPDATE users as a INNER join users as b on a.id=b.id set a.balance = b.balance - 1520 WHERE a.id=?");

deductMoney.setInt(1, customer\_id);

int money\_deducted = deductMoney.executeUpdate();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Money Deducted \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* "+money\_deducted+" from "+customer\_id);

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*free lock id\*\*\*\*\*\*\*\* "+l\_id);

// rs.close();

PreparedStatement stmt=this.conn.prepareStatement("UPDATE `locker` SET `customer` = ?, `available`='no' WHERE `locker`.`id` = ?");

stmt.setInt(1, customer\_id);

stmt.setInt(2, l\_id);

int rs\_updation = stmt.executeUpdate();

// stmt.close();

return rs\_updation==1;

}

public Boolean updatedAddLockerRequestTable(int customer\_id) throws SQLException {

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("UPDATE `addlocker\_request` SET `approved\_date` = ? , `expiry\_date` = ? WHERE `addlocker\_request`.`customer` = ?;");

Calendar c = Calendar.getInstance();

c.setTime(new Date());

c.add(Calendar.YEAR, 1);

Date newDate = c.getTime();

DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");

stmt.setString(1, dateFormat.format(new Date()));

stmt.setString(2,dateFormat.format(newDate));

stmt.setInt(3, customer\_id);

int rs = stmt.executeUpdate();

return rs==1;

}

public ArrayList getLockerDetails(String username) throws SQLException {

//get requested date - addlocker\_request

//get expired date - addlocker\_request

// balance - users

// locker id - locker

this.conn = Conn.getMysqlConnection();

int l\_id = 0;

//get the id of unused locker

PreparedStatement locker\_info=this.conn.prepareStatement("SELECT users.balance, locker.id, addlocker\_request.approved\_date, addlocker\_request.expiry\_date from addlocker\_request inner join users on users.id=addlocker\_request.customer inner join locker on users.id=locker.customer where users.username=?");

locker\_info.setString(1, username);

ResultSet rs = locker\_info.executeQuery();

ArrayList locker\_details = new ArrayList();

if(rs.next()){

locker\_details.add(rs.getInt("balance"));

locker\_details.add(rs.getInt("id"));

locker\_details.add(rs.getDate("approved\_date"));

locker\_details.add(rs.getDate("expiry\_date"));

}

System.out.println("Locker ========================== "+locker\_details);

return locker\_details;

}

public JSONObject allLockerTableData() throws SQLException {

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("SELECT \* from locker");

ResultSet rs = stmt.executeQuery();

JSONArray jsonList = new JSONArray();

while(rs.next()){

JSONObject approvals = new JSONObject();

approvals.put("locker\_id", rs.getInt("id"));

approvals.put("account\_number", rs.getString("customer")==null?false:rs.getString("customer"));

approvals.put("availability",rs.getString("available"));

// approvals.put("customer\_id",rs.getInt("users.id"));

// jsonList.put(approvals);

jsonList.add(approvals);

}

System.out.println("======================================================"+jsonList);

JSONObject data=new JSONObject();

data.put("locker\_info", jsonList);

System.out.println("======================================================"+data);

return data;

}

public Boolean removeCustomerId(int customer\_id) throws SQLException {

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("UPDATE `locker` SET `customer` = NULL, `available`='yes' WHERE `locker`.`customer` = ?;");

stmt.setInt(1, customer\_id);

int rs = stmt.executeUpdate();

return rs==1;

}

public Boolean insertCancellationDataIntoAddLocker\_requestTable(int customer\_id) throws SQLException {

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("UPDATE `addlocker\_request` SET `cancelled\_date` = ? WHERE `addlocker\_request`.`customer` = ?;");

stmt.setString(1, new SimpleDateFormat("yyyy/MM/dd").format(new Date()));

stmt.setInt(2, customer\_id);

int rs = stmt.executeUpdate();

return rs==1;

}

public JSONObject allAppointments() throws SQLException {

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("SELECT users.\*, access\_locker.\* from access\_locker INNER JOIN locker on locker.id=access\_locker.locker\_id inner join users on users.id=locker.customer where access\_locker.permission=0");

ResultSet rs = stmt.executeQuery();

JSONArray jsonList = new JSONArray();

while(rs.next()){

JSONObject approvals = new JSONObject();

approvals.put("access\_id", rs.getInt("access\_locker.id"));

approvals.put("locker\_id", rs.getInt("access\_locker.locker\_id"));

approvals.put("account", rs.getInt("users.id"));

approvals.put("date",rs.getString("access\_locker.date"));

approvals.put("time",rs.getString("access\_locker.time"));

approvals.put("charges",rs.getString("access\_locker.charges"));

jsonList.add(approvals);

}

JSONObject data=new JSONObject();

data.put("appointments", jsonList);

return data;

}

public Boolean setPermission2One(int access\_id) throws SQLException {

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("UPDATE `access\_locker` SET `permission` = 1 WHERE `access\_locker`.`id` = ?");

stmt.setInt(1, access\_id);

int rs\_updation = stmt.executeUpdate();

return rs\_updation==1;

}

public Boolean rejectAppointment(int access\_id) throws SQLException {

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("UPDATE `access\_locker` SET `permission` = 2 WHERE `access\_locker`.`id` = ?");

stmt.setInt(1, access\_id);

int rs\_updation = stmt.executeUpdate();

return rs\_updation==1;

}

public String openAccount(String name, String contact, String address, String openingamount, String accounttype, String password) throws SQLException {

System.out.print("==============================================="+address);

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("insert into users (`username`,`password`,`nature`,`balance`,`name`,`contact`,`address`,`accounttype`,`is\_active`) values(?,?,'customer',?,?,?,?,?,1)");

stmt.setString(1,"");

stmt.setString(2,password);

// stmt.setInt(3,Integer.parseInt(openingamount));

stmt.setString(3, openingamount);

stmt.setString(4,name);

stmt.setString(5,contact);

stmt.setString(6,address);

stmt.setString(7,accounttype);

int rs = stmt.executeUpdate();

if(rs==1){

stmt.close();

stmt = this.conn.prepareStatement("Select id from users where password = ?");

stmt.setString(1, password);

ResultSet result = stmt.executeQuery();

if(result.next()){

String account\_number = Integer.toString(result.getInt("id"));

stmt.close();

stmt = this.conn.prepareStatement("Update users set username = ? where id=?");

stmt.setString(1, account\_number);

stmt.setInt(2, Integer.parseInt(account\_number));

int update\_result = stmt.executeUpdate();

if(update\_result==1){

return account\_number;

}

}

}

return null;

}

public JSONObject allActiveAccounts() throws SQLException {

this.conn = Conn.getMysqlConnection();

PreparedStatement stmt=this.conn.prepareStatement("SELECT \* from users where is\_active=1");

ResultSet rs = stmt.executeQuery();

JSONArray jsonList = new JSONArray();

while(rs.next()){

JSONObject approvals = new JSONObject();

approvals.put("username", rs.getString("username"));

approvals.put("name", rs.getString("name"));

approvals.put("balance", rs.getString("balance"));

approvals.put("address",rs.getString("address"));

approvals.put("accounttype",rs.getString("accounttype"));

approvals.put("contact",rs.getString("contact"));

jsonList.add(approvals);

}

JSONObject data=new JSONObject();

data.put("accounts", jsonList);

System.out.println("=================================================="+data);

return data;

}

/\*\*

\* @return the id

\*/

public int getId() {

return id;

}

/\*\*

\* @param id the id to set

\*/

public void setId(int id) {

this.id = id;

}

/\*\*

\* @return the customer

\*/

public String getCustomer() {

return customer;

}

/\*\*

\* @param customer the customer to set

\*/

public void setCustomer(String customer) {

this.customer = customer;

}

/\*\*

\* @return the request\_date

\*/

public Date getRequest\_date() {

return request\_date;

}

/\*\*

\* @param request\_date the request\_date to set

\*/

public void setRequest\_date(Date request\_date) {

this.request\_date = request\_date;

}

/\*\*

\* @return the approved\_date

\*/

public Date getApproved\_date() {

return approved\_date;

}

/\*\*

\* @param approved\_date the approved\_date to set

\*/

public void setApproved\_date(Date approved\_date) {

this.approved\_date = approved\_date;

}

/\*\*

\* @return the cancellation\_date

\*/

public Date getCancellation\_date() {

return cancellation\_date;

}

/\*\*

\* @param cancellation\_date the cancellation\_date to set

\*/

public void setCancellation\_date(Date cancellation\_date) {

this.cancellation\_date = cancellation\_date;

}

}

* + - 1. **Mysql code :**

**Miniproject.sql:**

-- phpMyAdmin SQL Dump

-- version 4.8.3

-- https://www.phpmyadmin.net/

--

-- Host: 127.0.0.1

-- Generation Time: Mar 26, 2019 at 05:31 PM

-- Server version: 10.1.35-MariaDB

-- PHP Version: 7.2.9

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!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 utf8mb4 \*/;

--

-- Database: `miniproject`

--

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

--

-- Table structure for table `access\_locker`

--

CREATE TABLE `access\_locker` (

`id` int(11) NOT NULL,

`locker\_id` int(11) NOT NULL,

`date` date NOT NULL,

`time` time NOT NULL,

`permission` int(1) NOT NULL,

`charges` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

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

--

-- Table structure for table `addlocker\_request`

--

CREATE TABLE `addlocker\_request` (

`id` int(11) NOT NULL,

`customer` int(11) NOT NULL,

`request\_date` date DEFAULT NULL,

`approved\_date` date DEFAULT NULL,

`expiry\_date` date DEFAULT NULL,

`cancelled\_date` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Dumping data for table `addlocker\_request`

--

INSERT INTO `addlocker\_request` (`id`, `customer`, `request\_date`, `approved\_date`, `expiry\_date`, `cancelled\_date`) VALUES

(1, 152901, '2019-03-24', '2019-03-24', '2020-03-24', NULL),

(2, 152903, '2019-03-24', '2019-03-24', '2020-03-24', NULL);

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

--

-- Table structure for table `locker`

--

CREATE TABLE `locker` (

`id` int(11) NOT NULL,

`customer` int(11) DEFAULT NULL,

`locker\_key` int(15) NOT NULL,

`available` varchar(20) NOT NULL DEFAULT 'yes'

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Dumping data for table `locker`

--

INSERT INTO `locker` (`id`, `customer`, `locker\_key`, `available`) VALUES

(2, 152901, 10002, 'no'),

(14524, 152903, 10001, 'no'),

(14526, NULL, 10003, 'yes');

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

--

-- Table structure for table `users`

--

CREATE TABLE `users` (

`id` int(30) NOT NULL,

`username` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

`nature` varchar(255) NOT NULL,

`balance` int(11) NOT NULL,

`name` varchar(255) NOT NULL,

`contact` varchar(18) NOT NULL,

`address` text NOT NULL,

`accounttype` varchar(255) NOT NULL,

`is\_active` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Dumping data for table `users`

--

INSERT INTO `users` (`id`, `username`, `password`, `nature`, `balance`, `name`, `contact`, `address`, `accounttype`, `is\_active`) VALUES

(152900, '152900', 'Deepak@123', 'customer', 13999, 'Deepak', '8377083727', '923', 'Saving', 1),

(152901, '152901', 'Kaju@123', 'customer', 50000, 'Kajal', '8377083727', 'South Extension', 'Saving', 1),

(152902, 'banker', 'banker', 'banker', 0, '', '', '', '', 0),

(152903, '152903', 'Shakir@123', 'customer', 11730, 'Shakir', '7501452484', 'Batla House', 'Saving', 1);

--

-- Indexes for dumped tables

--

--

-- Indexes for table `access\_locker`

--

ALTER TABLE `access\_locker`

ADD PRIMARY KEY (`id`),

ADD KEY `locker\_id\_2` (`locker\_id`);

--

-- Indexes for table `addlocker\_request`

--

ALTER TABLE `addlocker\_request`

ADD PRIMARY KEY (`id`),

ADD KEY `customer\_3` (`customer`);

--

-- Indexes for table `locker`

--

ALTER TABLE `locker`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `customer` (`customer`);

--

-- Indexes for table `users`

--

ALTER TABLE `users`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `username` (`username`);

--

-- AUTO\_INCREMENT for dumped tables

--

--

-- AUTO\_INCREMENT for table `access\_locker`

--

ALTER TABLE `access\_locker`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT;

--

-- AUTO\_INCREMENT for table `addlocker\_request`

--

ALTER TABLE `addlocker\_request`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT for table `locker`

--

ALTER TABLE `locker`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=14527;

--

-- AUTO\_INCREMENT for table `users`

--

ALTER TABLE `users`

MODIFY `id` int(30) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=152904;

--

-- Constraints for dumped tables

--

--

-- Constraints for table `access\_locker`

--

ALTER TABLE `access\_locker`

ADD CONSTRAINT `access\_locker\_ibfk\_1` FOREIGN KEY (`locker\_id`) REFERENCES `locker` (`id`);

--

-- Constraints for table `addlocker\_request`

--

ALTER TABLE `addlocker\_request`

ADD CONSTRAINT `addlocker\_request\_ibfk\_1` FOREIGN KEY (`customer`) REFERENCES `users` (`id`);

--

-- Constraints for table `locker`

--

ALTER TABLE `locker`

ADD CONSTRAINT `locker\_ibfk\_1` FOREIGN KEY (`customer`) REFERENCES `users` (`id`);

COMMIT;

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

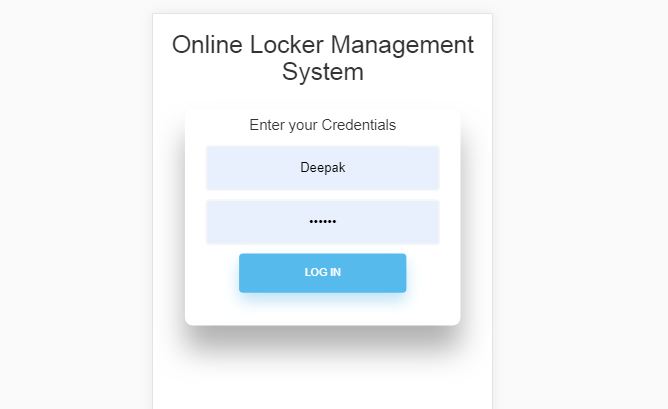
/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

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

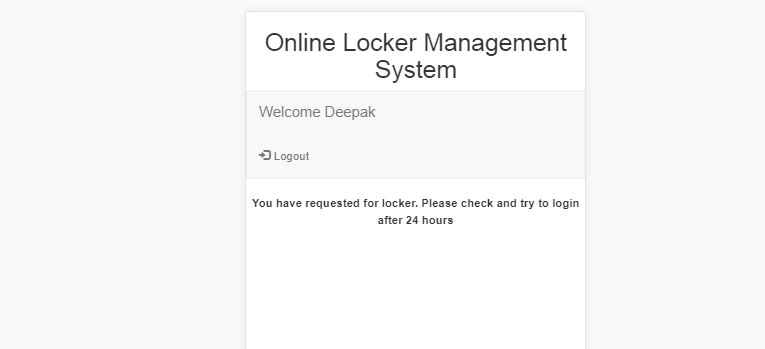
* + - 1. **Output Screens**

**Customer side :**

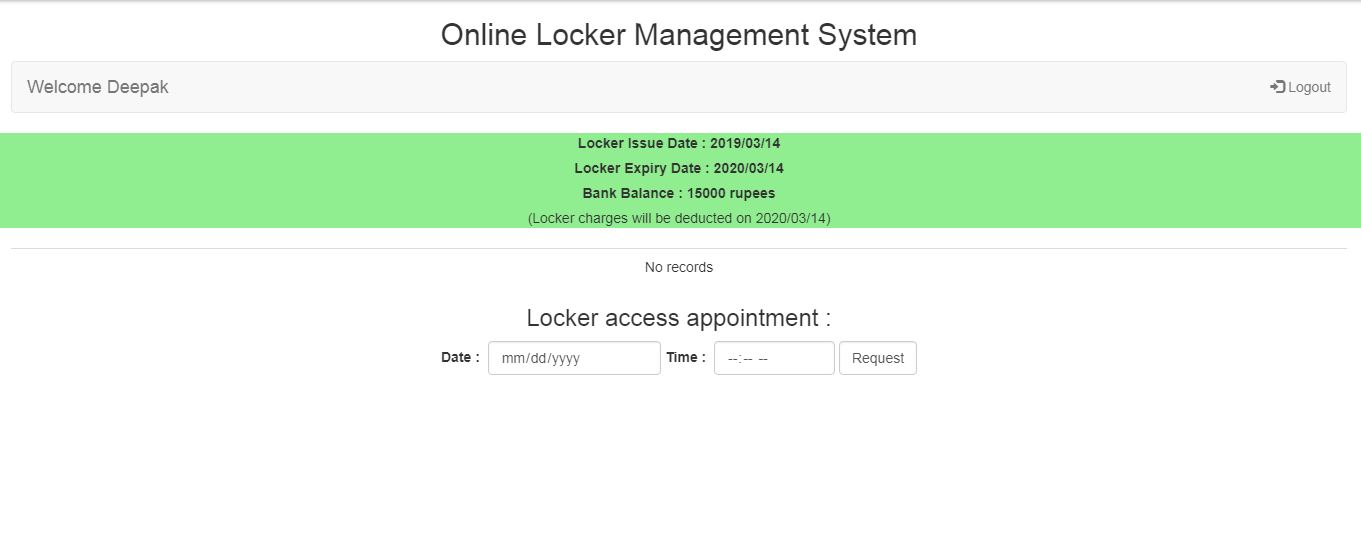
**Login form:**

****

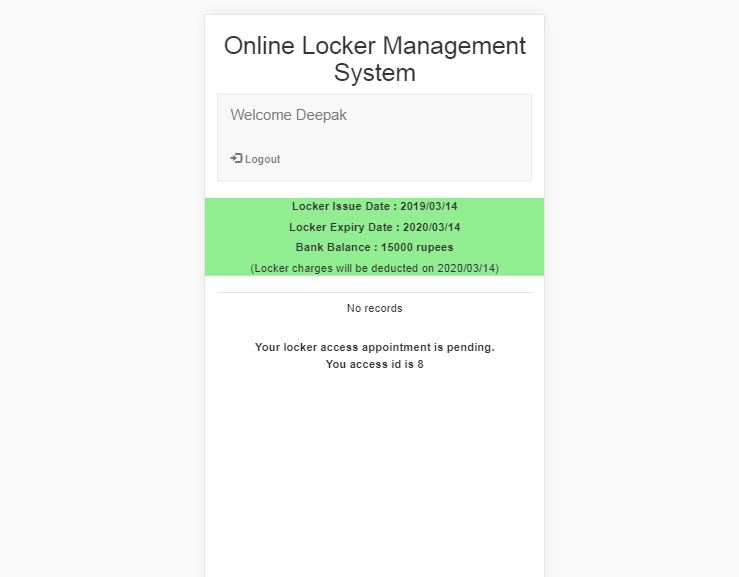
**Locker Application pending:**

****

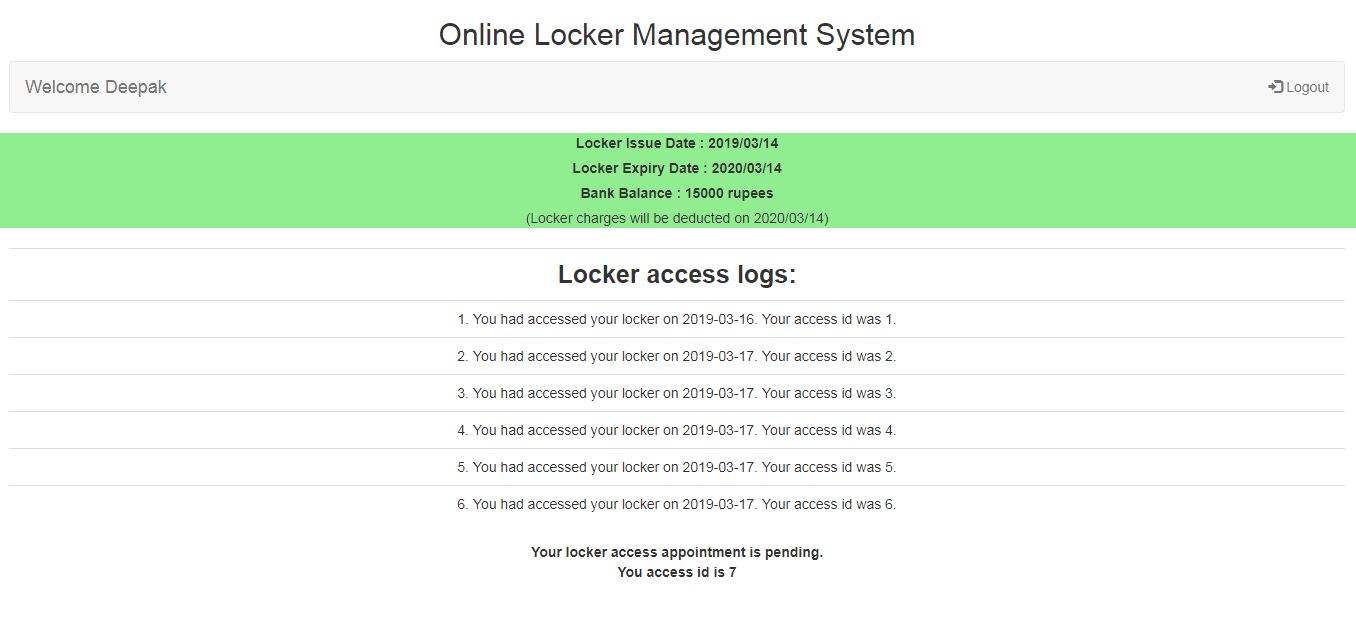
**Appointment Screen:**

****

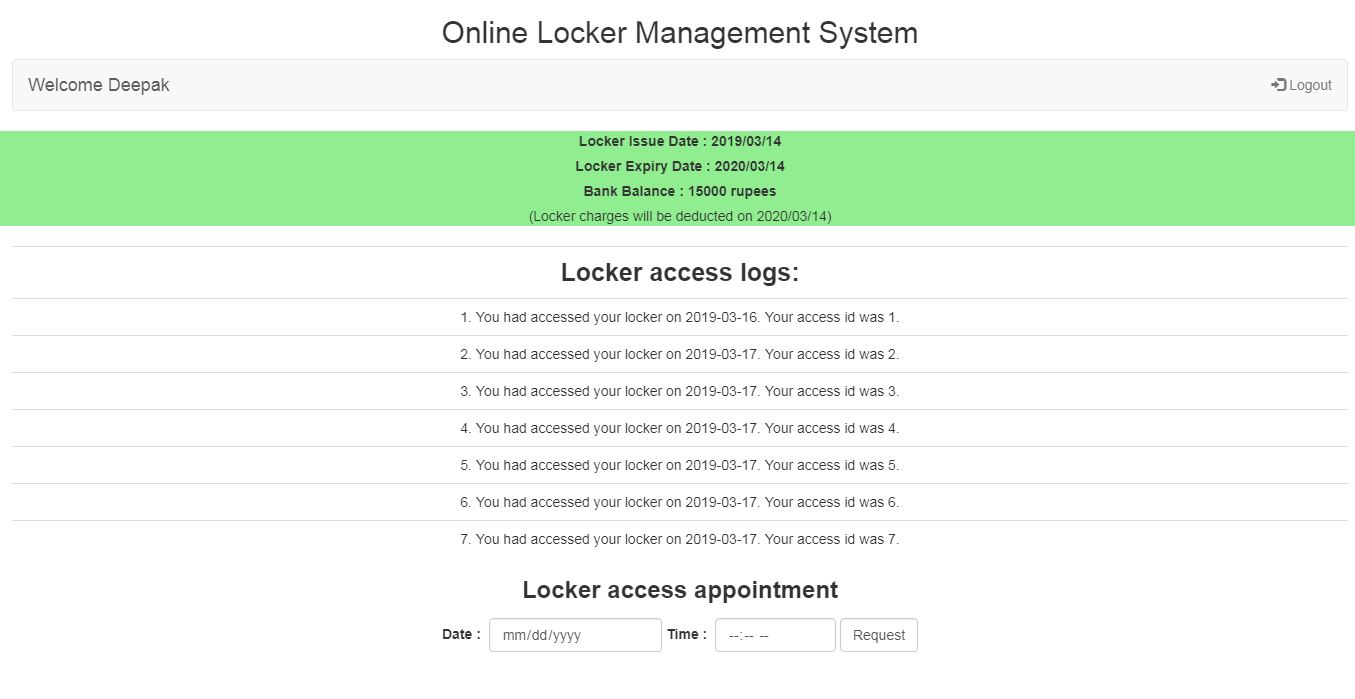
**Appointment Pending:**

****

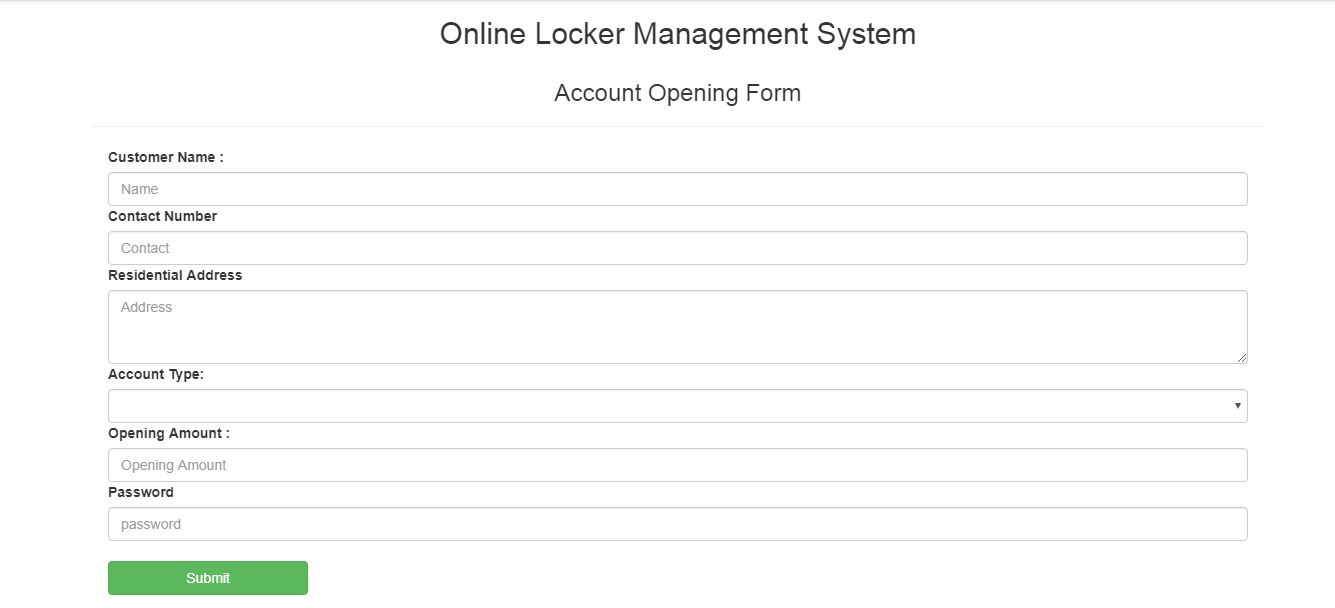
**Appointment history with pending request:**

****

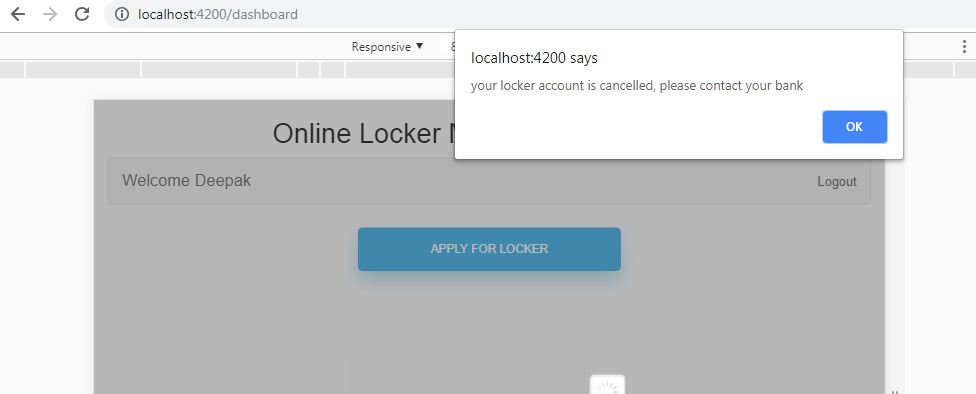
**Appointment history with new request form :**

****

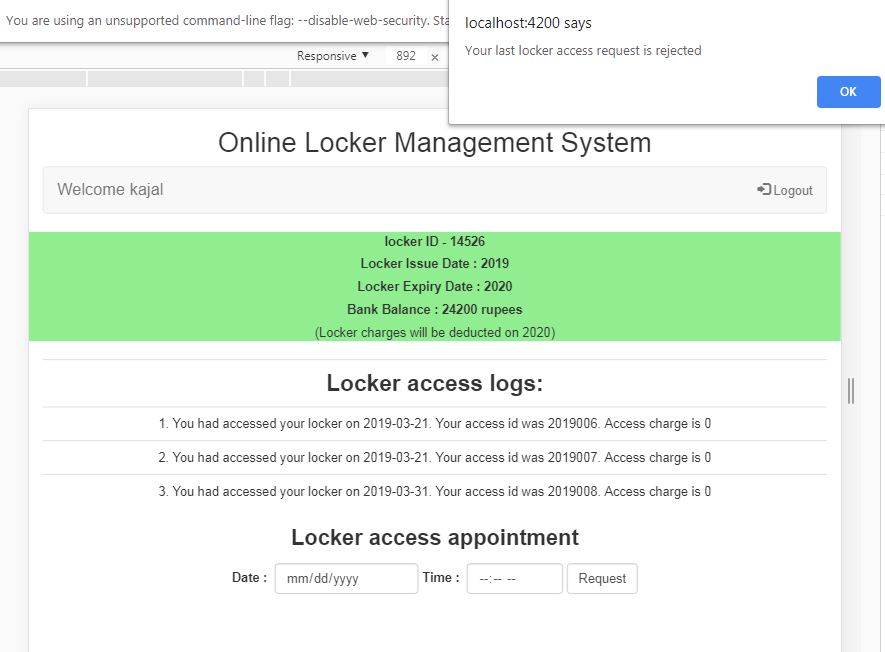
**Account opening form :**

****

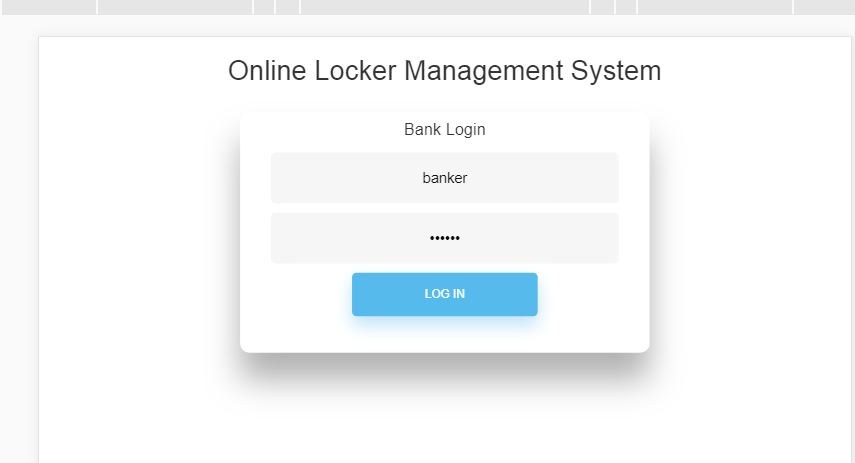
**When locker is cancelled by bank :**

****

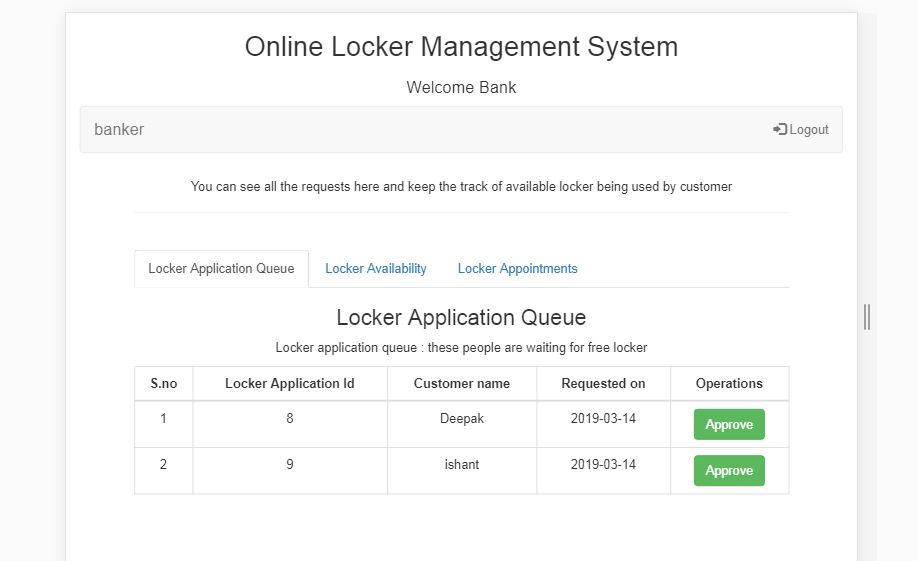
**When appointment is rejected by bank :**

****

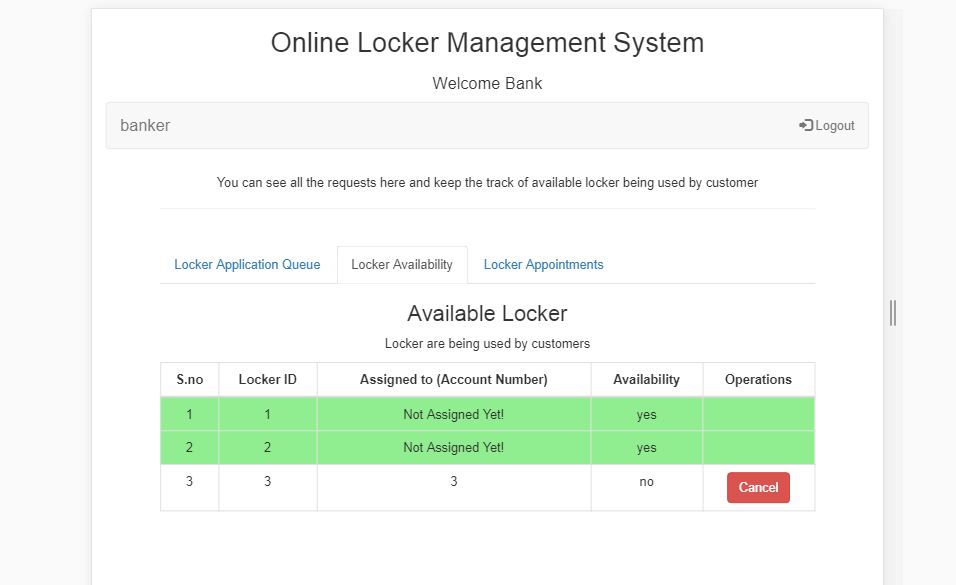
**Bank login :**

****

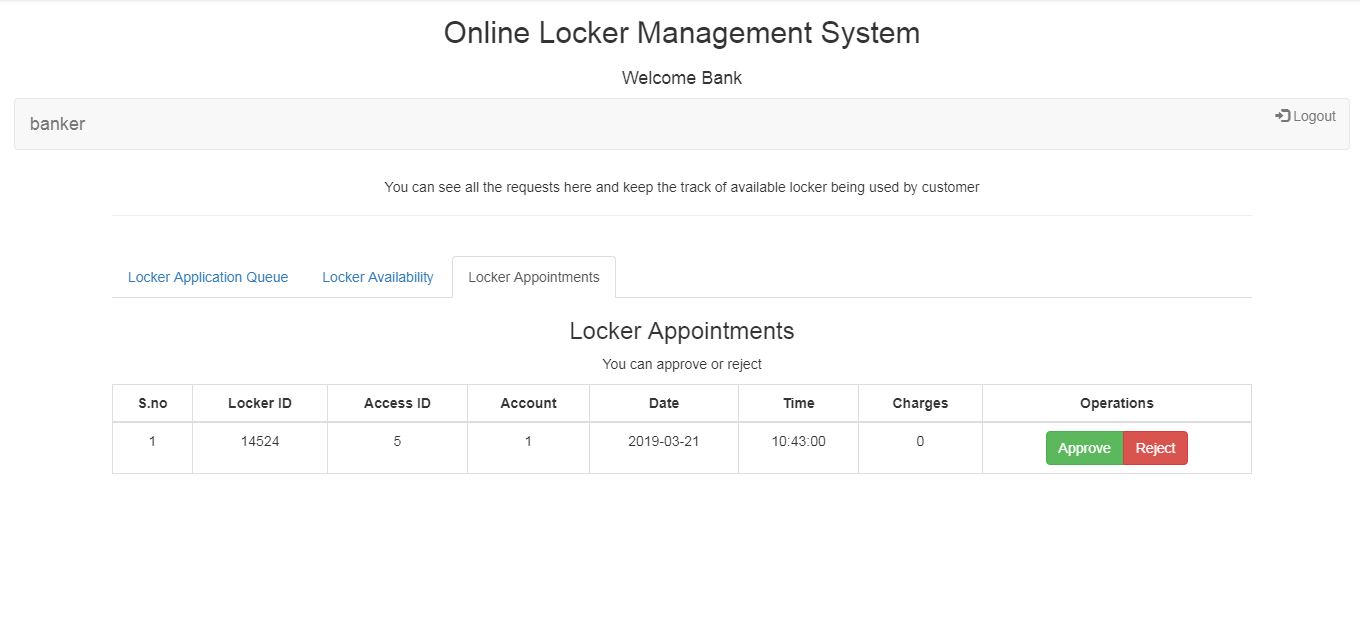
**Locker application queue :**

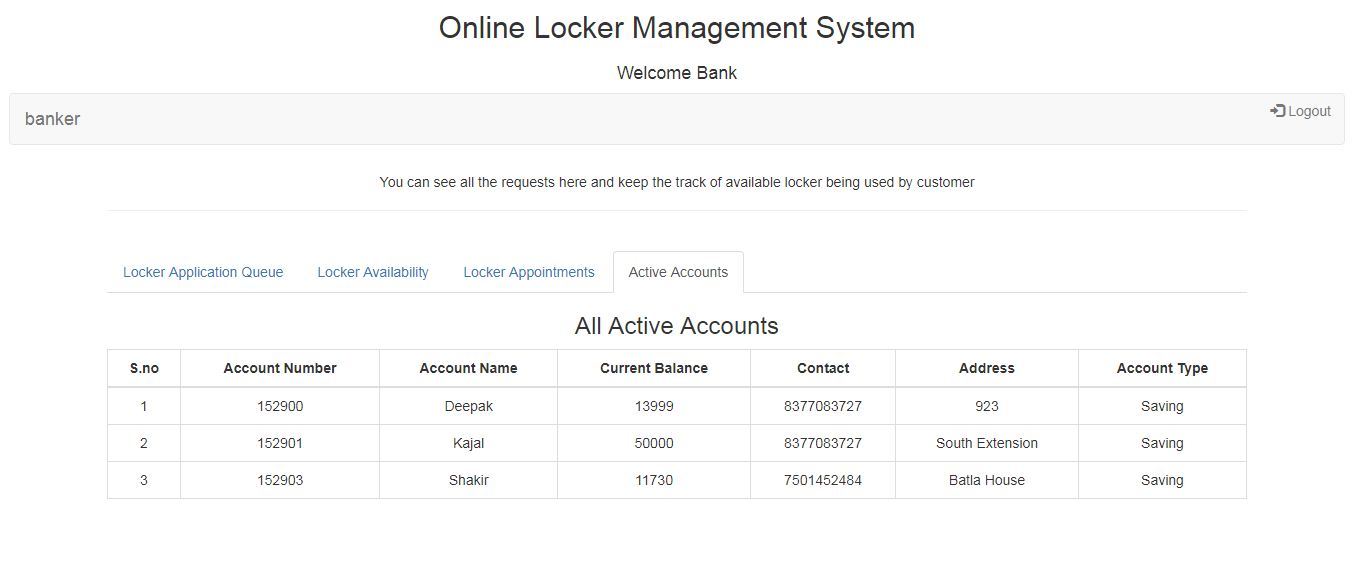
****

**Locker availability and occupied status :**

****

**Locker appointment from bank side :**

****

**Account Information:**

* 1. **Testing Approach**
     1. **Unit Testing**

Unit testing is a process to test group of codes, in order to satisfy customer’s requirement. This is also known as module testing process. Positive as well as negative test cases are designed according to the functional requirement of the system. This covers following topics such as field validation and recording the behavior of the code. Corresponding outputs, both expected and actual outputs are recorded. Further actions need to be taken to prevent runtime exceptions using concepts like assertion, etc.

* + 1. **Integrated Testing**

Integration testing is a level of software testing where individual and coupled modules are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

1. **RESULTS AND DISCUSSION**
   1. **Test Reports**

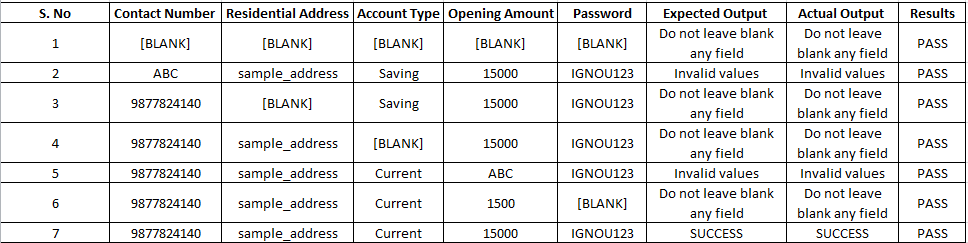
**Login form :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. no** | **User ID** | **Password** | **Expected Output** | **Actual Output** | **Results** |
| **1.** | **[Blank]** | **[Blank]** | **[Label] - Enter your credentials** | **[Label] - Enter your credentials** | **PASS** |
| **2.** | **[Blank]** | **Sample\_Password** | **[Label] - Enter your credentials** | **[Label] - Enter your credentials** | **PASS** |
| **3.** | **Sample\_Username** | **[Blank]** | **[Label] - Enter your credentials** | **[Label] - Enter your credentials** | **PASS** |
| **4.** | **[Incorrect username]** | **[Incorrect password]** | **[Label] – Invalid credentials** | **[Label] – Invalid credentials** | **PASS** |
| **5.** | **[Incorrect username]** | **[correct password]** | **[Label] – Invalid credentials** | **[Label] – Invalid credentials** | **PASS** |
| **6.** | **[correct username]** | **[Incorrect password]** | **[Label] – Invalid credentials** | **[Label] – Invalid credentials** | **PASS** |
| **7.** | **[correct username]** | **[correct password]** | **Logged in** | **Logged in** | **PASS** |

**Appointment Form**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No** | **Date** | **Time** | **Expected Output** | **Actual Output** | **Results** |
| 1 | [BLANK] | [BLANK] | Do not leave blank any field | Do not leave blank any field | PASS |
| 2 | Date | [BLANK] | Do not leave blank any field | Do not leave blank any field | PASS |
| 3 | [BLANK] | Time | Do not leave blank any field | Do not leave blank any field | PASS |
| 4 | Date | Time | SUCCESS | SUCCESS | PASS |

**Account Opening Form:**



1. **CONCLUSIONS**
   1. **Conclusion**

The web application provides facilities to conduct online locker management worldwide.

It saves time as it allows a number of customers to apply for locker at a time and displays the results as the lockers request is accepted or rejected, no need to wait for the result because it is automatically generated by the server. Administrator has the privilege to create, modify and delete the locker details and its particular users. Users can register, login and apply for the bank locker with her/his specific ID and can see the results as well.

* 1. **Enhancements of the System**

With upcoming programming approaches, design patterns and technologies, it will be enhanced to the next level wherein more functionality will be added. Upcoming functionality will secure the system to the next level and it will be more user friendly for ease of bank locker operations. Upcoming functionalities will be following:

1. Cancelled lockers records
2. Cancelled appointment records
3. Activities recorders
4. Appointment on bank credit.
   1. **Future Scope of the Project**

The future of the project will be useful because with this system many other industries can also use it, such as hospital: for employee securities, schools, banks, post office, etc.

**REFERENCES**

<https://www.javatpoint.com/>

<https://angular.io/>

<https://www.vogella.com/tutorials/REST/article.html>

Complete Reference JAVA