**DEVELOPMENT OF A CUSTOMER RELATION,INTERATION AND COMPLAIN MANAGEMENT**

**BY**

**NKENWOKENEME MAXWELL**

Registration No:

**U14/NAS/CSC/094**

Being a Bsc project report submitted in partial fulfillment of the requirements for the award of a Bachelor’s Degree in Computer Science of the Godfrey Okoye University.

**Supervisor: Mr. Benson Ikenna**

**Department of Computer Science/Mathematics**

**Godfrey Okoye University,Enugu.**

**July 2018**

**CERTIFICATION**

I hereby declare that the work presented herein was done by me, and not by a third party , should I be convicted of having cheated in this work, I shall accept

the verdict of the university.

Nkenwokeneme Maxwell

**APPROVAL**

This certify that this project work was carried out by Nkenwokeneme Maxwell in the Department of Computer Science, Godfrey Okoye University, in partial fulfillment of the requirements for the award of bachelor of the science in computer science.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**-------------------- Date**

**Mr. Benson Ikennna**

**(Project Supervisor)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**------------------------- Date**

**DR.(Mrs).Monica Agu**

## (Head of Department)

## DEDICATION

## This project work is dedicated to the Almighty God, for is wisdom and understanding ,also to my loving sister Rev.sis. Judith Nkenewokeneme and to Gloria Mbaliri who has always cared about me, inspired me and always motivating me to complete my project.

**ACKNOWLEDGEMENT**

Initially, I should firstly convey my feeling of thankfulness to Godfrey Okoye University by giving me the opportunity to gain more knowledge from such a reputable university and respected Professors, especially in the field of Computer Science. In this acknowledgement, I also would like to express distinctively my gratitude to parties underneath the entire process of finishing this project as following:

Mr. Benson, as my project Supervisor for giving me such outstanding discussions, comments and contributions for the perfection of the project;

Last but not least, all my friends whom we spent 4 (four) years together in the spirit of friendships and brotherhoods for all the time we shared, discussions, debates, critiques, and inputs; both during the joyful and the saddest moments. To close this acknowledgement, I do hope that this project would be advantageous to enrich the knowledge in the field of Computer science and any countries in the world.

**Abstract**

The Issue Tracker system is web based application and it is designed to keep track of complaints registered by the college clients and staffs, so this system need to have distributed platform independent web application. In case registration it should be open and assigned to technical and functional staff can update the issue status to closed. The methodology used is the object oriented model. The task of Administrator executives can control all the activities in the system, for creating issue using case registration, assign to technical staff and check the service performance. The web application was developed using JAVA, HTML, JAVASCRIPT, MySQL, CSS

**TABLE OF CONTENTS**

1.0 Background of study 1-2

1.1 Statement of Problem 2

1.2 Objective 3

1.3 Significance of the Project 3

Chapter 2: Literature Review

2.0 Introduction 4

2.1 Theoretical Background 4-5

2.2 Review of Related Literature 6-8

2.3 Summary 8

Chapter 3: System Analysis and Design

3.0 Introduction 9-10

3.1 Describe the existing system 11

3.2 Analysis of the proposed system 11-13

3.3 Design of the proposed system 14-17

Chapter 4: System Implementation

4.0 Introduction 19

4.1 Choice of development Environment 19

4.2 Implementation Architecture 20

4.3 Software Testing 20-26  
4.4 Documentation 26

4.4.1 User Manuel 26

4.4.2 Source code listing 27

**Chapter 5: Summary and Conclusion**

5.0 Summary 28

5.1 Conclusion 28

5.2 Recommendation 29

**References**

**Appendices**

**LIST OF FIGURES**

Fig1.0 Key Component of E-CRM - - - 6

Fig3.1 Use case Diagram of the propose system - - - 13

Fig 4.0Implementation Architecture - - - 20

Fig 4.1Snapshot of the login panel - - - 22

Fig 4.2Snapshot of adding product - - - 23

Fig 4.3Snapshot of the manage case - - - 23

Fig 4.4Snapshot of manage client - - - 24

Fig 4.5Snapshot of adding client - - - 24

Fig 4.6Snapshot of the DashBoard - - - 25

**LIST OF TABLE**

Table 3.1 User table of the database - - - - 15

Table 3.2 Product table of the database - - - - 16

Table 3.3 Client table of the database - - - - 16

Table 3.4 O.E.M table of the database - - - - 17

Table 3.5 Staff table of the database - - - - 17

**CHAPTER ONE**

**INTRODUCTION**

1. **BACKGrOUND OF STUDY**

Most times, customers have made complains to company under which they have been working with and the complaint have not been heard or look into for solution, or have complained and have not been directed to the right channel And at that process the issue may be left unsolved. This make customers angry, frustrated and wanting to stop doing business or buying or partnering with the company. Customers Happiness are the first priority of Any company, so this web application was created to help company look into rising issues and solve them as soon as possible .

This web application was created so as to help resolve complaints from customers ,and also follow up on the issue if it resolved or not. This project illustrates the development of a issue tracker application.

Issue Tracker is an application that allows customers using a company’s product, to make there complains through the application over the internet. A tracking application that typically runs on any browser on the computer where your Web site is located, allows customers to make complains from their homes, offices and were they find themselves as long as access to the internet is made available. Issue Tracker application will be able to notify complain and the complaint will be sent to dash board, were it will be indicated what section the issue will be put in. After the issue has been delivered to any section of your choice it will be processed and a reply will be sent.

The Manual form of making a complain has not been very reliable over the ages and no comprehensive reply or feedback nor those help in solving the issue at when due.

Issue tracking system helps the a company to look into rising issues, determine potential problems in the school .

With the advancement in message sending & saving and record retrieval system, the company cannot afford to be ignorance of the basic software tool, which is the driving force behind technological oriented administration. The university can achieve much with a well inform, tracking and well defined database system . The management of institutions is done through network technology, where all the systems of information management has been digitized. All these innovations have the aim to simplify life by making a lot of things ease and efficient.

**1.1 STATEMENT OF THE PROBLEM**

My research is aimed at seeing ways of overcoming the problem from customers with conventional complaint management system.

These include;

1. Incomprehensive complaints history.
2. Inconsistency in customer interaction.
3. Lack of prompt updating as to when a complaint issues has been resolved.
4. Poor performance of the manual system may lead into the missing or exploitative of the complaint by the staff or any member of the management,

**1.2 OBJECTIVES**

This project is aimed at how company can resolve rising issues and improve the efficiency. Thisweb application involves almost all the sections of staff in the company; the future implementation will be online help for the customers .

* Allow customer to registered and obtain username and password to login into the system and lodge in their complain and view previous complain.
* To implement a new system called issueTracker. With the use of HTML, JAVA and MYSQL.
* Rapid responses to customer issues
* Staff receiving emails and notification for rising issues
* Customer getting quick response

**1.3 SIGNIFICANCE OF THE PROBLEM**

The significance of this study is to serve better than the existing system which is highly manual and therefore difficult in terms of monitoring the complaint, it improve database and enhance effectiveness, efficiency and security of the system. It is also intended that the study will help in the development of a new and hopefully and standard better computer aided system.

The new system will save time, reduce improper handling of complaint system and also improve relationship between customer, staff and management.

The system is expected to be easy as customer can login their complaint anytime, staff and management also can equally response to online complaint in a more easy way and reduced turnaround time in responding to customers issues.

**CHAPTER TWO**

**LITERATURE REVIEW**

**2.0 INTRODUCTION**

The purpose of this part is to provide a brief description about terms that are used during development of this project. It deals with theoretical concepts and fundamentals that support this project. It provides definitions and characteristics of technologies used. Since theoretical concepts involved are complex the description provides only the over view.

**2.1** **THEORETICAL BACKGROUND**

The Technologies used in this work are listed below

1. JAVA
2. JAVA script
3. Hypertext Markup Language
4. ORACLE database
5. Bootstraps

* **JAVA:** is a general-purpose [computer-programming language](https://en.wikipedia.org/wiki/Programming_language) that is [concurrent](https://en.wikipedia.org/wiki/Concurrent_computing), [class-based](https://en.wikipedia.org/wiki/Class-based_programming), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "[write once, run anywhere](https://en.wikipedia.org/wiki/Write_once,_run_anywhere)" (WORA), meaning that [compiled](https://en.wikipedia.org/wiki/Compiler) Java code can run on all platforms that support Java without the need for recompilation.[[17]](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_note-design_goals-17) Java applications are typically compiled to [bytecode](https://en.wikipedia.org/wiki/Java_bytecode) that can run on any [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) regardless of [computer architecture](https://en.wikipedia.org/wiki/Computer_architecture).
* **JAVASCRIPT:** JavaScript is a client-side script, used to control a web page and buttons at the client, it also used to make webpages interactive. It supports event-driven, functional, and imperative programming style.

**<script type="text/javascript">**

**alert("Check your mail! If you don't see any messages, repeat the process!");**

**window.location = 'index.php';**

**</script>**

* **HTML:** HTML is the hyper-text markup language which is the basic language used in the construction of websites all over the world. It was invented in the 1990’s by the W3C (World Wide Web consortium) as the standard language for setting up any application that runs on the internet or on the web. The basic syntax for HTML is given below;

**<html>**

**<head>**

**<title> My Title is Here </title>**

**</head>**

**<body>**

**<label> This is HTML </label>**

**<h1> topic </h1>**

**</body>**

**</html>**

* **CSS**: It stands for Cascading Style Sheet. Style sheet refers to the document itself. Style sheets have been used for document design for years. They are the technical specifications for a layout, whether print or online. Print designers use style sheets to insure that their designs are printed exactly to specifications.

**Html selector {**

**Property: value;**

**}**

**Example:**

**div[id = 'header'] {**

**background-color: grey;**

**width: cover;**

**height: 10%;**

**padding: 0.5%;**

**}**

|  |
| --- |
|  |

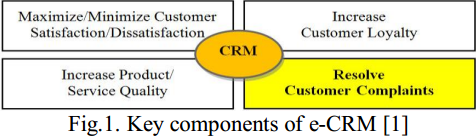
* **padding: 0.5%;**
* **}**

**2.2 Review of Relevant Literature**

Out of the previous related work done concerning customer complaint, the most recent research. [6] develop a new complaint management system called (e-Aduan) as a platform for UiTM Pahang’s customers to complaint and comment regarding the services and facilities provided by the university. The researcher found out that the most appropriate to the research topic handling customer complaint using SOA . [5] tried to improve relation between Citizens and Government by presenting a new model based on Service Oriented Architecture (SOA). With utilizing the presented model in Government body on one hand Governments will have the ability to minimize Citizens' dissatisfaction and on the other hand it can encourage Citizens to participate in controlling Government body such as Governments' staffs and organizations.

**E-CRM**

Electronic Customer Relationship Management (e-CRM) is gaining the attention of e-business managers who are interested in increasing repeat business and customer loyalty[4] . [1] CRM has four key components, as shown:



Complaint information should be shared among departments within a company and even other companies through the supply chain. Best-Practice companies have realized that using CRM strategy solves expected problems in implementation as managing all complaint information and defining relations between different complaints.

**E-Complaint**  
Each organization has its own definition for complaint. They define complaint related to the services they provide for users. Hence definitions are different because of the variety of services among the organizations [5]. Customer complaining behavior defined as the consequences of customer dissatisfaction [7], it has long been considered an important form of market feedback [2]. On other words, Customer Complaints Management is becoming a critical key success factor in today’s business environment. Complaint Management System is a system that can survey customer feedback about any organization. Best-practice organizations consider complaints as opportunities for improvement. These companies understand the link between complaint resolution and customer loyalty and work hard to act immediately on problems that can be easily resolved. In addition, the researcher believes that either Government or any system needs user feedback to find out the challenges after running new system. User feedback can be considered as complaints and suggestions, which can be instrumented in the organization to improve its services and products.

**Service-oriented architecture (SOA)**

Service-oriented architecture (SOA) is a style of developing and integrating software [3]. It is designing system with services instead of procedures [5]. It involves breaking an application down into common, repeatable “services” that can be used by other applications, both internal and external, in an organization – independent of the applications and computing platforms on which the business and its partners rely [3]. Therefore it helps system to be more flexible to change. With using SOA the costs of change will be decreased enormously since administrator does not need to change the whole system and they just need to change or add the service which system needed. There are many definitions for the SOA but the researcher deeps all definitions are same in concept. In SOA base systems we have different systems with different services which have to collaborate with each other through a common bus and unique data type although each system has its own data type [5]. The researcher believes that e-complaint based on SOA talks about designing a system based on different services beyond a common interface.

**2.3 Summary**

The study is focused on the tracking issues or complaint from customer in an organization or company, through the design and development of a system providing a means of information between the company and affiliated customers. This system will be able to handle complaints by recording and giving feedback for each raised complaint.

This web service is designed to provide the various services to the clients it uses the web server and application server. Server receives the various requests from the client and the server has to respond the client’s request.

Chapter 3:

System Analysis And Design

1. **Introduction**

The analysis of the proposed system for the issue tracking web application will be the main focus of attention in this particular chapter. The system Analysis of the existing system to be improved upon and the proposed solutions to the short comings of the current mode of web application Issue Tracker will be thoroughly discussed. Also the methodology used is the object oriented methodology. This methodology involves four stages

* [System Analysis](https://www.freetutes.com/systemanalysis/sa2-object-oriented-methodology.html#system-analysis)
* [System Design](https://www.freetutes.com/systemanalysis/sa2-object-oriented-methodology.html#system-design)
* [Object Design](https://www.freetutes.com/systemanalysis/sa2-object-oriented-methodology.html#object-design)
* [Implementation](https://www.freetutes.com/systemanalysis/sa2-object-oriented-methodology.html#implementation)

## System Analysis

system analysis is the first phase of development in case of Object Modeling too. In this phase, the developer interacts with the user of the system to find out the user requirements and analyses the system to understand the functioning.

## System Design

System Design is the next development stage where the overall architecture of the desired system is decided. The system is organized as a set of sub systems interacting with each other.

## Object Design

In this phase, the details of the system analysis and system design are implemented. The Objects identified in the system design phase are designed. Here the implementation of these objects is decided as the data structures get defined and also the interrelationships between the objects are defined. This concept is known as creating a class.

**Class**: A class is a collection of similar objects. It is a template where certain basic characteristics of a set of objects are defined. The class defines the basic attributes and the operations of the objects of that type.

**Abstraction**: Classes are built on the basis of abstraction, where a set of similar objects are observed and their common characteristics are listed.

**Inheritance**: Inheritance is another important concept in this regard. This concept is used to apply the idea of reusability of the objects.

## Implementation

During this phase, the class objects and the interrelationships of these classes are translated and actually coded using the programming language decided upon. The databases are made and the complete system is given a functional shape.

**3.1** **Description of the existing system**

The existing system lacks some important functional units for instant message delivery, chat box, dashboard, and storage of customers and staffs data and easy retrieval, which recorded in book. Sending the complaint to the wrong department it leads to unsolved or forgotten issue. This is why it is very crucial to study and understand the existing system properly, its problems and lapses, its inefficiencies and inability.

During the course of the analysis, the following problems were discovered:

* When an issue comes up, it those not go to the right department that it meant for, this lead to the delay to the issue.
* No dashboard to keep reminding them of unsolved problems . Some issues that need a degree of discretion may be forgotten.
* The cost of calling and text is always very high.
* There is no email or message sent to the customer that the issue has been solve

**3.2 Analysis of the Proposed System**

The proposed application was meant to overcome every error of the existing system, meet the demand of the customer/user. Every user will have its own special login details sent to them by E-mail, for them to drop there complains on a particular product. The data needed from the owner of the company are company logo, company name, company address, city, state, company phone number, company e-mail address, which will be generated automatic to ensure authenticity of the web application.

The System is now been able to send mails ,save messages , complains going to the right the source ,customer getting feedback immediately ,section of staffs have been divided into different parts and there can be a flow of chat in the system .

The system can also be used to edit, update, and delete to ensure that the data recorded in database are error free.

**USECASE DIAGRAM**

In [software](http://en.wikipedia.org/wiki/Software_engineering) and [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering), a **use case** is a list of steps, typically defining interactions between a role (known in [UML](http://en.wikipedia.org/wiki/Unified_Modeling_Language) as an "[actor](http://en.wikipedia.org/wiki/Actor_%28UML%29)") and a system, to achieve a goal. Below is the use case diagram of the Issue Tracker system.

**3.3 Design of the Proposed System**

The Online Issue Tracker Website will be implemented using three major components which are a database server for information storage, a middleware application and a client side application. The client side will be designed using HTML (Hypertext Markup Language) and will be viewed with a web browser. The records and information about the product will be stored using the ORACLE DATABASE online database server while the middleware application will be implemented using JAVA programming language.

**Database Design**

The database management system used in this research work is ORACLE DBFORG DATABASE. This is open source relational database management system that uses structured query language.

The required tables in the database include the following:

* User
* Product
* Client
* Original enterprise manufacturer
* Staff

**Table.3.1**

**Users Table**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Field | Data type | Size | null | Description | Action | Extra |
|  | User\_id | Int | 8 |  | Unique id | Primary key | Auto increment |
|  | First\_Nm | Varchar | 34 |  | User name |  |  |
|  | Last\_Nm | Varchar | 34 |  | User surname |  |  |
|  | E-mail | Varchar | 25 |  | User email |  |  |
|  | Role | Varchar | 6 |  | Staff/client |  |  |
|  | Password | Varchar | 45 |  |  |  |  |
|  | Status | Varchar | 5 |  | Active /inactive |  |  |
|  | Create\_date | Time stamp | : |  | Current date |  |  |

**Table .3.2 Product table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | null | Description | Action | Extra |
| Product\_id | Int | 8 |  | Unique id | Primary key | Auto increment |
| Product\_Nm | Varchar | 34 |  | Name |  |  |
| Oem\_id | Int | 8 |  | Unique id | Primary key |  |
| Create\_date | Time stamp | 0:0 |  | Current date |  |  |
| Status | Varchar | 5 |  | Active/inactive |  |  |

**Table 3.3.Client table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Datatype | Size | null | Description | Action | Extra |
| Client\_Id | Int | 8 |  | Number | Primary key | Auto increment |
| Client\_Nm | Varchar | 34 |  | Name |  |  |
| E-mail | Varchar | 34 |  | Client emai |  |  |
| Phone\_No | Int | 12 |  |  |  |  |
| Create\_Date | Time stamp | 0:0 |  | current |  |  |
| Case\_No | Int | 8 |  | Number |  |  |

**Table.3.4.Original Enterprise Manufacturer (OEM) TABLE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Datatype | Size | null | Description | Action | Extra |
| OEM\_Id | Int | 8 |  | Unique key | Primary key | Increment |
| OEM\_Nms | Varchar | 34 |  | Oem name |  |  |
| E-mail | Varchar | 34 |  | Oem email |  |  |
| Phone-No | Int | 12 |  | number |  |  |
| Create\_Date | Time st | 0:0 |  | current |  |  |
| Status | Varchar | 5 |  | Active /inactive |  |  |

**Table .3.5. Staff Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Datatype | Size | null | Description | Action | Extra |
| User\_id | Int | 8 |  | Unque key | Primary key | Auto increment |
| first\_Nm | varchar | 34 |  | Staff name |  |  |
| Last\_Nm | varchar | 34 |  | Surname |  |  |
| E-mail | varchar | 34 |  | Staff e-mail |  |  |
| Role | varchar | 8 |  | Technical/Funtion |  |  |

**System Architecture**

The system architecture is the conceptual model that defines the behavior and and the design of a system. The system is run using database correlation. Most operations carried out using the web application require one or more queries to the database. The following rules are observed;

* The user must login to the website to see information related to his complains.
* The user will have access to some particular fields .
* The administrator must also have an account with the application for him to access information about the customers.
* All administrators’ have a unique password that acts as a passkey and serves as an identity.
* The dashboard is accessible by all, who have login details.

**CHAPTER FOUR**

**System implementation**

**4.0 Introduction**

This chapter is the part that puts a planned system into action and examine in details the analysis and design of the Online Shopping Website. The present chapter discusses the implementation of the system, highlighting the testing exercise and describing some of the main components of the system's Graphical User Interface (GUI). It will give an output from programming language and other tools used to develop our system. According to this plan, the activities are to be carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system.

**4.1 Choice of Development Environment**

The system was developed as an interactive mechanism between the user at the interface, and the database using the web browser. It was designed using Eclipse and Apech server. These tools enable the admin through a browser to interact with ORACLE database to enter, edit, view, and retrieve data from the database, as a privilege granted. These activities were achieved using JAVA, HTML forms which offer the best layout to enter data and view the database content. This form was also kept as short and simple as possible to suit the individual who will provide data.

**4.2 Implementation Architecture**

**BACK END**

**HTML ,BOOTSTRAP, JQUERY,JAVASCRIPT**

**MIDDLEEND**

**FRONT END**

**JAVA (forms)**

**Oracle Database**

**Fig 4.0 implementation Architecture**

**4.3 Software Testing**

During the development of the system, the application undergoes two phases of testing:

Firstly, testing done during development phase. This testing includes:

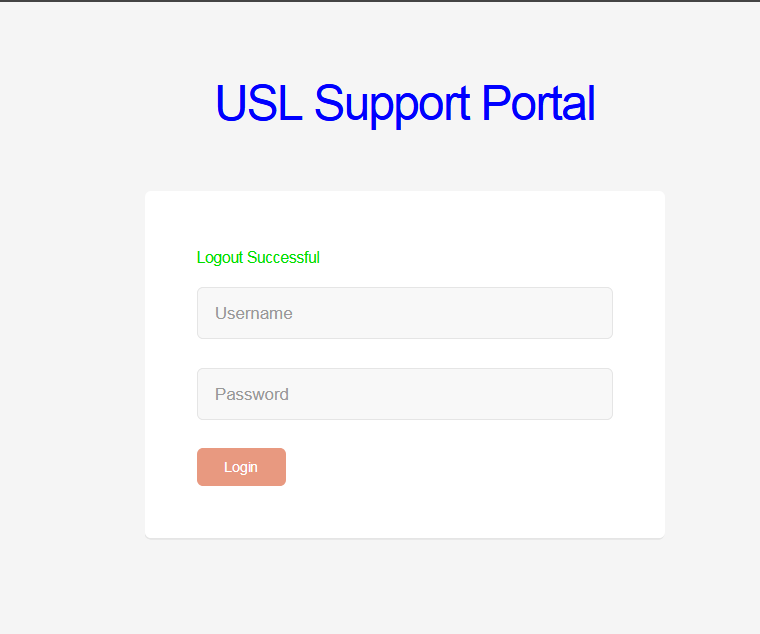
* Syntax error testing: this testing was done to the detect errors during development
* Compatibility testing: this testing is used to test if all the languages used during the programming were compatible with each other.
* Logical testing: This will check whether the argument is accepted by the system or not.

Secondly, testing them by running the software on realistic data samples. This testing includes:

* Running the application in its fullness using a local server Apache Tomcat sever that have local host which uses Apache as engine and MYSQL as the database.
* Browser testing: this test is done to view the design ,looks and format in it took on the browser

Shown below are the sample outputs of Issue Tracker management system.

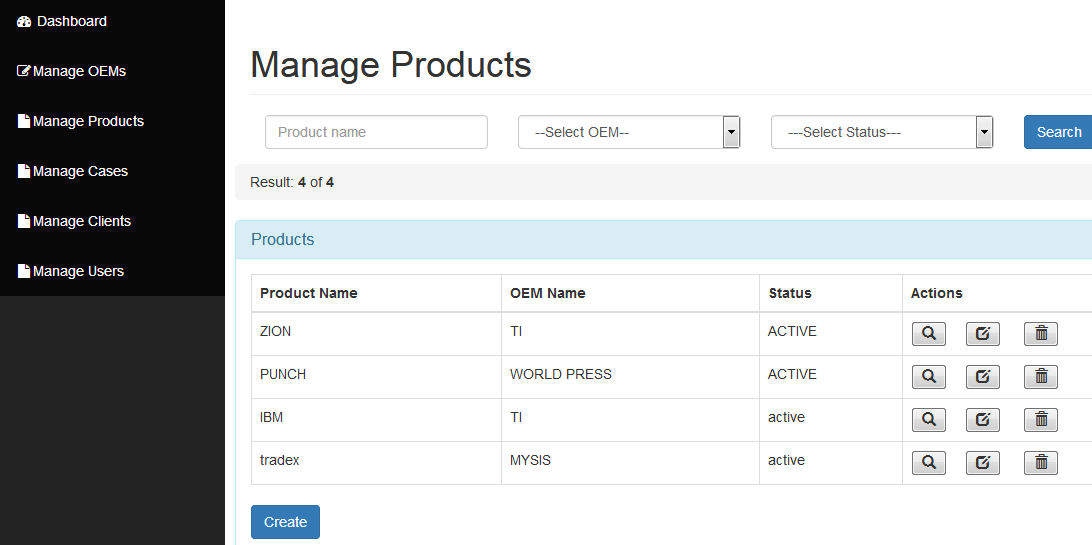
**Login page**: where only those registered by the administrator have full access to make use of the web application. If a user have been added, he/she must have access to their mail because every details like (user name and password)will be sent to the mail. It is the work of the admin manager to assign a normal user the admin role by changing the type from user to admin .



**Figure 4.1 login panel**

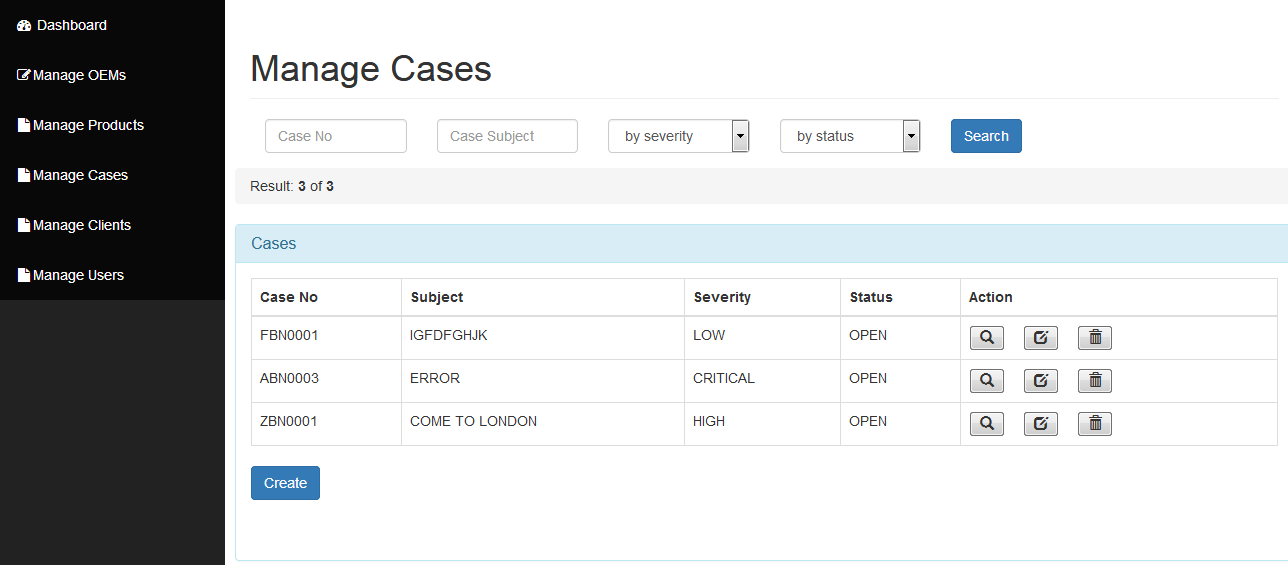
As an Admin you have access to every entry’s in the web page, creating, managing, adding users, deleting users, following up on issues, also have the power to assign the right department to an issue.

**Product** : this can only be accessed by the admin



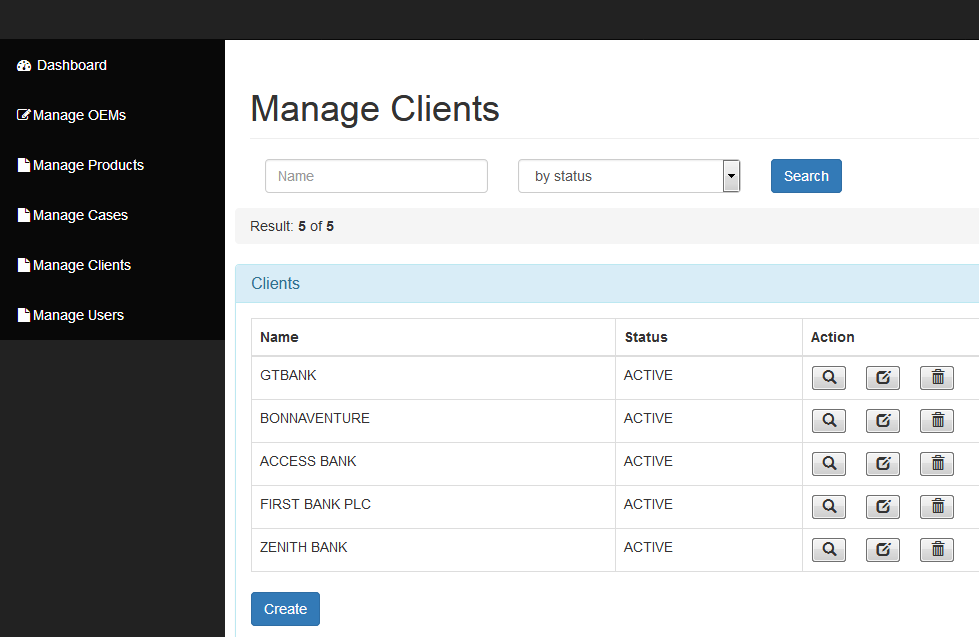
**Figure 4.2 manage product**

**Cases**: this is where admin and other user who have access to this page drop there complains and make sure it resolve.



**Figure 4.3 manage cases**

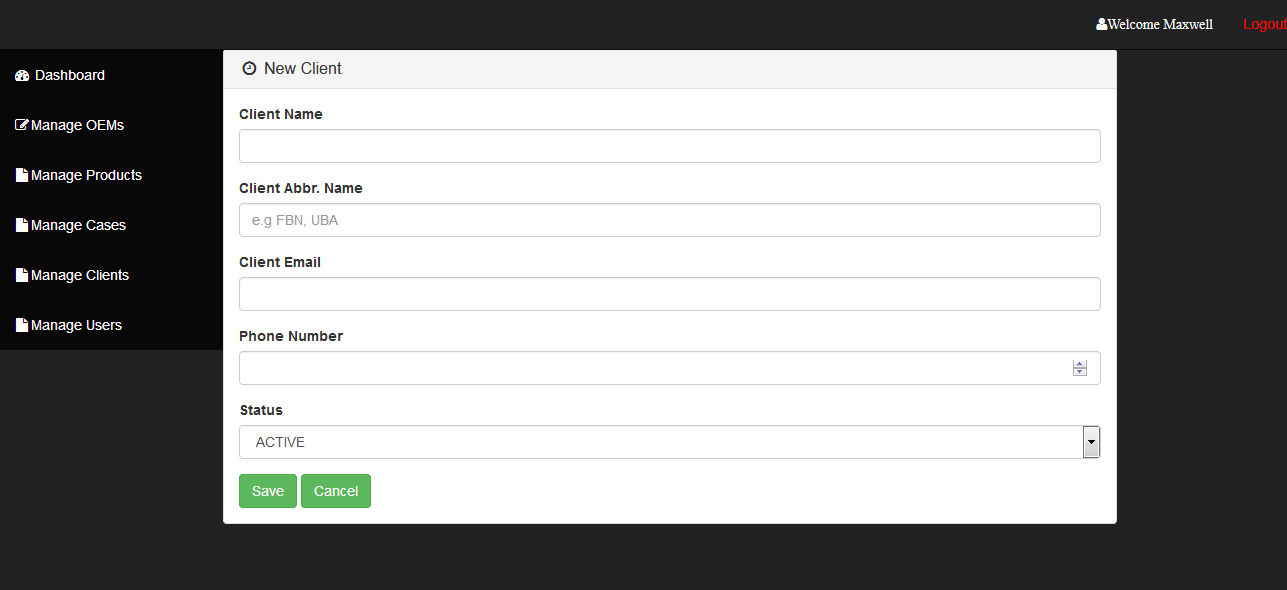
**Clients page:**



**Figure 4.4 manage clients**

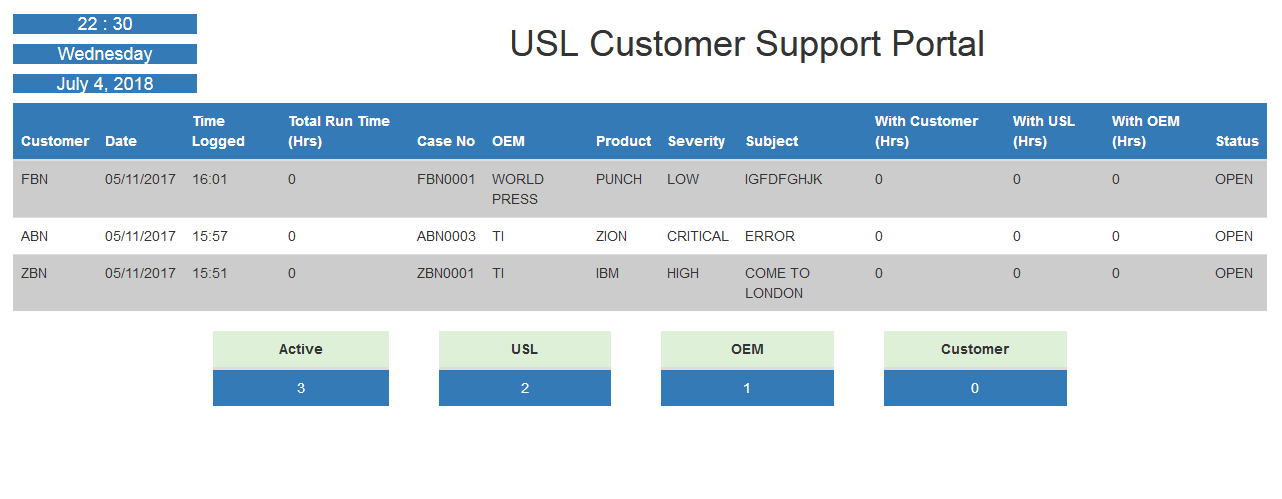
This page below shows the details when adding a client

**Adding New Client**



**Figure 4.5 adding client**

**Dash Board:** This page shows the complains made by customers. The features on the dash board are,customer name, date of complaint, time logged, total run time , case number, and status.



**Figure 4.6 Dashboard**

**4.4 Documentation**

This programs can be executed by simply running them in eclispe programming platform, it can also be converted to War File so it can run in other systems that have the eclispe platform. Copying the folder stored on the computer and executing them won’t work. This is quit advanced in nature because of the advancement in technology. Other programs are supplied in a form unsuitable for immediate execution and therefore need an installation procedure.

This system was built with HTML tags, JAVA, Boostrap, and Oracle query language, they are open source program that allows modification to be done in future.

**4.4.1 User Manual**

The following are the step involve in setting up the Issue Tracker website:

1. Install the Apache Tomcat Server and Eclipse IDE .
2. Copy the **IssueTracker** folder to your www directory.

***Locating Root Folder.....Click on Computer....Click on Local disk c...navigate to Tomcat.....Click on htdocs …..Paste the folder......***

1. Database file will be imported through **oracle dbforge** connected to the eclipse IDE.
2. Launch your browser.
3. Type **http://localhost/IssueTrackingSystem**.
4. Click enter.

**The system has the following menu commands**

1. *Add: this menu for adding records into the database*
2. *Search*: this menu search for the customer ,products, OEM
3. *View*. On clicking this menu all details that where submitted to the database will be displayed.
4. *Edit*. On clicking this menu it will open form that will give access for correction of error.
5. *Delete.* On clicking this menu it will erase the record that is not needed on the table.

**4.4.2 Source Code**

The source code of this system will be attached on the appendix.

CHAPTER FIVE

SUMMARY, CONCLUSION, AND RECOMMENDATION

5.0 Summary

Issue Tracker is web application specially design for making complains and the tracking of every complains.

The system has the benefits of easy access because it is be developed as a platform independent web application, so the admin can maintain a proper contact with their users, which may be access anywhere . All communications between the client/user and administrator has done through the online, so this communication cost also be reduced.

5.1 Conclusion

The main objective of this Issue Tracker system is to focus on the issues related to company system. Issue Tracker system is a platform independent application, so this web application can be accessed anywhere in the system. This is also developed for reduces the communication cost between the staffs and to provide the efficient service to their customers.

5.2 Recommendation

This system is found tested and examined for its successful processing. Future change in the environment or processing can be easily adopted by having simple  
change in coding. It is very user friendly, cost effective, feature rich and it provides very high level of security. It protects the unauthorized users. Moreover, the system coding is so well designed that new operations can be easily incorporated without much modification. A facility to inform through SMS or Email on landing of the consignment can be added in future.

**References:**

[1] Cho Y., Hiltz R., & Fjermestad J., “An Analysis of Online Customer Complaints: Implications for Web Complaint Management.” in *Proceedings of the 35th* *Hawaii International Conference on System* *Sciences*, Hawaii, (2002).

[2] Fornell C., & Westbrook A. Robert, “The Vicious Circle of Consumer Complaints,” *journal of Marketing,(summer)(1984), 68-78*

[3] Jay DiMare & Richard S. Ma, “Serviceoriented architecture Revolutionizing today's banking systems.” White Paper.

[4] Julta, D., Craig, J., & Bodorik, P., “Enabling and Measuring Electronic Customer Relationship Management Readiness,” in *Proceedings of the 34th Hawaii International* *Conference on System Sciences*, Hawaii,(2001).

[5] Najar, A. S., Al-Sukhni, H. A., & Aghakhani, N., “The Application of Service-Oriented Architecture in E-complaint System.” *Paper* *presented at (ICCSN '10) the Second* *International Conference on Communication* *Software and Networks,* (2010, 26-28 Feb.2010).

[6] Razali R., Abd Halim K. N., & Jusoff K., “Quality Improvement of Services in Unversiti Teknologi Mara Pahang from a Management Perspective.” Management Science & Engineering Vol.5, No.1, (2011), pp. 71-80.

[7] Yi, Yooujae, “A Critical Review of Consumer Satisfaction.” In *Review of Markiting,* Valerie A, Zeithaml, ed. Chicago, American Marketing Association, (1990).

[8] Blodgett, J. G., K. L. "The Effects of Customer Service on Consumer Complaining Behavior." Journal of Services Marketing 9(4): Wakefield, et al. 1995.pp 31- 42.

[9] Chung -Herrera, B. G. "Customer and Employee Views of Critical Service Complaints Management Systems Guide to Design and Implementation. British Standards Institute. N. Goldschmidt, et al. (2004).pp45-49.

**Appendix**

**Interface case list view**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta name="description" content="">

<meta name="author" content="">

<title>USL-Issue Tracker</title>

<!-- Bootstrap Core CSS -->

<link href="<%= request.getContextPath() %>/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom CSS -->

<link href="<%= request.getContextPath() %>/css/sb-admin.css" rel="stylesheet">

<!-- Morris Charts CSS -->

<link href="<%= request.getContextPath() %>/css/plugins/morris.css"rel="stylesheet">

<!-- Custom Fonts -->

<link href="<%= request.getContextPath() %>/font-awesome/css/font awesome.min.css"rel="stylesheet" type="text/css">

<!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->

<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->

<!--[if lt IE 9]>

<script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>

<script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>

<![endif]-->

<script src="<%=request.getContextPath()%>/js/caseActions.js"></script>

</head>

<body>

<c:if test="${**empty** sessionScope.user }">

<% response.sendRedirect(request.getContextPath()); %>

</c:if>

<div id="wrapper">

<jsp:include page="/views/sideBarMenuView.jsp" />

<div id="page-wrapper">

<div class="container-fluid">

<!-- Page Heading -->

<div class="row">

<div class="col-lg-12">

<h1 class="page-header">Manage Cases</h1><!-- Search -->

<div class="row">

<form action="<c:url value="searchCase?page=1"/>" method="post">

<div class="col-lg-12">

<div class="col-lg-2">

<input class="form-control" type="text" placeholder="Case No" name="caseNo" value="${requestScope.caseNo != null ? caseNo : ''}"></div>

<div class="col-lg-2">

<input class="form-control" type="text" placeholder="Case Subject" name="searchKeyword" value="${requestScope.searchKeyword != null ? searchKeyword : ''}">

</div>

<div class="col-lg-2">

<div class="form-group">

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

<option value="">by severity</option>

<option value="LOW" ${param.bySeverity != null && param.bySeverity == 'LOW' ? 'selected' : ''}>LOW</option>

<option value="MEDIUM" ${param.bySeverity != null && param.bySeverity == 'MEDIUM' ? 'selected' : ''}>MEDIUM</option>

<option value="HIGH" ${param.bySeverity != null && param.bySeverity == 'HIGH' ? 'selected' : ''}>HIGH</option>

<option value="CRITICAL" ${param.bySeverity != null && param.bySeverity == 'CRITICAL' ? 'selected' : ''}>CRITICAL</option>

</select></div></div> <div class="col-lg-2">

<div class="form-group">

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

<option value="">by status</option>

<option value="OPEN" ${param.byStatus != null && param.byStatus == 'OPEN' ? 'selected' : ''}>OPEN</option>

<option value="CLOSED" ${param.byStatus != null && param.byStatus == 'CLOSED' ? 'selected' : ''}>CLOSED</option>

</select>

</div>

</div>

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

<div class="form-group">

<button class="btn btn-secondary btn-primary" type="submit" name="searchcase" value="">Search</button>

</div>

</div>

</div>

</form>

</div>

</div> <!-- .col-lg-12 -->

<div class="row">

<div class="col-lg-12">

<ol class="breadcrumb">

<li class="breadcrumb-item">Result: <c:choose>

<c:when test="${requestScope.totalRecords == 0}">

<b>${requestScope.totalRecords}</b>

</c:when><c:when test="${requestScope.totalRecords <= sessionScope.recordsPerPage }">

<b>${requestScope.totalRecords} </b> of <b>${requestScope.totalRecords}</b>

</c:when><c:when test="${requestScope.endingRowNo >= requestScope.totalRecords}"><b>${requestScope.startingRowNo}</b> - <b>${requestScope.totalRecords}</b> of <b>${requestScope.totalRecords}</b>

</c:when><c:otherwise>

<b>${requestScope.startingRowNo}</b> - <b>${requestScope.endingRowNo}</b> of <b>${requestScope.totalRecords}</b>

</c:otherwise>

</c:choose>

</li>

</ol>

</div>

</div>

<div class="row">

<div class="col-lg-12">

<div class="panel panel-info">

<div class="panel-heading">

<h3 class="panel-title"> Cases</h3>

</div>

<div class="panel-body">

<c:choose>

<c:when test="${**empty** requestScope.cases}">

<c:choose>

<c:when test="${requestScope.searchcase != null }"><h4>No Result Found</h4></c:when>

<c:otherwise><h4>No item added yet.</h4></c:otherwise>

</c:choose>

</c:when>

<c:otherwise>

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

<thead>

<tr>

<th>Case No</th>

<th>Subject</th>

<th>Severity</th>

<th>Status</th>

<th>Action</th>

</tr>

</thead>

<tbody>

<c:forEach var="kase" items="${requestScope.cases}">

<tr>

<td>${kase.caseNo}</td>

<td>${kase.subject}</td>

<td>${kase.severity}</td>

<td>${kase.status}</td>

<td>

<button id="viewCase" onclick="passDataToForm(${kase.caseId},'view')" title="view"><span class="glyphicon glyphicon-search" aria-hidden="true"></span></button>&emsp;

<c:if test="${fn:containsIgnoreCase('superuser support', sessionScope.user.role)}">

<button id="updateCase" onclick="passDataToForm(${kase.caseId},'update')" title="edit"><span class="glyphicon glyphicon-edit" aria-hidden="true"></span></button>&emsp;

</c:if>

<c:if test="${fn:containsIgnoreCase(sessionScope.user.role, 'superuser')}">

<button id="deleteCase" onclick="passDataToForm(${kase.caseId},'delete')" title="delete"><span class="glyphicon glyphicon-trash" aria-hidden="true"></span></button>

</c:if>

</td>

</tr>

</c:forEach>

</tbody>

</table>

</c:otherwise>

</c:choose>

<form id="casePrepareActionForm" action="<c:url value="/maintainCase"/>" method="post" style="display:none;">

<input id="caseId" type="hidden" name="caseId"/>

<input id="mode" type="hidden" name="mode"/>

<button type="submit" style="display:none;"></button>

</form>

<!-- Create case button -->

<div class="text-left">

<c:if test="${fn:containsIgnoreCase('client superuser', sessionScope.user.role)}">

<nav aria-label="Page navigation example">

<button class="btn btn-primary" onclick="passDataToForm(0, 'create')" role="button">Create</button>

</nav>

</c:if>

</div>

<!-- Pagination -->

<div class="text-right">

<nav aria-label="Page navigation example">

<ul class="pagination">

<c:if test="${requestScope.page != null && requestScope.page > 1 }">

<c:choose>

<c:when test="${requestScope.searchcase != null }">

<li class="page-item"><a class="page-link" href="<c:url value="/searchCase?page=${requestScope.page - 1 }&searchKeyword=${requestScope.searchKeyword}&caseNo=${requestScope.caseNo}&byProductId=${requestScope.byProductId}&bySeverity=${requestScope.bySeverity}&byStatus=${requestScope.byStatus}&searchcase=case"/>">Previous</a></li>

</c:when>

<c:otherwise>

<li class="page-item"><a class="page-link" href="<c:url value="/cases?page=${requestScope.page - 1}"/>">Previous</a></li>

</c:otherwise>

</c:choose>

</c:if>

<c:if test="${**empty** requestScope.endOfRecordsReached}">

<c:choose>

<c:when test="${**not empty** requestScope.searchcase}">

<li class="page-item"><a class="page-link" href="<c:url value="/searchCase?page=${requestScope.page + 1 }&searchKeyword=${requestScope.searchKeyword}&caseNo=${requestScope.caseNo}&byProductId=${requestScope.byProductId}&bySeverity=${requestScope.bySeverity}&byStatus=${requestScope.byStatus}&searchcase=case"/>">Next</a></li>

</c:when>

<c:otherwise>

<li class="page-item"><a class="page-link" href="<c:url value="/cases?page=${requestScope.page + 1}"/>">Next</a></li>

</c:otherwise>

</c:choose>

</c:if>

</ul>

</nav>

</div>

</div> <!-- .panel-body -->

</div> <!-- .panel -->

</div>

</div>

</div>

</div> <!-- /.container-fluid -->

</div> <!-- /#page-wrapper -->

</div> <!-- /#wrapper -->

<!-- jQuery -->

<script src="<%=request.getContextPath() %>/js/jquery.js"></script>

<!-- Bootstrap Core JavaScript -->

<script src="<%=request.getContextPath() %>/js/bootstrap.min.js"></script>

<!-- Morris Charts JavaScript -->

<script src="<%=request.getContextPath() %>/js/plugins/morris/raphael.min.js"></script>

<script src="<%=request.getContextPath() %>/js/plugins/morris/morris.min.js"></script>

<script src="<%=request.getContextPath() %>/js/plugins/morris/morris-data.js"></script>

<!-- Case actions javascript -->

<script src="<%=request.getContextPath()%>/js/caseActions.js"></script>

</body>

</html>

Dashboard view

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"

import="com.unionsystems.models.\*"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta name="description" content="">

<meta name="author" content="">

<title>USL-Issue Tracker</title>

<!-- Bootstrap Core CSS -->

<link href="<%=request.getContextPath()%>/css/bootstrap.min.css"

rel="stylesheet">

<!-- Custom CSS -->

<link href="<%=request.getContextPath()%>/css/sb-admin.css"

rel="stylesheet">

<!-- Morris Charts CSS -->

<link href="<%=request.getContextPath()%>/css/plugins/morris.css"

rel="stylesheet">

<!-- Custom Fonts -->

<link

href="<%=request.getContextPath()%>/font-awesome/css/font-awesome.min.css"

rel="stylesheet" type="text/css">

<!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->

<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->

<!--[if lt IE 9]>

<script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>

<script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>

<![endif]-->

<style>

.productList {

width: 25%;

height: 28px;

padding: 6px 12px;

background-color: #fff;

border: 1px solid;

border-radius: 4px;

font-size: 12px;

margin-right: 30px;

}

.prod {

width: 22%;

height: 28px;

padding: 6px 12px;

background-color: #fff;

border: 1px solid;

border-radius: 4px;

font-size: 12px;

margin-right: 30px;

}

**table**.t01 {

border-collapse: separate;

border-spacing: 50px 0px;

}

.dashboard-row:nth-child(odd) {

background-color: #ccc;

}

</style>

</head>

<body style="background-color: #fff">

<div style="width: 100%;height: 100%">

<div id="page-wrapper"style="width: 100%;height: 100%">

<div class="container-fluid">

<div class="row">

<div class="col-lg-2">

<div>

<h4 class="bg-primary text-center">${requestScope.currentTime}</h4>

</div>

<div>

<h4 class="bg-primary text-center">${requestScope.currentDay}</h4>

</div>

<div>

<h4 class="bg-primary text-center">${requestScope.fullDate}</h4>

</div>

</div>

<div class="col-lg-10">

<h1 class="text-center">USL Customer Support Portal</h1>

</div>

</div>

<!-- Current date -->

<div>

<table class="table table-stripped table-responsive">

<thead>

<tr class="bg-primary">

<th>Customer</th>

<th>Date</th>

<th>Time Logged</th>

<th>Total Run Time (Hrs)</th>

<th>Case No</th>

<th>OEM</th>

<th>Product</th>

<th>Severity</th>

<th>Subject</th>

<th>With Customer (Hrs)</th>

<th>With USL (Hrs)</th>

<th>With OEM (Hrs)</th>

<th>Status</th>

</tr>

</thead>

<tbody>

<c:if test="${**not empty** requestScope.kases}">

<c:forEach var="kase" items="${requestScope.kases}">

<tr class="dashboard-row">

<td>${kase.clientShortNm}</td>

<td><fmt:formatDate value="${kase.createDate}" pattern="dd/MM/yyyy"/></td>

<td><fmt:formatDate value="${kase.createDate}" pattern="HH:mm"/></td>

<td>${kase.totalRuntime}</td>

<td>${kase.caseNo}</td>

<td>${kase.oemNm}</td>

<td>${kase.productNm}</td>

<td>${kase.severity}</td>

<td>${kase.subject}</td>

<td>${kase.totalCustHours}</td>

<td>${kase.totalOwnHours}</td>

<td>${kase.totalOemHours}</td>

<td>${kase.status}</td>

</tr>

</c:forEach>

</c:if>

</tbody>

</table>

</div>

<!-- Table data -->

<div>

<table class="table table-responsive t01">

<thead>

<tr class="row">

<th class="bg-success col-lg-3 text-center">Active</th>

<th class="bg-success col-lg-3 text-center">USL</th>

<th class="bg-success col-lg-3 text-center">OEM</th>

<th class="bg-success col-lg-3 text-center">Customer</th>

</tr>

</thead>

<tbody>

<tr class="row">

<td class="bg-primary col-lg-3 text-center">${requestScope.activeCases}</td>

<td class="bg-primary col-lg-3 text-center">${requestScope.casesWithUsl}</td>

<td class="bg-primary col-lg-3 text-center">${requestScope.casesWithOem}</td>

<td class="bg-primary col-lg-3 text-center">${requestScope.casesWithCustomer}</td>

</tr>

</tbody>

</table>

</div>

</div>

<!-- /.container-fluid -->

</div>

<!-- /#page-wrapper -->

</div>

<!-- /#wrapper -->

<!-- jQuery -->

<script src="..js/jquery.js"></script>

<!-- Bootstrap Core JavaScript -->

<script src="..js/bootstrap.min.js"></script>

<!-- Morris Charts JavaScript -->

<script

src="<%= request.getContextPath() %>/js/plugins/morris/raphael.min.js"></script>

<script

src="<%= request.getContextPath() %>/js/plugins/morris/morris.min.js"></script>

<script

src="<%= request.getContextPath() %>/js/plugins/morris/morris-data.js"></script>

</body>

</html>

**Connection to the database**

package com.unionsystems.conn;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class ConnectionUtils {

private static String dbURL = "jdbc:oracle:thin:@localhost:1521:xe";

private static String dbUsername = "casetracker";

private static String dbPassword = "casetracker";

static String driverName = "oracle.jdbc.driver.OracleDriver";

public static Connection getConnection() throws SQLException

{

Connection conn = null;

try {

Class.forName(driverName);

try {

conn = DriverManager.getConnection(dbURL, dbUsername, dbPassword);

} finally {}

;

} catch (ClassNotFoundException e1) {

e1.printStackTrace();

}

return conn;

}}

**Users models**

package com.unionsystems.models;

import com.unionsystems.services.ClientServices;

import java.sql.\*;

public class User {

private int user\_id;

private String first\_name;

private String last\_name;

private String email;

private String role;

private String password;

private String status;

private Timestamp create\_date;

private int client\_id;

private String team;

private String change\_pass\_fg;

public String getChange\_pass\_fg() {

return change\_pass\_fg;

}

public void setChange\_pass\_fg(String change\_pass\_fg) {

this.change\_pass\_fg = change\_pass\_fg;

}

public User() {

}

public int getUser\_id() {

return user\_id;

}

public void setUser\_id(int user\_id) {

this.user\_id = user\_id;

}

public String getFirst\_name() {

return first\_name;

}

public void setFirst\_name(String first\_name) {

this.first\_name = first\_name;

}

public String getLast\_name() {

return last\_name;

}

public void setLast\_name(String last\_name) {

this.last\_name = last\_name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public Timestamp getCreate\_date() {

return create\_date;

}

public void setCreate\_date(Timestamp create\_date) {

this.create\_date = create\_date;

}

public int getClient\_id() {

return client\_id;

}

public void setClient\_id(int client\_id) {

this.client\_id = client\_id;

}

public String getTeam() {

return team;

}

public void setTeam(String team) {

this.team = team;

}

public String getClientName() {

return ClientServices.findClient(client\_id).getClient\_Name();

}

}

package com.unionsystems.services;

import java.io.IOException;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import java.util.Set;

import com.unionsystems.conn.ConnectionUtils;

import com.unionsystems.models.Case;

public class CaseServices {

public static boolean createCase(Case case1) throws SQLException

{

boolean rowInserted = false;

try (Connection conn = ConnectionUtils.getConnection())

{

String sql = "INSERT INTO case (case\_id, client\_id, product\_id, total\_cust\_hours, total\_oem\_hours, total\_own\_hours," +

" case\_no, product\_version, subject, pending\_on, assigned\_to, severity," +

" status, pending\_date, create\_date, note, closed\_date)";

sql += " VALUES (case\_sequence.nextval, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

try (PreparedStatement pStmt = conn.prepareStatement(sql)) {

pStmt.setInt(1, case1.getClientId());

pStmt.setInt(2, case1.getProductId());

pStmt.setInt(3, case1.getTotalCustHours());

pStmt.setInt(4, case1.getTotalOemHours());

pStmt.setInt(5, case1.getTotalOwnHours());

pStmt.setString(6, case1.getCaseNo().toUpperCase());

pStmt.setString(7, case1.getProductVersion().toUpperCase());

pStmt.setString(8, case1.getSubject().toUpperCase());

pStmt.setString(9, case1.getPendingOn().toUpperCase());

pStmt.setInt(10, case1.getAssignedTo());

pStmt.setString(11, case1.getSeverity().toUpperCase());

pStmt.setString(12, case1.getStatus().toUpperCase());

pStmt.setTimestamp(13, case1.getPendingDate());

pStmt.setTimestamp(14, case1.getCreateDate());

pStmt.setString(15, case1.getNote());

pStmt.setTimestamp(16, case1.getClosedDate());

rowInserted = pStmt.executeUpdate() > 0;

}

}

return rowInserted;

}

public static Case findCaseById(int caseId)

{

Case kase = null;

try (Connection conn = ConnectionUtils.getConnection())

{

String sql = "SELECT \* FROM case WHERE case\_id = ?";

try (PreparedStatement pStmt = conn.prepareStatement(sql)) {

pStmt.setInt(1, caseId);

ResultSet resultSet = pStmt.executeQuery();

if (resultSet.next())

{

kase = new Case();

kase.setCaseId(caseId);

kase.setCaseNo(resultSet.getString("case\_no"));

kase.setProductId(resultSet.getInt("product\_id"));

kase.setClientId(resultSet.getInt("client\_id"));

kase.setProductVersion(resultSet.getString("product\_version"));

kase.setSubject(resultSet.getString("subject"));

kase.setPendingDate(resultSet.getTimestamp("pending\_date"));

kase.setPendingOn(resultSet.getString("pending\_on"));

kase.setTotalCustHours(resultSet.getInt("total\_cust\_hours"));

kase.setTotalOemHours(resultSet.getInt("total\_oem\_hours"));

kase.setTotalOwnHours(resultSet.getInt("total\_own\_hours"));

kase.setAssignedTo(resultSet.getInt("assigned\_to"));

kase.setSeverity(resultSet.getString("severity"));

kase.setCreateDate(resultSet.getTimestamp("create\_date"));

kase.setStatus(resultSet.getString("status"));

kase.setNote(resultSet.getString("note"));

kase.setClosedDate(resultSet.getTimestamp("closed\_date"));

}

} catch (SQLException e) {

e.printStackTrace();

}

} catch (SQLException e1) {

e1.printStackTrace();

}

return kase;

}

public static List<Case> findCases(Map<String, String> searchCriteria) throws SQLException, IOException

{

List<Case> cases = new ArrayList<Case>();

try (Connection conn = ConnectionUtils.getConnection())

{

StringBuilder sql = new StringBuilder("SELECT case\_id, product\_id, client\_id, create\_date, total\_cust\_hours, total\_oem\_hours, total\_own\_hours, case\_no, subject, pending\_on, severity, status FROM case ");

Set<String> colSet = null;

if (searchCriteria != null && !searchCriteria.isEmpty())

{

colSet = searchCriteria.keySet();

if (colSet != null && !colSet.isEmpty())

{

StringBuilder whereClause = new StringBuilder(" WHERE");

String andOper = "";

for (String colName : colSet)

{

whereClause.append(andOper);

whereClause.append(" ");

whereClause.append(colName);

if(colName.equals("subject") || colName.equals("case\_no"))

{

whereClause.append(" LIKE ?");

} else {

whereClause.append(" = ?");

}

andOper = " AND ";

}

sql.append(whereClause);

}

}

sql.append(" ORDER BY create\_date DESC");

try( PreparedStatement pStmt = conn.prepareStatement(sql.toString()) )

{

int paramPos = 1;

if (colSet != null) {

for(String colName : colSet)

{

if (colName.equals("subject") || colName.equals("case\_no"))

{

pStmt.setString(paramPos, String.format("%s%s%s", "%", searchCriteria.get(colName).toUpperCase(), "%"));

} else if (colName.equals("product\_id") || colName.equals("client\_id")) {

pStmt.setInt(paramPos, Integer.parseInt(searchCriteria.get(colName)));

} else {

pStmt.setString(paramPos, searchCriteria.get(colName));

}

paramPos++;

}

}

try ( ResultSet result = pStmt.executeQuery() )

{

while (result.next())

{

Case kase = new Case();

setCaseOutput(kase, result);

cases.add(kase);

}

}

}

}

return cases;

}

public static List<Case> findCases(Map<String, String> searchCriteria, int startingRowNo, int endingRowNo) throws SQLException, IOException

{

List<Case> cases = new ArrayList<Case>();

try (Connection conn = ConnectionUtils.getConnection())

{

StringBuilder sql = new StringBuilder("SELECT \* FROM (");

sql.append("SELECT a.\*, ROWNUM rnum FROM (");

sql.append("SELECT case\_id, product\_id, client\_id, create\_date, total\_cust\_hours, total\_oem\_hours, total\_own\_hours, case\_no, subject, pending\_on, severity, status FROM case ");

Set<String> colSet = null;

if (searchCriteria != null && !searchCriteria.isEmpty())

{

colSet = searchCriteria.keySet();

if (colSet != null && !colSet.isEmpty())

{

StringBuilder whereClause = new StringBuilder(" WHERE");

String andOper = "";

for (String colName : colSet)

{

whereClause.append(andOper);

whereClause.append(" ");

whereClause.append(colName);

if(colName.equals("subject") || colName.equals("case\_no"))

{

whereClause.append(" LIKE ?");

} else {

whereClause.append(" = ?");

}

andOper = " AND ";

}

sql.append(whereClause);

}

}

sql.append(" ORDER BY create\_date DESC");

sql.append(") a WHERE ROWNUM <= ?");

sql.append(") WHERE rnum >= ?");

try( PreparedStatement pStmt = conn.prepareStatement(sql.toString()) )

{

int paramPos = 1;

if (colSet != null) {

for(String colName : colSet)

{

if (colName.equals("subject") || colName.equals("case\_no"))

{

pStmt.setString(paramPos, String.format("%s%s%s", "%", searchCriteria.get(colName).toUpperCase(), "%"));

} else if (colName.equals("product\_id") || colName.equals("client\_id") ) {

pStmt.setInt(paramPos, Integer.parseInt(searchCriteria.get(colName)));

} else {

pStmt.setString(paramPos, searchCriteria.get(colName));

}

paramPos++;

}

}

pStmt.setInt(paramPos++, endingRowNo);

pStmt.setInt(paramPos, startingRowNo);

try ( ResultSet result = pStmt.executeQuery() )

{

while (result.next())

{

Case kase = new Case();

setCaseOutput(kase, result);

cases.add(kase);

}

}

}

}

return cases;

}

public static boolean updateCase(Case case1) throws SQLException

{

boolean rowUpdated = false;

try (Connection conn = ConnectionUtils.getConnection())

{

String sql = "UPDATE case SET case\_no=?, product\_id=?, client\_id=?, product\_version=?, subject=?, pending\_date=?," +

" pending\_on=?, total\_cust\_hours=?, total\_oem\_hours=?, total\_own\_hours=?, assigned\_to=?, severity=?, status=?," +

"note=?, closed\_date=? WHERE case\_id = ?";

try (PreparedStatement pStmt = conn.prepareStatement(sql)) {

pStmt.setString(1, case1.getCaseNo().toUpperCase());

pStmt.setInt(2, case1.getProductId());

pStmt.setInt(3, case1.getClientId());

pStmt.setString(4, case1.getProductVersion().toUpperCase());

pStmt.setString(5, case1.getSubject().toUpperCase());

pStmt.setTimestamp(6, case1.getPendingDate());

pStmt.setString(7, case1.getPendingOn().toUpperCase());

pStmt.setInt(8, case1.getTotalCustHours());

pStmt.setInt(9, case1.getTotalOemHours());

pStmt.setInt(10, case1.getTotalOwnHours());

pStmt.setInt(11, case1.getAssignedTo());

pStmt.setString(12, case1.getSeverity().toUpperCase());

pStmt.setString(13, case1.getStatus().toUpperCase());

pStmt.setString(14, case1.getNote());

pStmt.setTimestamp(15, case1.getClosedDate());

pStmt.setInt(16, case1.getCaseId());

rowUpdated = pStmt.executeUpdate() > 0;

}

}

return rowUpdated;

}

public static boolean deleteCase(int caseId)

{

boolean rowDeleted = false;

try (Connection conn = ConnectionUtils.getConnection())

{

String sql = "DELETE FROM case WHERE case\_id = ?";

try (PreparedStatement pStmt = conn.prepareStatement(sql)) {

pStmt.setInt(1, caseId);

rowDeleted = pStmt.executeUpdate() > 0;

} catch (SQLException e) {

e.printStackTrace();

}

} catch (SQLException e1) {

e1.printStackTrace();

}

return rowDeleted;

}

public static int getTotalNoOfSearchResult(Map<String, String> searchCriteria) throws SQLException {

int totalRecords = 0;

try (Connection conn = ConnectionUtils.getConnection()) {

StringBuilder sql = new StringBuilder("SELECT COUNT(case\_id) total\_records FROM case");

Set<String> colSet = null;

if (searchCriteria != null && !searchCriteria.isEmpty())

{

colSet = searchCriteria.keySet();

if (colSet != null && !colSet.isEmpty())

{

StringBuilder whereClause = new StringBuilder(" WHERE");

String andOper = "";

for (String colName : colSet)

{

whereClause.append(andOper);

whereClause.append(" ");

whereClause.append(colName);

if(colName.equals("subject") || colName.equals("case\_no"))

{

whereClause.append(" LIKE ?");

} else {

whereClause.append(" = ?");

}

andOper = " AND ";

}

sql.append(whereClause);

}

}

try( PreparedStatement pStmt = conn.prepareStatement(sql.toString()) )

{

int paramPos = 1;

if (colSet != null) {

for(String colName : colSet)

{

if (colName.equals("subject") || colName.equals("case\_no"))

{

pStmt.setString(paramPos, String.format("%s%s%s", "%", searchCriteria.get(colName).toUpperCase(), "%"));

} else if ("product\_id client\_id ".contains(colName.toLowerCase())) {

pStmt.setInt(paramPos, Integer.parseInt(searchCriteria.get(colName)));

} else {

pStmt.setString(paramPos, searchCriteria.get(colName));

}

paramPos++;

}

}

try(ResultSet result = pStmt.executeQuery()) {

if(result.next()) {

totalRecords = result.getInt("total\_records");

}

}

}

}

return totalRecords;

}

private static void setCaseOutput(Case kase, ResultSet result) throws SQLException {

kase.setCaseId(result.getInt("case\_id"));

kase.setCaseNo(result.getString("case\_no"));

kase.setSubject(result.getString("subject"));

kase.setPendingOn(result.getString("pending\_on"));

kase.setSeverity(result.getString("severity"));

kase.setStatus(result.getString("status"));

kase.setProductId(result.getInt("product\_id"));

kase.setClientId(result.getInt("client\_id"));

kase.setTotalCustHours(result.getInt("total\_cust\_hours"));

kase.setTotalOemHours(result.getInt("total\_oem\_hours"));

kase.setTotalOwnHours(result.getInt("total\_own\_hours"));

kase.setCreateDate(result.getTimestamp("create\_date"));

}

}

package com.unionsystems.servlets;

import java.io.IOException;

import java.sql.SQLException;

import java.sql.Timestamp;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.mail.MessagingException;

import javax.mail.internet.AddressException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import com.unionsystems.servlets.AbstractCrudServlet;

import com.unionsystems.models.Case;

import com.unionsystems.models.Client;

import com.unionsystems.models.Comment;

import com.unionsystems.models.Product;

import com.unionsystems.models.User;

import com.unionsystems.services.CaseServices;

import com.unionsystems.services.ClientServices;

import com.unionsystems.services.CommentService;

import com.unionsystems.services.EmailService;

import com.unionsystems.services.ProductServices;

import com.unionsystems.services.UserServices;

@WebServlet("/maintainCase")

public class CaseCrudServlet extends AbstractCrudServlet {

private static final long serialVersionUID = -3520642614907403131L;

@Override

protected void prepareCreate(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

HttpSession session = request.getSession();

if(session != null && ( (User)session.getAttribute("user")) != null ) {

List<Client> clients = ClientServices.queryClient();

if(!clients.isEmpty()) {

session.setAttribute("clients", clients);

}

List<Product> products = ProductServices.getAllRecords();

if(!products.isEmpty()) {

session.setAttribute("products", products);

}

Map<String, String> searchCriteria = new HashMap<String, String>();

searchCriteria.put("role", "STAFF");

searchCriteria.put("status", "ACTIVE");

List<User> staffList = new ArrayList<User>();

try {

staffList = UserServices.findAllUsers(searchCriteria);

} catch (SQLException e) {

e.printStackTrace();

}

if(!staffList.isEmpty()) {

session.setAttribute("staffList", staffList);

}

}

request.getRequestDispatcher("/views/maintainCaseView.jsp?mode=createCase").forward(request,

response);

}

/\*

\* Sends the necessary info needed for view, update and delete actions, to maintainCaseView.jsp

\*/

@Override

protected void prepareAction(HttpServletRequest request, HttpServletResponse response, String mode)

throws ServletException, IOException {

String strCaseId = request.getParameter("caseId");

HttpSession session = request.getSession();

if (strCaseId != null && session != null && ((User)session.getAttribute("user")) != null) {

int caseId = Integer.parseInt(strCaseId);

Case kase = CaseServices.findCaseById(caseId);

if(kase != null) {

request.setAttribute("case", kase);

int productId = kase.getProductId();

String productNm = ProductServices.findProduct(productId).getName();

request.setAttribute("productNm", productNm);

int clientId = kase.getClientId();

String clientNm = ClientServices.findClient(clientId).getClient\_Name();

request.setAttribute("clientNm", clientNm);

int staffId = kase.getAssignedTo();

// if case has been assigned to a staff

if(staffId > 0) {

User staff = UserServices.findUserById(staffId);

request.setAttribute("staff", staff);

}

Map<String, String> searchCriteria = new HashMap<String, String>();

searchCriteria.put("role", "STAFF");

searchCriteria.put("status", "ACTIVE");

List<User> staffList = new ArrayList<User>();

try {

staffList = UserServices.findAllUsers(searchCriteria);

} catch (SQLException e1) {

e1.printStackTrace();

}

request.setAttribute("allStaff", staffList);

try {

List<Comment> comments = CommentService.findCommentsByCaseId(caseId);

session.setAttribute("comments", comments);

} catch (SQLException e) {

e.printStackTrace();

}

}

request.getRequestDispatcher("/views/maintainCaseView.jsp?mode=" + mode + "Case").forward(request, response);

} else {

response.sendRedirect(request.getContextPath() + "/cases");

}

}

@Override

protected void performCreateAction(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

int clientId = Integer.parseInt(request.getParameter("clientId"));

int productId = Integer.parseInt(request.getParameter("productId"));

String productVersion = request.getParameter("productVersion");

String subject = request.getParameter("subject");

String note = request.getParameter("note");

String severity = request.getParameter("severity");

Case case1 = new Case();

String strAssignedTo = request.getParameter("staffId");

if(strAssignedTo != null && !strAssignedTo.isEmpty()) {

case1.setAssignedTo(Integer.parseInt(strAssignedTo));

}

case1.setClientId(clientId);

case1.setProductId(productId);

case1.setProductVersion(productVersion);

case1.setSubject(subject);

case1.setNote(note);

case1.setSeverity(severity);

case1.setCaseNo(generateCaseNo(clientId));

Timestamp createDate = new Timestamp(System.currentTimeMillis());

case1.setCreateDate(createDate);

case1.setPendingDate(createDate);

case1.setPendingOn("OWN");

case1.setStatus("OPEN");

HttpSession session = request.getSession();

User user = null;

if(session != null && (user = (User) session.getAttribute("user")) != null) {

case1.setAuthorId(user.getUser\_id());

}

try {

CaseServices.createCase(case1);

// Send new case email alert

StringBuilder recipients = new StringBuilder();

Map<String, String> searchCriteria = new HashMap<String, String>();

searchCriteria.put("role", "SUPPORT");

searchCriteria.put("status", "ACTIVE");

try {

List<User> supportTeam = UserServices.findAllUsers(searchCriteria);

for(User supportPerson : supportTeam) {

recipients.append(String.format("%s,", supportPerson.getEmail()));

}

} catch (SQLException e1) {

e1.printStackTrace();

}

// if a support person has been added to the system

if (recipients.length() > 0) {

String emailToAddress = recipients.toString();

String emailSubject = String.format("New Case (%s)", case1.getCaseNo());

String message = String.format("A new case with Case No %s has been created.", case1.getCaseNo());

try {

EmailService.sendEmail(emailToAddress, emailSubject, message);

} catch (AddressException e) {

e.printStackTrace();

} catch (MessagingException e) {

e.printStackTrace();

}

}

} catch (SQLException e2) {

e2.printStackTrace();

}

response.sendRedirect(request.getContextPath() + "/cases");

}

@Override

protected void performUpdateAction(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String strCaseId = request.getParameter("caseId");

int caseId;

// if view Case action was performed and new comment added

if (request.getParameter("addCommentButton") != null && strCaseId != null) {

Comment comment = new Comment();

caseId = Integer.parseInt(strCaseId);

String newComment = request.getParameter("newComment");

String commentOwnerId = request.getParameter("commentOwnerId");

Timestamp createDate = new Timestamp(System.currentTimeMillis());

if (newComment != null && !newComment.isEmpty() && commentOwnerId != null) {

comment.setCaseId(caseId);

comment.setUserId(Integer.parseInt(commentOwnerId));

comment.setDetail(newComment);

comment.setCreateDate(createDate);

try {

CommentService.saveComment(comment);

// send new comment notification to resource person if role is client

User commentOwner = UserServices.findUserById(Integer.parseInt(commentOwnerId));

String recipientEmailAddress;

String emailSubject = "New comment";

Case kase = CaseServices.findCaseById(caseId);

String emailMessage = String.format("A new comment has been added to case %s", kase.getCaseNo());

if (commentOwner.getRole().equalsIgnoreCase("client")) {

int resourceId = kase.getAssignedTo();

// if the case has been assigned to a staff

if (resourceId > 0) {

recipientEmailAddress = UserServices.findUserById(resourceId).getEmail();

try {

EmailService.sendEmail(recipientEmailAddress, emailSubject, emailMessage);

} catch (AddressException e) {

e.printStackTrace();

} catch (MessagingException e) {

e.printStackTrace();

}

}

} else if (commentOwner.getRole().equalsIgnoreCase("staff")) {

int clientId = kase.getAuthorId();

recipientEmailAddress = UserServices.findUserById(clientId).getEmail();

try {

EmailService.sendEmail(recipientEmailAddress, emailSubject, emailMessage);

} catch (AddressException e) {

e.printStackTrace();

} catch (MessagingException e) {

e.printStackTrace();

}

}

} catch (SQLException e) {

e.printStackTrace();

}

}

// return back to case view page

prepareAction(request, response, "view");

} else if(request.getParameter("editCommentButton") != null && strCaseId != null) {

Comment updatedComment = new Comment();

String strLastCommentId = request.getParameter("lastCommentId");

String editedComment = request.getParameter("newComment");

if(strLastCommentId != null && editedComment != null) {

updatedComment.setCommentId(Integer.parseInt(strLastCommentId));

updatedComment.setDetail(editedComment);

updatedComment.setCreateDate(new Timestamp(System.currentTimeMillis()));

try {

CommentService.updateComment(updatedComment);

} catch (SQLException e) {

e.printStackTrace();

}

}

prepareAction(request, response, "view");

} else if (request.getParameter("updateCaseMode") != null && strCaseId != null) {

caseId = Integer.parseInt(strCaseId);

Case case1 = CaseServices.findCaseById(caseId);

if (case1 != null) {

String currPendingOn = request.getParameter("pendingOn");

String prevPendingOn = case1.getPendingOn();

if (!currPendingOn.equals(prevPendingOn)) {

Timestamp pendingDate = case1.getPendingDate();

long pendingDateInMillis = pendingDate.getTime();

long currDateInMillis = System.currentTimeMillis();

int timeDiffInHours = (int) ((currDateInMillis - pendingDateInMillis) / (60 \* 60 \* 1000));

switch (prevPendingOn) {

case "CLIENT":

int totalCustHours = case1.getTotalCustHours() + timeDiffInHours;

case1.setTotalCustHours(totalCustHours);

break;

case "OWN":

int totalOwnHours = case1.getTotalOwnHours() + timeDiffInHours;

case1.setTotalOwnHours(totalOwnHours);

break;

case "OEM":

int totalOemHours = case1.getTotalOemHours() + timeDiffInHours;

case1.setTotalOemHours(totalOemHours);

}

case1.setPendingOn(currPendingOn);

case1.setPendingDate(new Timestamp(currDateInMillis));

}

String status = request.getParameter("status");

String subject = request.getParameter("subject");

String note = request.getParameter("note");

String strStaffId = request.getParameter("staffId");

boolean staffAssignedToCaseHasChanged = false;

if (strStaffId != null) {

int staffId = Integer.parseInt(strStaffId);

if(staffId != case1.getAssignedTo()) {

staffAssignedToCaseHasChanged = true;

case1.setAssignedTo(staffId);

}

}

case1.setStatus(status);

case1.setSubject(subject);

case1.setNote(note);

try {

CaseServices.updateCase(case1);

// send an email alert to the newly assigned staff

if (staffAssignedToCaseHasChanged) {

String staffEmail = UserServices.findUserById(Integer.parseInt(strStaffId)).getEmail();

String emailSubject = "New case";

String emailMessage = String.format("A case with Case No %s has been assigned to you.", case1.getCaseNo());

try {

EmailService.sendEmail(staffEmail, emailSubject, emailMessage);

} catch (AddressException e) {

e.printStackTrace();

} catch (MessagingException e) {

e.printStackTrace();

}

}

} catch (SQLException e1) {

e1.printStackTrace();

}

}

response.sendRedirect(request.getContextPath() + "/cases");

}

}

@Override

protected void performDeleteAction(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String strCaseId = request.getParameter("caseId");

if (strCaseId != null) {

int caseId = Integer.parseInt(strCaseId);

CaseServices.deleteCase(caseId);

}

response.sendRedirect(request.getContextPath() + "/cases");

}

private String generateCaseNo(int clientId) {

Client client = ClientServices.findClient(clientId);

String clientShortNm = client.getClientShortNm();

int caseNoCounter = client.getCaseNoCounter();

String caseNo = String.format("%s%04d", clientShortNm, ++caseNoCounter);

client.setCaseNoCounter(caseNoCounter);

ClientServices.updateClient(client);

return caseNo;

}

}