

**BANK ACCOUNTS MANAGEMENT SYSTEM**

**NAME: Redwan Ahmed**

**INSTITUTION: University of Greenwich**

**COURSE TITLE: Programming Enterprise Components**

**COURSE CODE: COMP-1610-M01-2020-21**

**NAME OF INSRUCTOR: Dr. A K M Mahtab Hossain**

**DATE OF SUBMISSION: 08/04/2021**

## **INTRODUCTION:**

The aim of this project is to develop an online banking system for Net Bank. In currently existing systems customers have to go to the banks physically for any kind of transactions which includes deposit or withdrawal. The project that has been created provides an online platform which automate the banking process. This bank management system its customer can check the balance, transfer and withdraw forms their accounts. The core aim for developing this bank management system which has the ability to store 100% accurate data and provide customers related details.

This Net Bank management system allows users to add new customer accounts along with searching for any individual account. This system also includes feature like checking any transactions in any account by the client. This system also provides security check reducing the chances of fraud using verify login logics which is connected to the systems database.

## **Demonstration:**

Video link:

<https://gre.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=cc9b43d9-dae1-4e64-9f40-ad04017652ff>

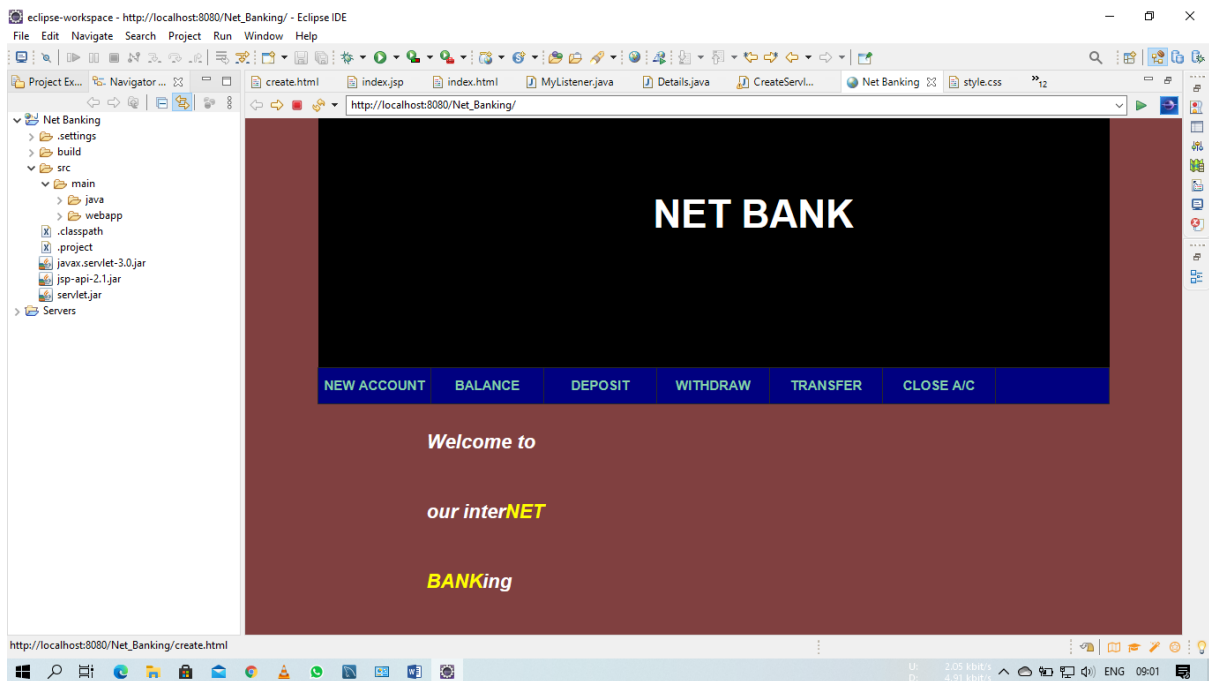
## SECTION 1

| Task List | Implementation summary.   |
|-----------|---|
| 1         | Implemented using java EE, MySQL, tomcat server, Html 5, css 3 and JavaScript |
| 2         | Implemented on the front end  |
| 3         | Tried but Not implemented   |
| 4         | Implemented   |
| 5         | Implemented 3 technologies  |
| 6         | Implemented.  |

## SECTION 2

### Task 1

In this task I have created a dashboard that contains a navigation bar as shown below. The navigation bar contains the attributes of a customer such as funds withdrawal, deposit funds transfer and check balance. On running the dynamic web project, a default file (index.html) set in the web.xml file runs on the server thereby rendering back the dashboard to the customer.

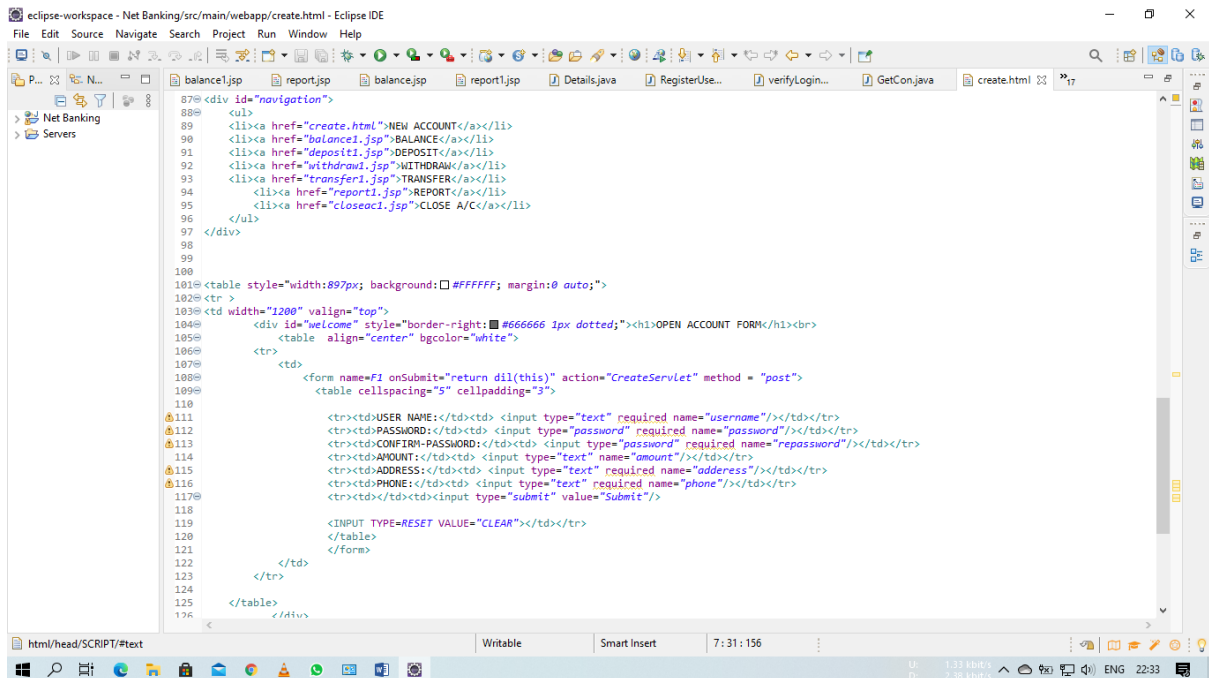


From the above screenshot, the customer navigates to whatever action they wish to take in the application.

However, the customer needs to be in the database otherwise they would not go past the verify login step. If they are not registered, then there is the 'NEW ACCOUNT' tab that offers a chance to carry out their registration.

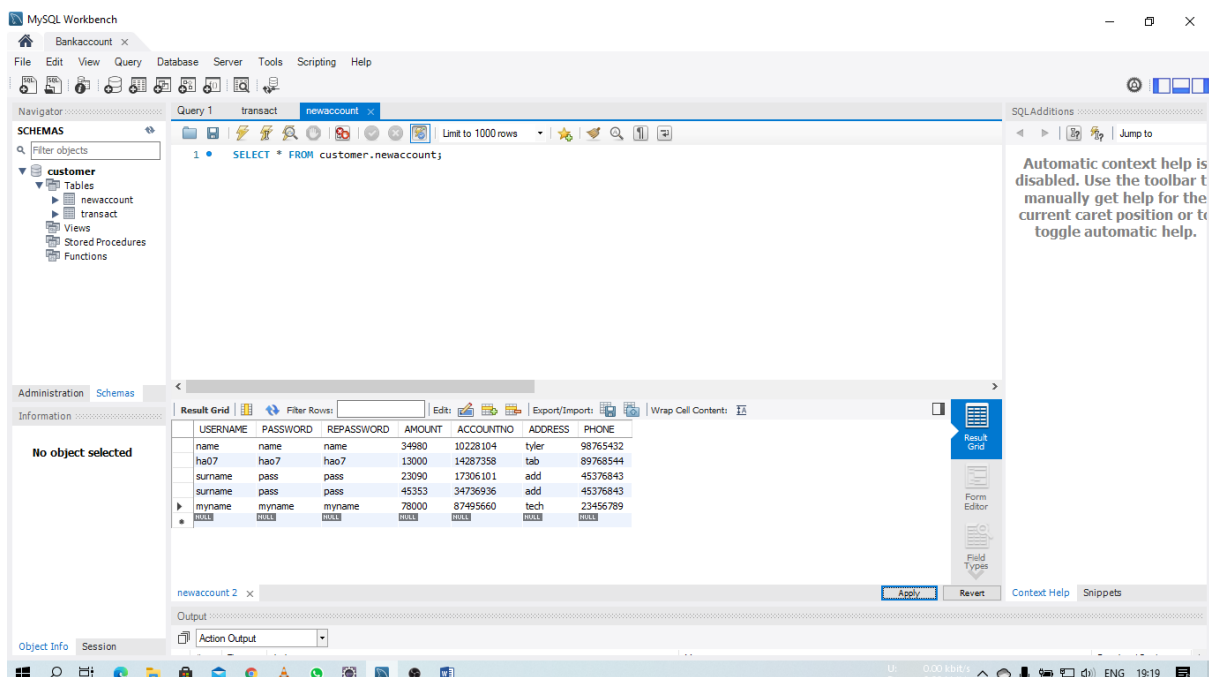
## Task 2

Every link in the navigation bar, redirects you to the page they describe. For every jsp page, the customer is required to provide their username and password. By doing this we assume that the user is already reflecting in the database. This also means that we can't use the java interface to add more users to the database. Therefore, as for provisions of creating a new account i.e. two different account numbers with the same user name, I created a file named 'create.html' which displays an interface giving the customer to register a new account.



```
87<div id="navigation">
88<ul>
89<li><a href="create.html">NEW ACCOUNT</a></li>
90<li><a href="balance.jsp">BALANCE</a></li>
91<li><a href="deposit.jsp">DEPOSIT</a></li>
92<li><a href="withdraw.jsp">WITHDRAW</a></li>
93<li><a href="transfer.jsp">TRANSFER</a></li>
94<li><a href="report.jsp">REPORT</a></li>
95<li><a href="closeacc.jsp">CLOSE A/C</a></li>
96</ul>
97</div>
98
99
100<table style="width:897px; background:#FFFFFF; margin:0 auto;">
101<tr>
102<td>
103<div width="1200" valign="top">
104<div id="welcome" style="border-right: 1px dotted #666666; height: 100px; text-align: center; background-color: white;">
105<h1>OPEN ACCOUNT FORM</h1><br>
106<table align="center" bgcolor="white">
107<tr>
108<td>
109<form name="F1" onSubmit="return dil(this)" action="CreateServlet" method="post">
110<table cellpadding="5" cellspacing="3">
111<tr><td>USER NAME:</td><td><input type="text" required name="username"/></td></tr>
112<tr><td>PASSWORD:</td><td><input type="password" required name="password"/></td></tr>
113<tr><td>CONFIRM-PASSWORD:</td><td><input type="password" required name="repassword"/></td></tr>
114<tr><td>AMOUNT:</td><td><input type="text" name="amount"/></td></tr>
115<tr><td>ADDRESS:</td><td><input type="text" required name="address"/></td></tr>
116<tr><td>PHONE:</td><td><input type="text" required name="phone"/></td></tr>
117<tr><td><td><input type="submit" value="Submit"/></td></tr>
118<tr><td><td><input type="reset" value="CLEAR"/></td></tr>
119</table>
120</form>
121</td>
122</tr>
123</table>
124</div>
125</td>
126</tr>
</table>
```

The screenshot below shows, the customer (surname) has two accounts i.e. with different account numbers, all in the same database.



```
Query 1
1 • SELECT * FROM customer.newaccount;
```

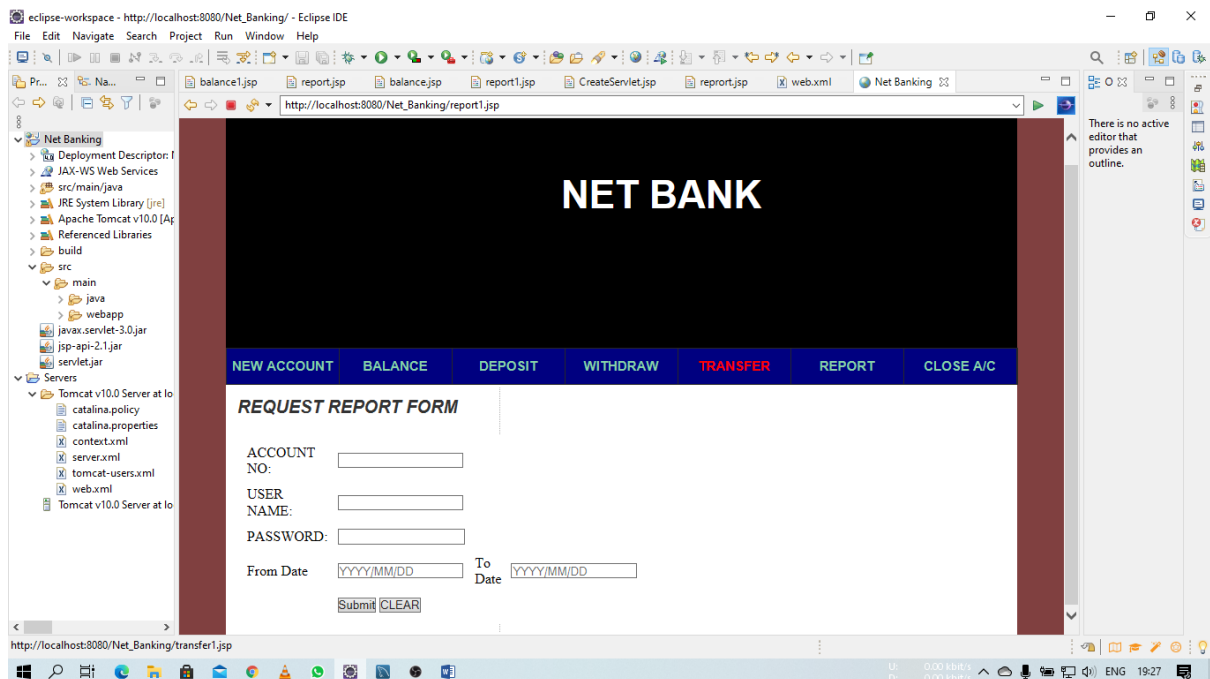
| USERNAME | PASSWORD | REPASSWORD | AMOUNT | ACCOUNTNO | ADDRESS | PHONE    |
|----------|----------|------------|--------|-----------|---------|----------|
| name     | name     | name       | 34980  | 10228104  | tyler   | 98765432 |
| hao7     | hao7     | hao7       | 13000  | 14287358  | tab     | 89765444 |
| surname  | pass     | pass       | 23090  | 17306101  | add     | 45376843 |
| surname  | pass     | pass       | 45353  | 34736936  | add     | 45376843 |
| myname   | myname   | myname     | 78000  | 87495660  | tech    | 23456789 |

### Task 3.

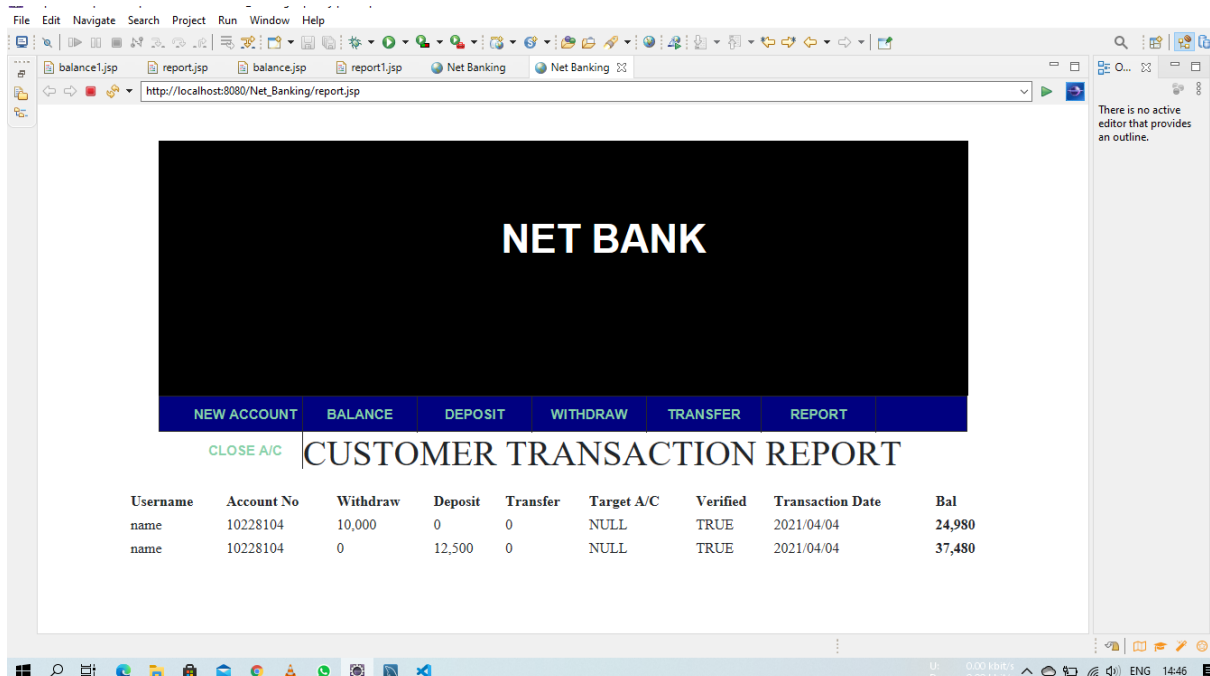
I created a java class "netbank.jpa.simple". I went further to create a folder "libs" where I added the required JPA jars and derby.jar for my database into this folder. I went further to add the libs to the project classpath. This however threw in errors which I think were caused by the web.xml file in the tomcat server. This made me delete the class file since the rest of the program worked just fine.

### Task 4.

In this task, we added a 'REPORT' tab on the navigation bar. This takes the customer to file report1.jsp which contains login prompt, the account number for which they want a report from and also prompts for date intervals.



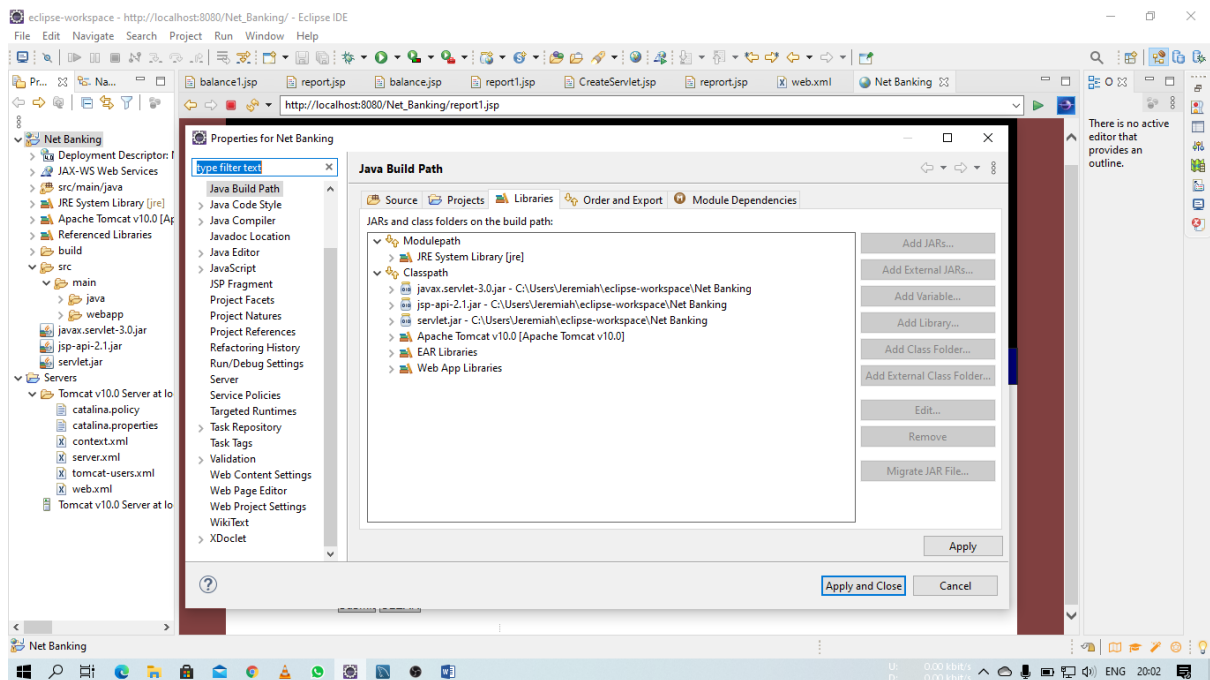
On filling out the form, data from the database about the requested account is processed and provided to the user. This report is comprehensive and contains all transactions done by the customer within a given time interval. It also shows the balance in the account at the end of each day.



From the above screenshot, the customer under the username 'name', conducted two transactions between 1<sup>st</sup> and 5<sup>th</sup> April 2020. His initial amount recorded in the database was 34,980 and on carrying out the first transaction which was a withdrawal, the amount decremented to 24,980 and on the second transaction which was a deposit, the amount increased to 37,480.

## Task 5

Some of the technologies incorporated include Java Servlet Technology, Java server faces (JSF), Java server Page (JSP), and servlet-api. Some these were added into the module and classpath as external jar files.



## Java servlets

Servlets help the programmer define unique servlet classes and extend server capabilities for hosting request-response applications. Servlets respond to most requests, which makes them extensive in applications hosted by web servers.

## Java Server Pages

JSP pages helped me in putting servlet code directly in the form of a text document. In my project, I used static text data which included HTML5 text and XML. I also used JSP elements which helped to determine how the page will construct dynamic content.

## Connector.

To connect my project with a database, I incorporated 'MySQL connector j' which acts as a pathway between the web project and my SQL workbench.

## Java Server Faces.

JSF stands for building component-based, event-oriented web interfaces. JSF allows access to server and logic. JSF is an XML document which represents formal components in a logical tree. The main reason I used JSF is that it offers clean separation of behavior and presentation, robust event handling mechanism, and offers multiple, standardized vendor implementations.

### E. Section 3

Java object oriented programming contains of four main concepts which are; Encapsulation, polymorphism, abstraction and inheritance. There are however other relationships such as aggregation and composition.

#### Encapsulation.

*“Encapsulation is a mechanism where a programmer binds java data and code together to form a single unit”* (H. Schildt, 2018). I have used encapsulation mostly in the customer.java file where I have created a class customer which has a private variables such as username, account no, email, and phone no and some other few. I have then created a getter and setter methods through which we can get and set the name of an employee. Through these methods, any class which wishes to access the name variable has to do it using these getter and setter methods.

#### Abstraction.

*“Abstraction is the mechanism of hiding complexity of a program to the user, so that the user sees only the relevant details”* (L. Cardelli., & P. Wegner, 1985). Abstraction can either be in form of an abstract class or an interface class. For instance, when a customer comes to the welcome page of my application, they find a simple welcome page and a navigation bar that has tabs containing form fields. However, the complexity of this program lies within the files which are hidden. Abstraction is faster and easier to execute and is the concept that has helped me most in keeping my code dry.

#### Polymorphism

This is the ability of a variable to take up multiple forms and occurs when we have many classes that are related to each other. *“Polymorphism uses methods and attributes from other classes to perform different tasks”* (L. Cardelli., & P. Wegner, 1985).

#### Inheritance

This is the process whereby one class (subclass) acquires the properties of another class also known as the super class. Most of my subclasses in the code have the ‘extends’ keyword which means inheritance from a base class. Inheritance helped to manage code in ha hierarchical order and also keeping my code ‘dry’, especially when I had to define classes with nearly same attributes and methods as others.

#### Composition.

Composition is a situation where a class cannot exist without another class also the parent class.



## F. References

Cardelli, L., & Wegner, P. (1985). On understanding types, data abstraction, and polymorphism. *ACM Computing Surveys (CSUR)*, 17(4), 471-523.

EDureka (2021) “*Edureka training course*” Available at: [www.edureka.co](http://www.edureka.co)

Github (2021) “Github JSF” Available at: [www.github.com](http://www.github.com)

Stackoverflow (2021) “*Stack over flow*” Available at: [www.stackoverflow.com](http://www.stackoverflow.com)

Schildt, H. (2018). The complete reference Java.

W3school (2021) “W3schools Java Script & My SQL” Available at: [www.w3schools.com](http://www.w3schools.com)

Youtube (2021) “*Schedule Automated AWS EBS Snapshots*” Available at: <https://www.youtube.com/channel/UC-85Tmx8lhNZQmBNbxZiqkw>

## **G. Read Me.**

The following is a list of items used;

1. Java Development kit 14
2. Eclipse IDE
3. Java Enterprise Edition.
4. Apache tomcat server. Version 10
5. MySQL workbench. Version 8.0

### **External libraries and jars**

6. MySQL connector j 8.0.23
7. Javax.servlet-3.0.jar
8. Jsp-api-2.1.jar
9. Servlet-api.jar

### **How to run.**

Once the above files have been installed and jars added to module and classpaths, you can run the project directory on the server.