Title: Taxi Driver System

Author: Usman Basharat

Date: 24th March 2019

Course: COMP-1610 Programming Enterprise Component

Word Count: 4449

Table of Contents

Introduction

In this report, I will be evaluating the Speedy Taxi application. A number of requirements were made to improve the current Speedy Taxi application that has been made. Within this report, I will be designing documentation such as Entity Relationship Diagram (ERD), UML Class Diagram and Use Case Diagram for this application. I will be explaining the functionality of the features that have been included. Alongside this, I will be evaluating the whole application to see what needs to be improved. Lastly, I will be carrying out Java EE research.

Design Documentation

This section of the report, I will be building UML design documentation for the Speedy Taxi of what to expect for this application. I will include Entity Relationship Diagram, UML Class Diagram and UML Use Case for this application. This will be the design element of this report.

Entity Relationship Diagram

Referring to Figure 1, this shows the Entity Relationship Diagram that will be used within Speedy Taxi database. This denotes the breakdown of primary key, columns and data types that each table has with each other. This database directly reflects the Entities that are shown in Figure 30.

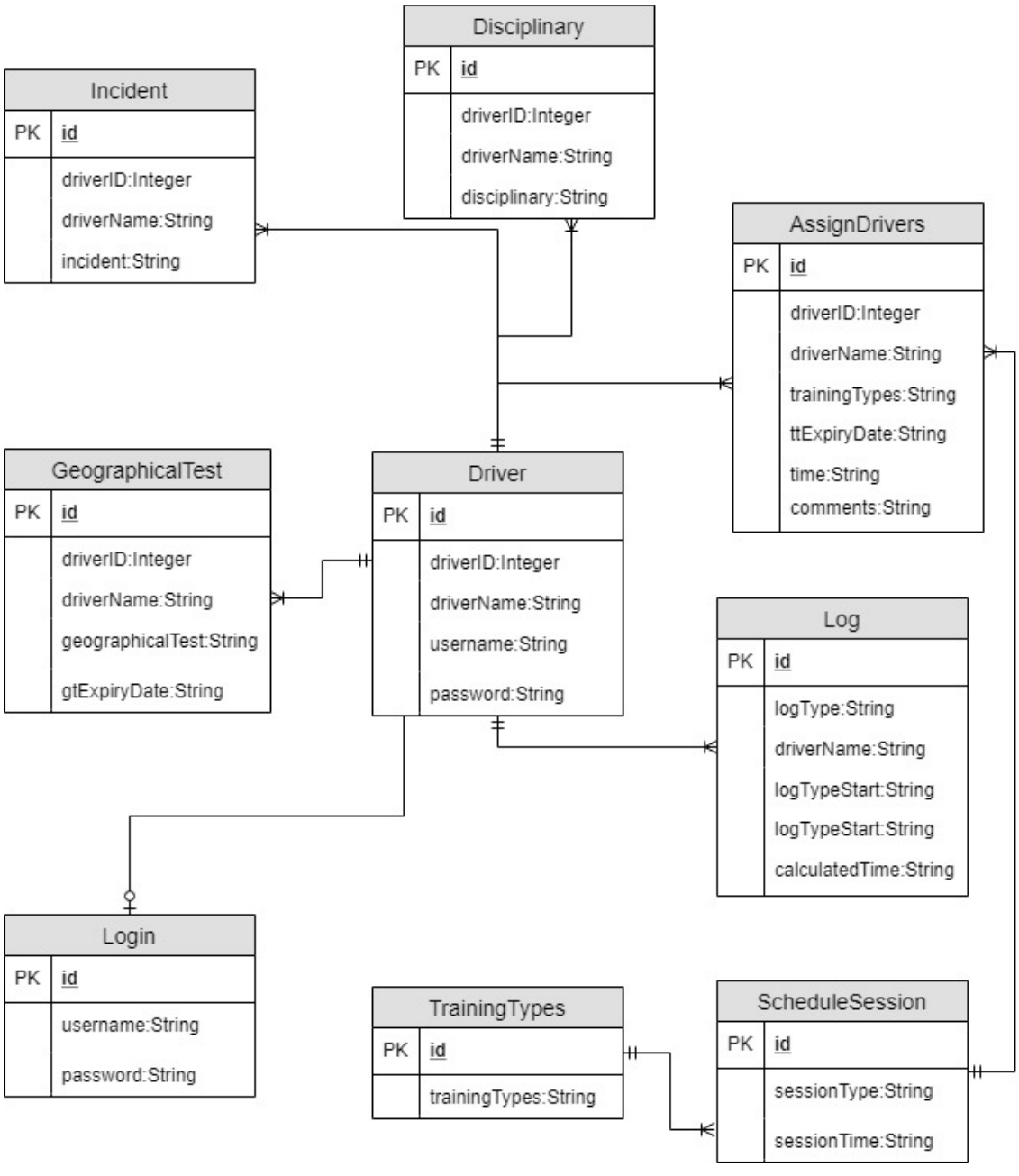


Figure 1 shows the Entity Relationship Diagram (ERD) for SpeedyTaxi

UML Class Diagram

Referring to Figure 1, this shows the UML Class Diagram for the Speedy Taxi Application. This is a general Class Diagram that will be explained. User Interfaces are the all the JSP files. Servlets are the Java files. I plainly stated this as a general for the UML Class Diagram not to be too long. However, the main outcome of this is the relationship for both Servlets and User Interface are a must have for both. This is the same with HibernateInterface is an interface that HibernateSingleton implements. They are more methods that are used. However, only one is used for the same reason as User Interface and Servlets. Please notice that this is just a general UML Class Diagram for the purpose of the relationship of each other. This has been verified by Markus Wolf.

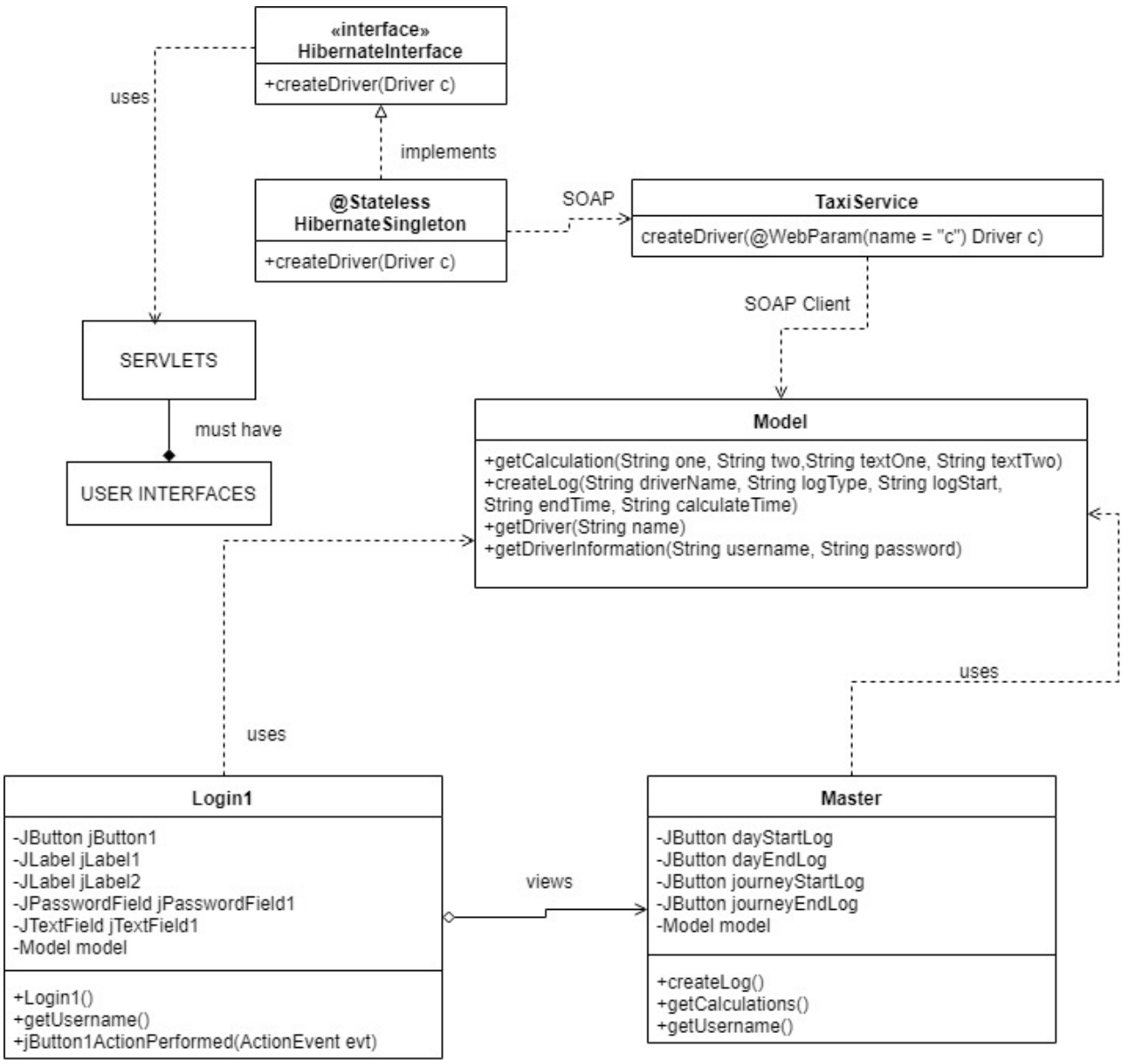


Figure 2 shows the UML Diagram for the Speedy Taxi.

UML Use Case

Referring to Figure 3, this is the Use Case all the actors within the system. This also provides the overall concepts of the system.

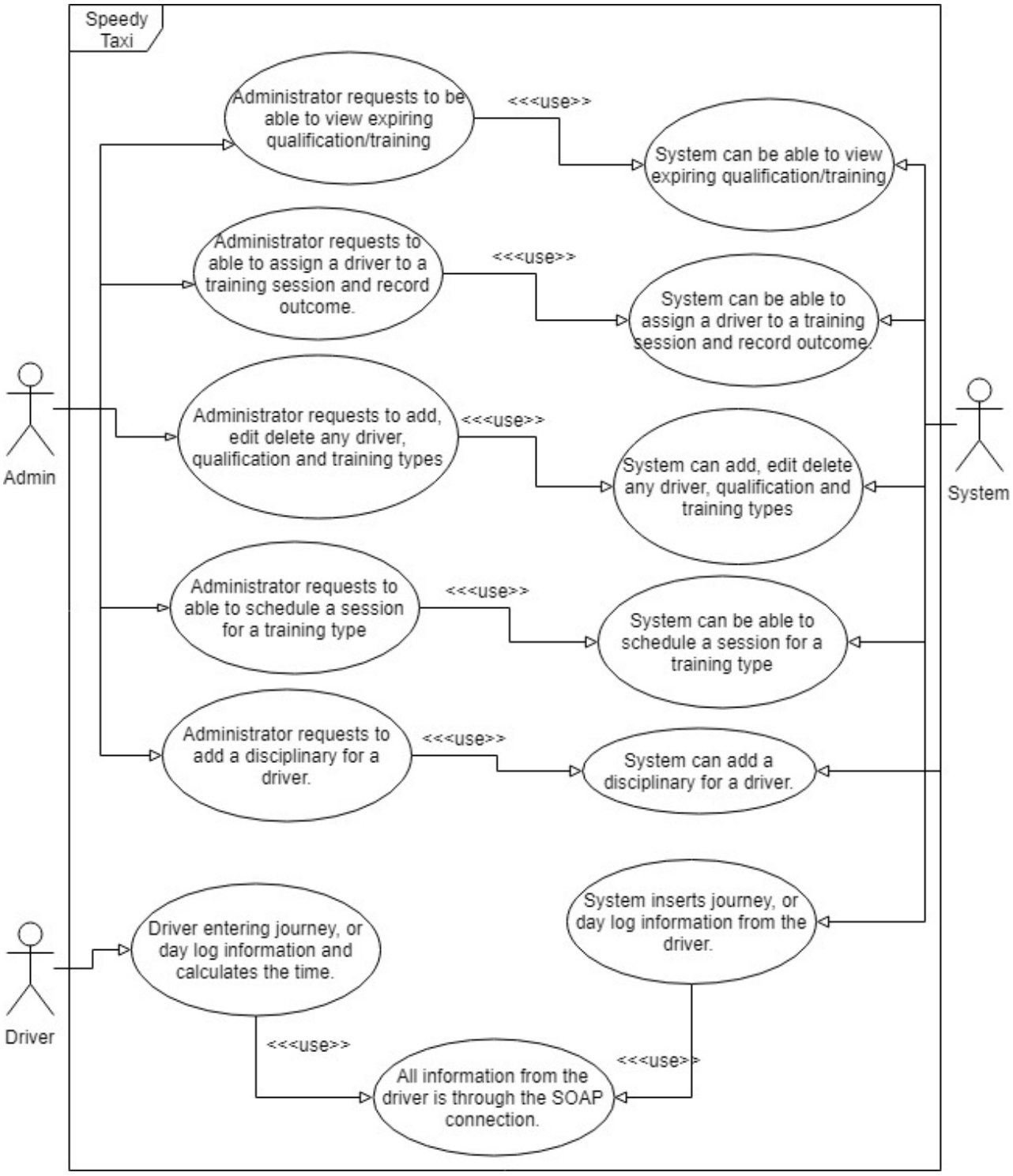


Figure 3 shows the Use Case Diagram in Speedy Taxi application.

Screenshot of Features

This is split into three functionalities of basic, intermediate and advanced. Basic Functionality includes a Java EE Website for the administrator to be able to login; add, edit and delete drivers, qualifications, training types; search driver with a profile including qualifications and training; and a list of drivers who have qualifications that are expiring within the next 30 days. This is only for basic functionality for the administrator-side of the application.
Intermediate Functionality involves using SOAP to connect the two applications together for communication. In addition, this is a desktop application that logs the day and journey from start and day. This is for the driver to log information in. All of this information needs to connect through to the SOAP to the Website and communicate through this in order to save it into the database. Advanced Functionality includes to tweaking different aspects of the application to improve this.

Speedy Taxi Website

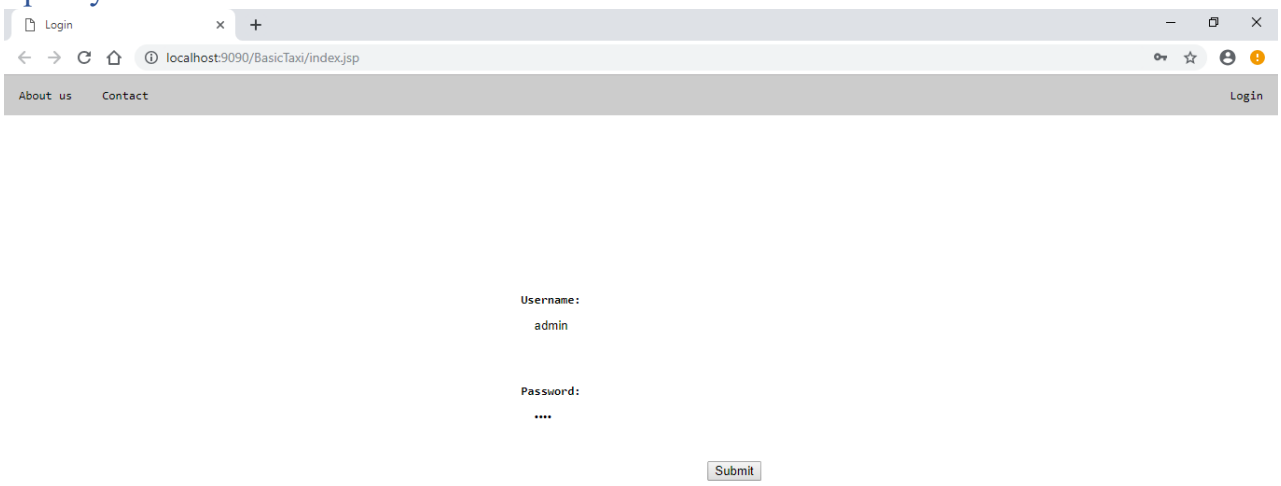


Figure 4 shows administrator login for the Speedy Taxi application.

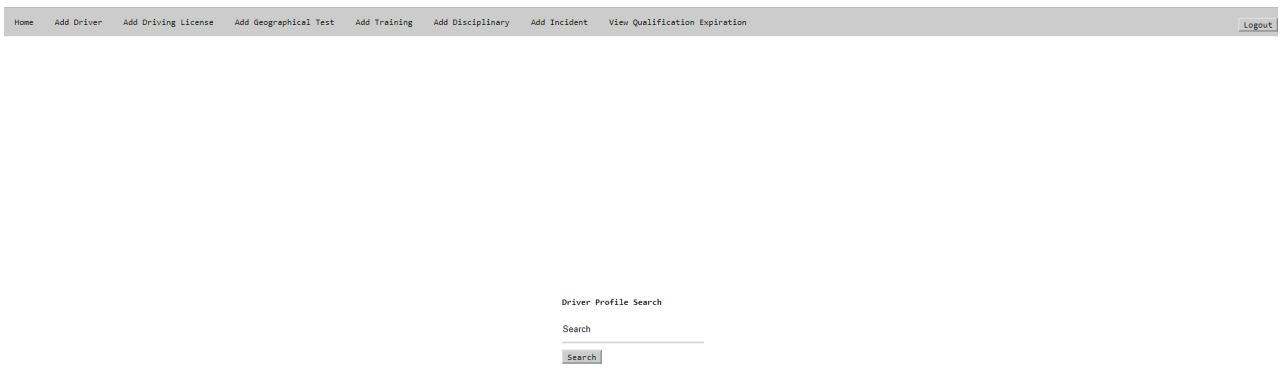


Figure 5 shows the main screen for the administrator

Referring to Figure 4, the administrator is expected to fulfil in the username and password for this login to authenticate. This is running through the database as both are expected to match in order to get to Figure 5. Referring to Figure 5, this is where the administrator has variety of options to explore.

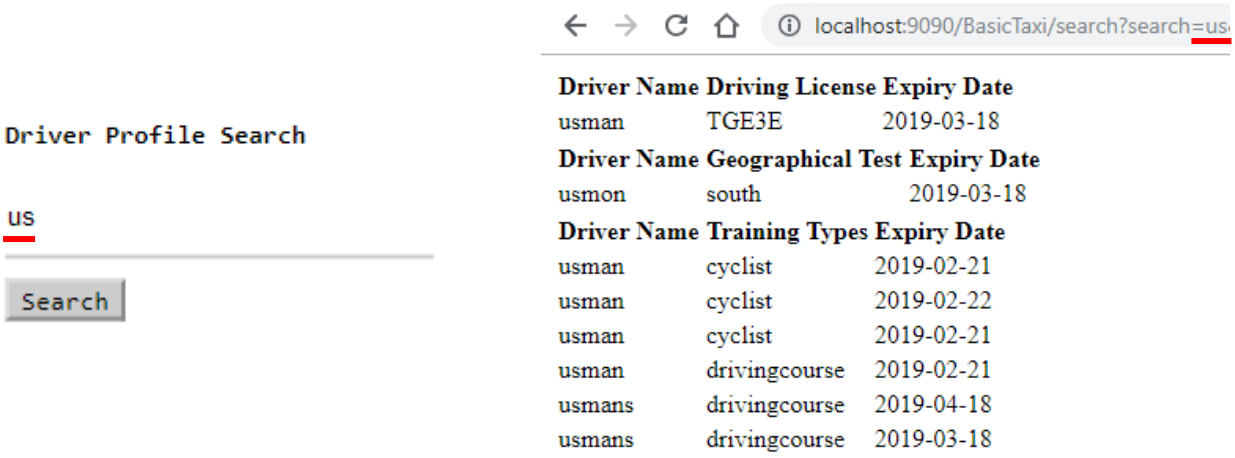
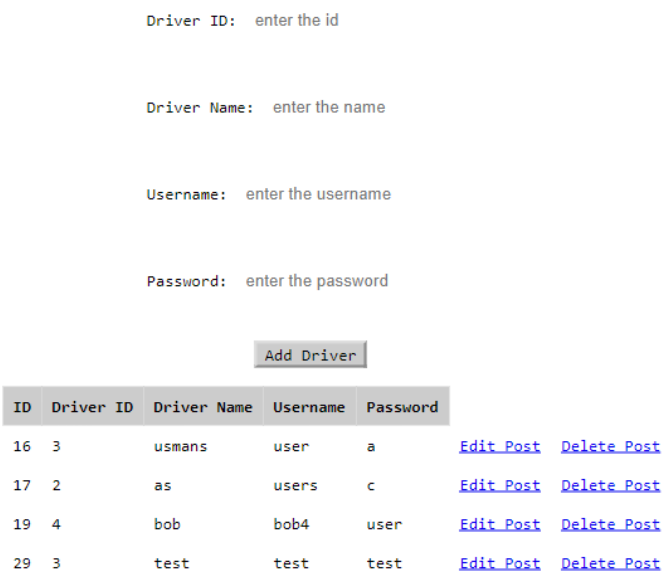


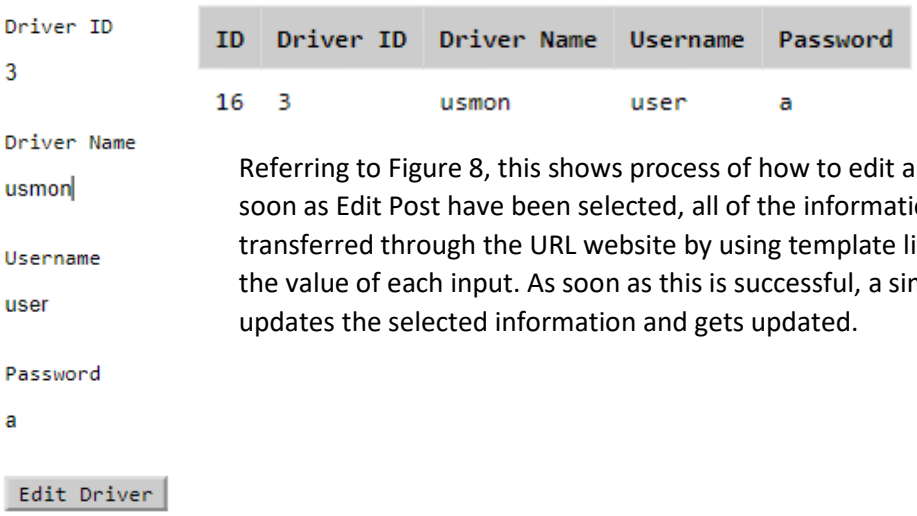
Figure 6 shows the driver profile search alongside its results.

Referring to Figure 6, this shows the profile of the qualifications and training for the searched driver. Regular Expressions are used to match if the driver's name contains any of what has been searched. As you can see, "us" is underlined in red as this shows what has been typed into the search bar. This brings up all of driver that has "us" within qualifications and training types within the Speedy Taxi database.



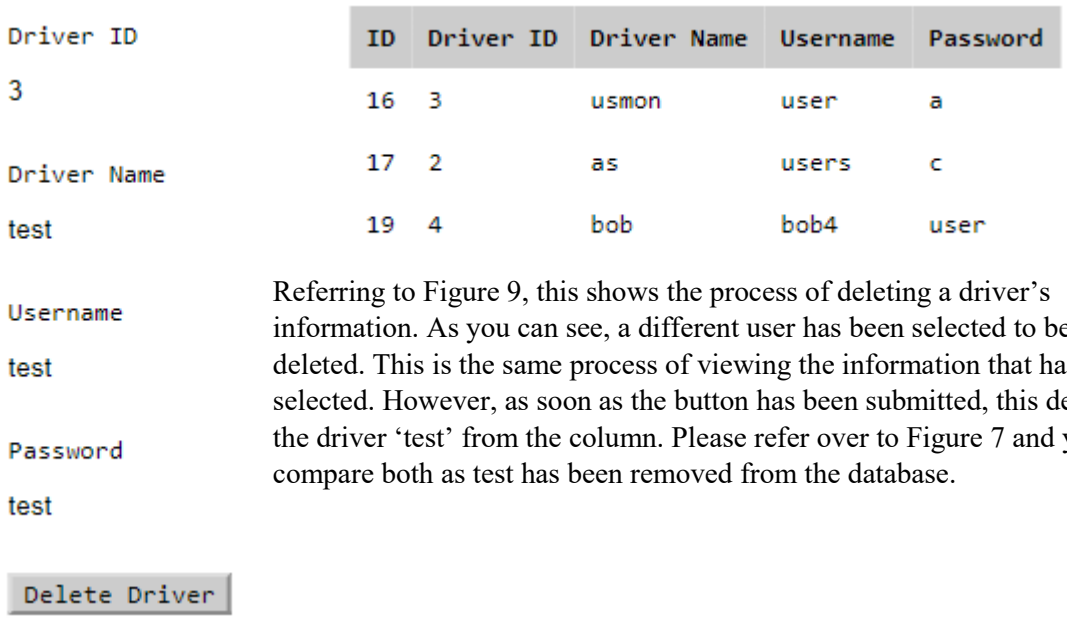
Referring to Figure 7, this shows the process of adding a driver into the system. This is simple of as shown on the image. Username and Password has been included for the only purpose of the user can log into the desktop application. This is entered alongside each user to distinguish of each user. Once this has been entered, administrator has the option to edit and delete each driver as they wish too. Edit and Delete is alongside to see which driver has been selected. As you can see on the table, this information shows that these have been added.

Figure 7 shows the process of adding a driver into the system with the option of editing and deleting it.



Referring to Figure 8, this shows process of how to edit any driver's information. As soon as Edit Post have been selected, all of the information that has been selected gets transferred through the URL website by using template literals to denote this through the value of each input. As soon as this is successful, a simple UPDATE statement updates the selected information and gets updated.

Figure 8 shows the driver can edit the post.



Referring to Figure 9, this shows the process of deleting a driver's information. As you can see, a different user has been selected to be deleted. This is the same process of viewing the information that has been selected. However, as soon as the button has been submitted, this deletes the driver 'test' from the column. Please refer over to Figure 7 and you can compare both as test has been removed from the database.

Figure 9 shows the delete option for driver.

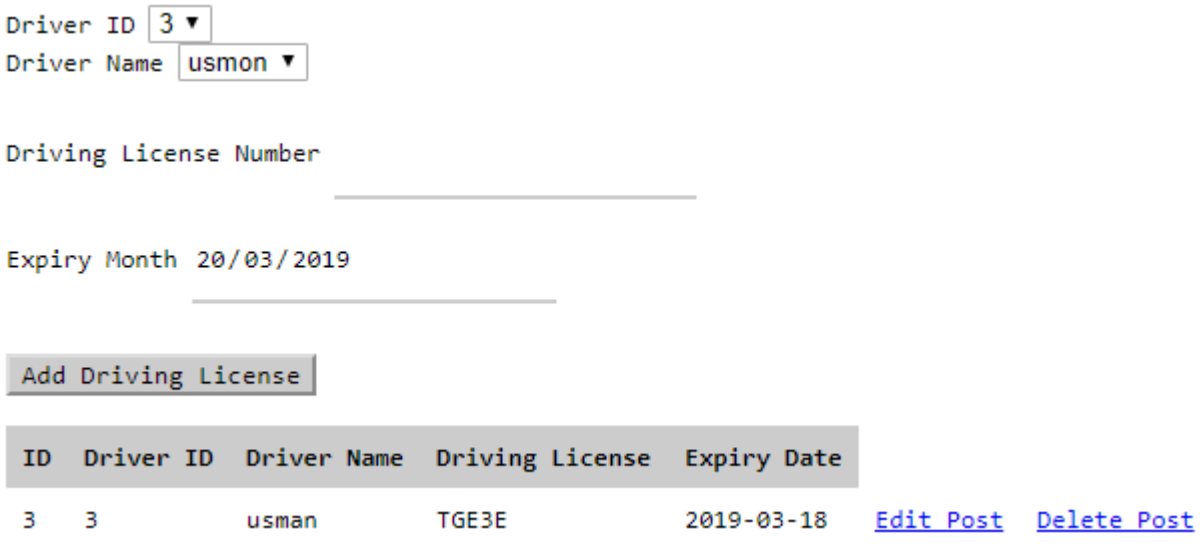
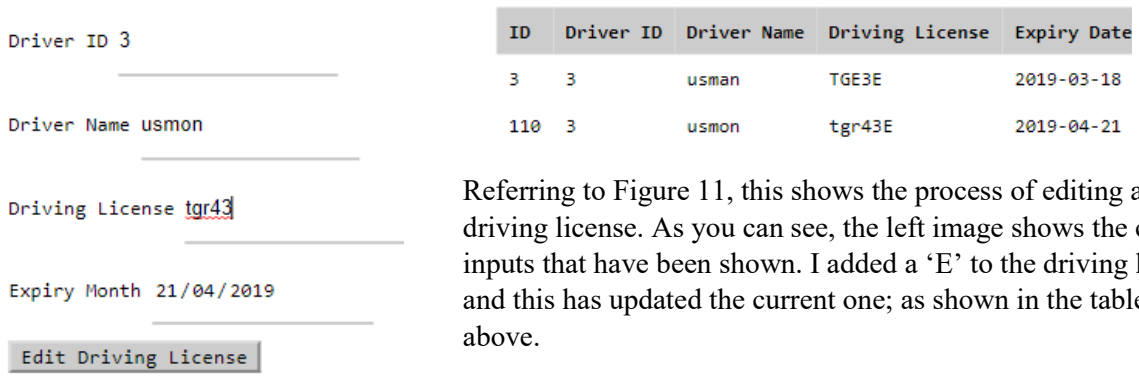


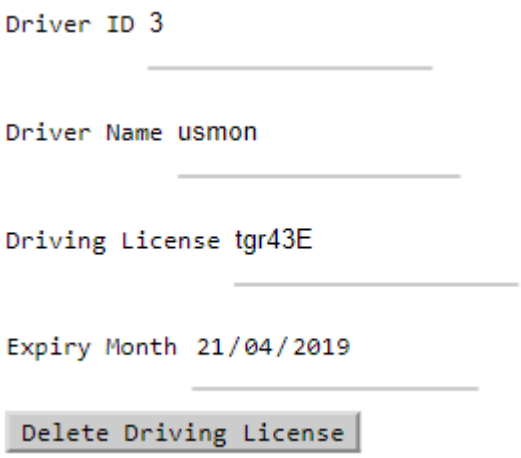
Figure 10 shows to add driving license as a qualification.

Referring to Figure 10, this shows the process of how to add a driving license within the Speedy Taxi website application. This is the same process of how to add, edit and delete a driver too.



Referring to Figure 11, this shows the process of editing a new driving license. As you can see, the left image shows the current inputs that have been shown. I added a 'E' to the driving license, and this has updated the current one; as shown in the table above.

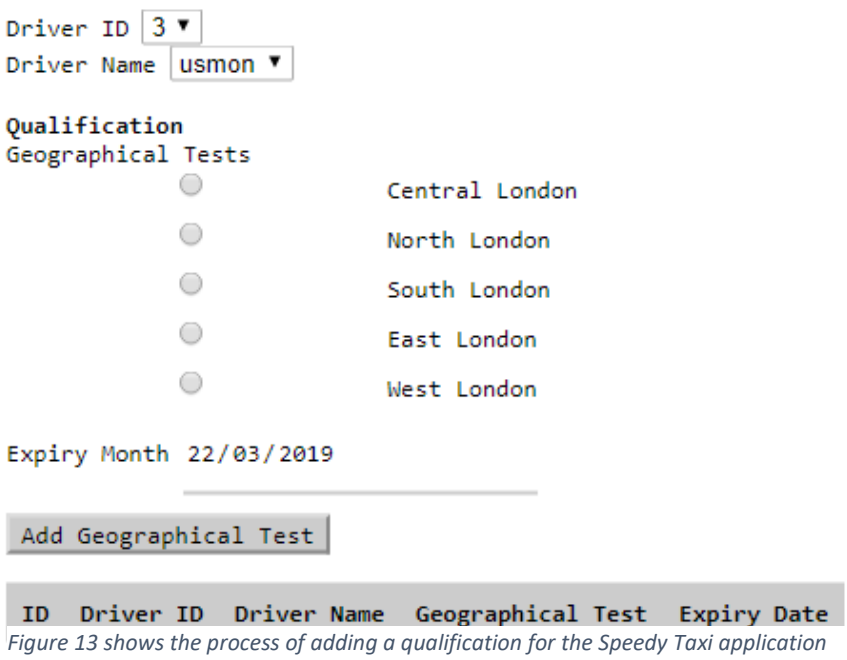
Figure 11 shows the process of editing the qualification for the Speedy Taxi application



ID	Driver ID	Driver Name	Driving License	Expiry Date
3	3	usman	TGE3E	2019-03-18

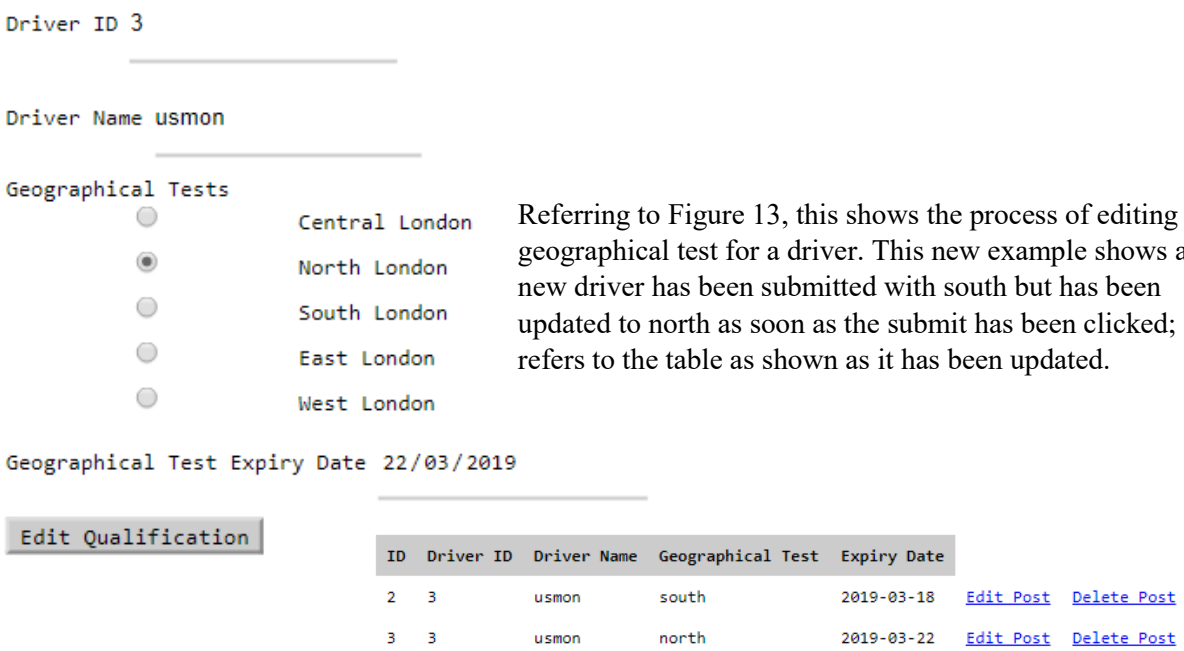
Referring to Figure 12, this is the same process for any of the above that has been shown. The left image is shown as the selected driving license that needs to be deleted off the system. As soon as the deleted button has been clicked, the same image above has been selected.

Figure 12 shows the process of deleting a qualification for the Speedy Taxi application



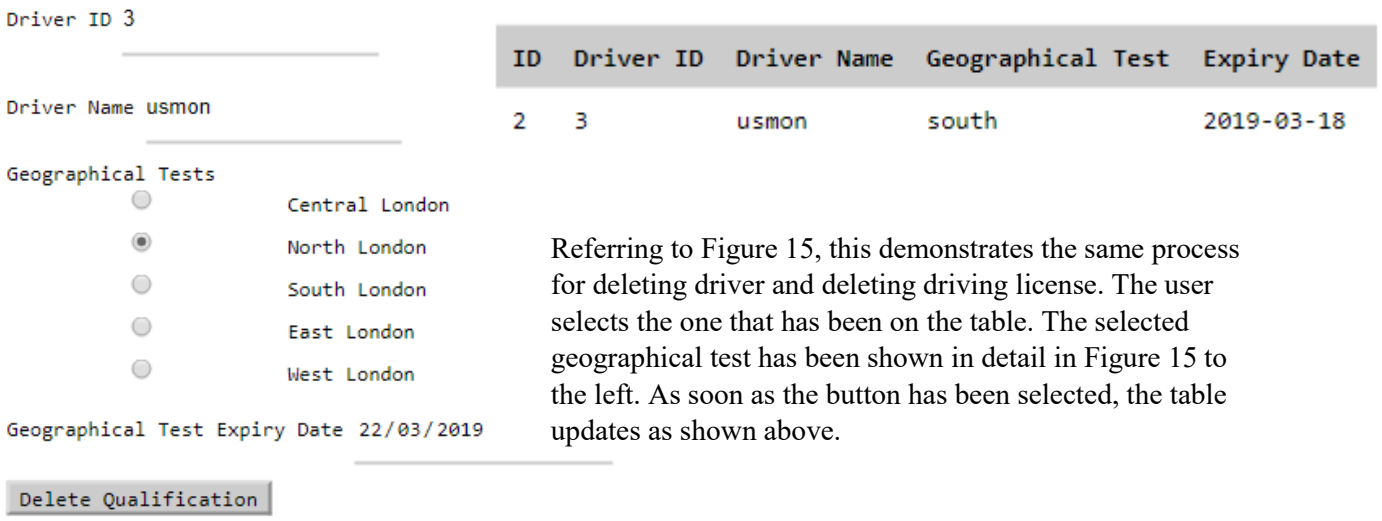
Referring to Figure 14, this demonstrates the process of adding a Geographical Test for each user. The selected tests have been shown for the user to select which wants are needed. As soon as these details are filled out, they are shown on the table below. As you can see this in Figure 14, an example has already been shown.

Figure 13 shows the process of adding a qualification for the Speedy Taxi application



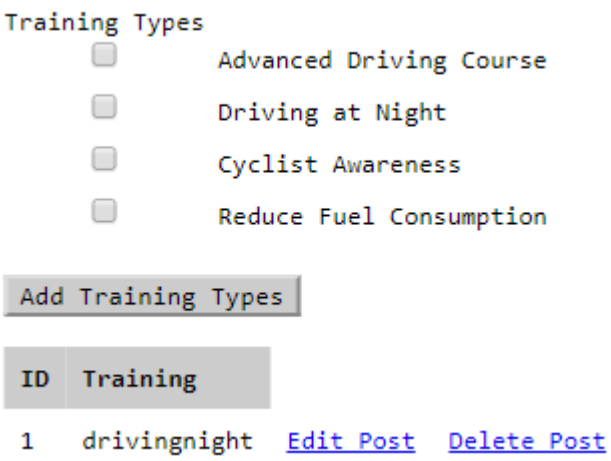
Referring to Figure 13, this shows the process of editing a geographical test for a driver. This new example shows a new driver has been submitted with south but has been updated to north as soon as the submit has been clicked; it refers to the table as shown as it has been updated.

Figure 14 shows the process of editing a qualification for the Speedy Taxi application.



Referring to Figure 15, this demonstrates the same process for deleting driver and deleting driving license. The user selects the one that has been on the table. The selected geographical test has been shown in detail in Figure 15 to the left. As soon as the button has been selected, the table updates as shown above.

Figure 15 shows the process of deleting a qualification for the Speedy Taxi application.



Referring to Figure 16, this shows the process of adding training type for this application. As soon as the checkbox has been selected, the training type is available to be scheduled. The process of editing and deleting is the same process as shown above. However, Figure 17 and Figure 18 show the process for this anyway just for completion version. Figure 17 shows the ID 1 being selected and editing from driving at night to cyclist awareness. Referring to Figure 18, a new training type cyclist awareness has been added. This was selected and deleted to show that this can be completed too.

Figure 16 shows the process of adding a training type for the Speedy Taxi application.

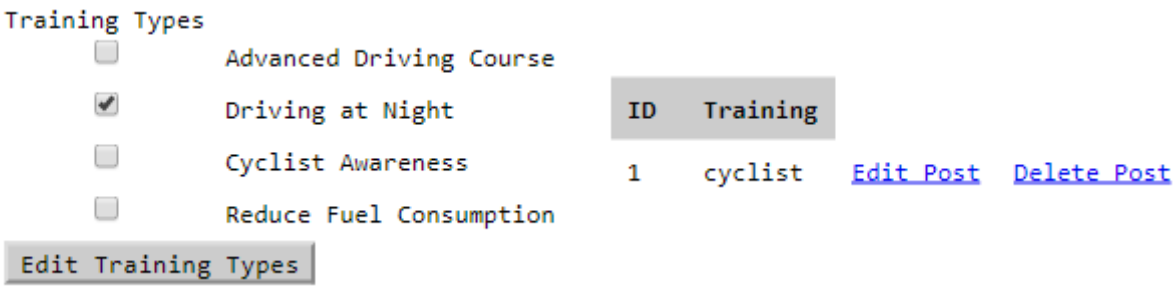


Figure 17 shows the process of editing a training type.

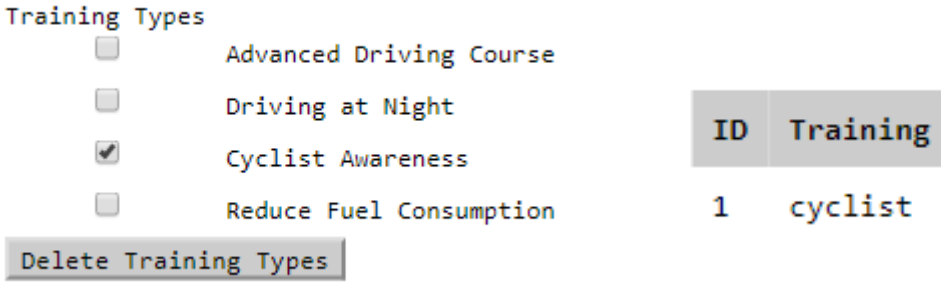
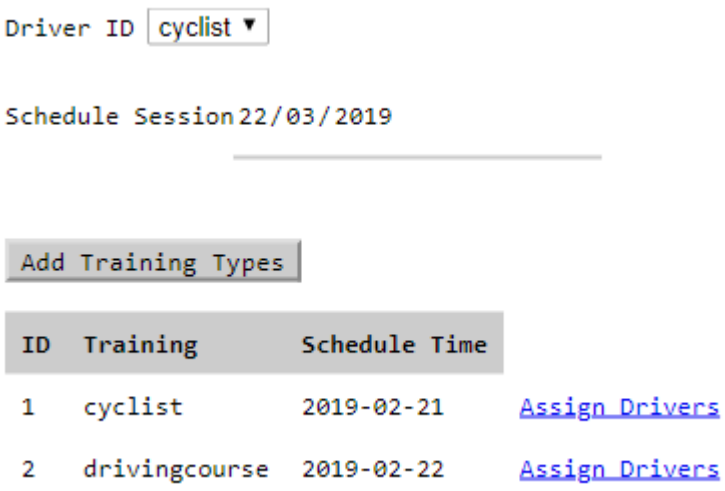


Figure 18 shows the process of deleting a new training type being added.



Referring to Figure 19, this shows the process of scheduling a session alongside a scheduled session for this event. As you can see in Figure 19, this shows that two events have been added. As soon as this is completed, you can assign drivers to this course of event. Figure 20 demonstrates how assign drivers works.

Figure 19 shows the process of scheduling a session for the application

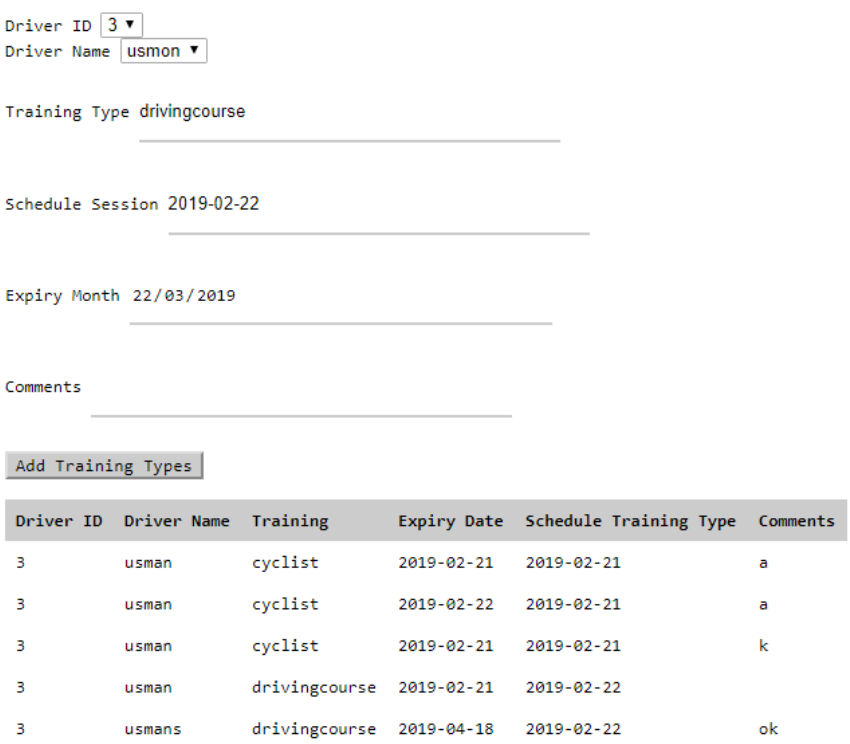


Figure 20 shows the assign drivers process for the application.

Referring to Figure 20, this demonstrates the process of assigning drivers. This shows that once the scheduled session has been clicked, the course and session is already set. All the admin needs to do is add the expiry date, selected driver and comments on this. As you can see in Figure 20, this shows some examples of already drivers that have been added onto this system.

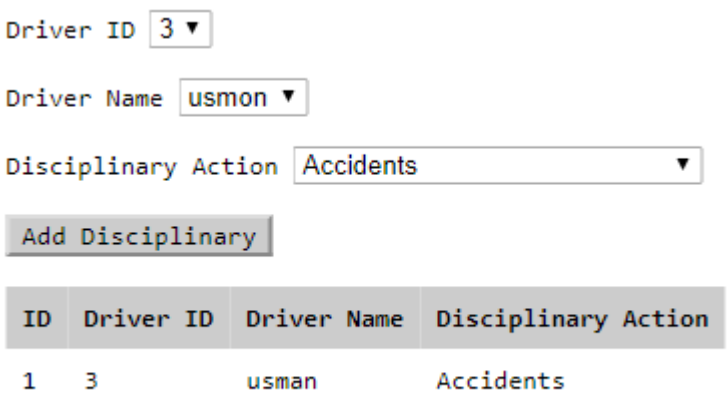
ID	Driver ID	Driver Name	Driving License	Expiry Date

ID	Driver ID	Driver Name	Training	Expiry Date	Schedule Training Type	Comments
6	3	usmans	drivingcourse	2019-04-18	2019-02-22	ok

ID	Driver ID	Driver Name	Geographical Test	Expiry Date

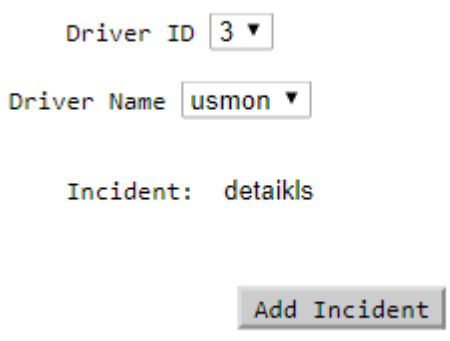
Figure 21 shows the view expiration date for the next 30 days.

Referring to Figure 21, this shows the expiration date for next 30 days. As you can see, the current training type that has been added has a date selected on the 18th April 2019. From the current date, 22nd March 2019, this should be valid and shown. However, all of the others from other figures above are not shown as they are invalid in the bracket. This shows that the current expiration date for the next 30 days is valid. These are three tables and they work for all three of these.



Referring to Figure 22, this shows how to add disciplinary for each driver. This is for the advanced part of the report as this is displayed through the database too for the admin to view.

Figure 22 shows the process of adding disciplinary to each driver for the application



Referring to Figure 23, this demonstrates the process of recording an incident and sending a message whilst this has been recorded through JMS. I only managed to get this to be sent. I did not manage to get the message to be retrieved by the user. However, Figure 25 shows the connections that have been made for this to be half-completed. This will be evaluated thoroughly in the Evaluation section for this.

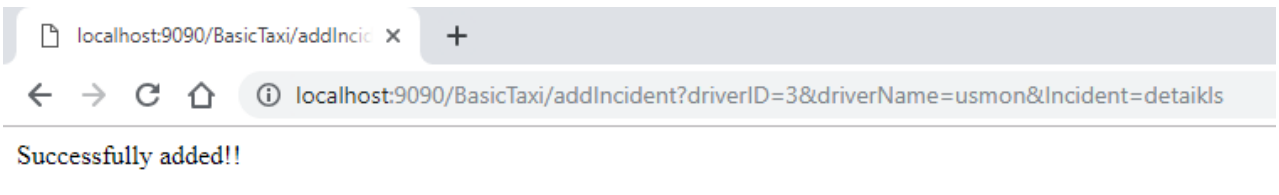


Figure 23 shows the process of sending JMS message when an incident has been recorded.

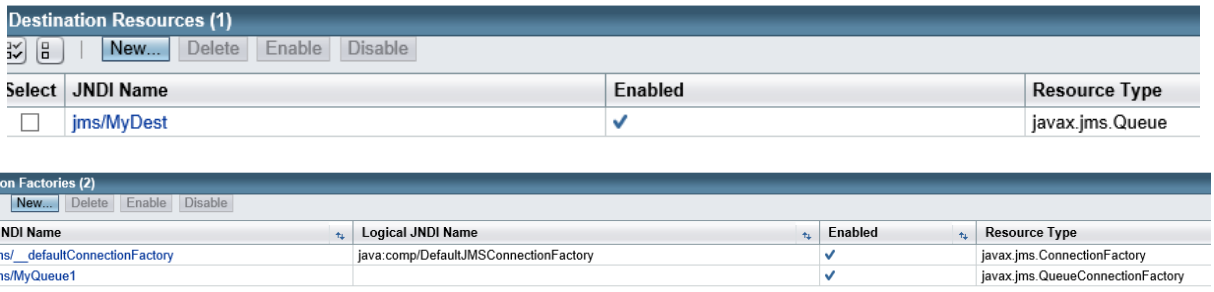


Figure 25 shows the JMS connections that have been made for this to be possible.

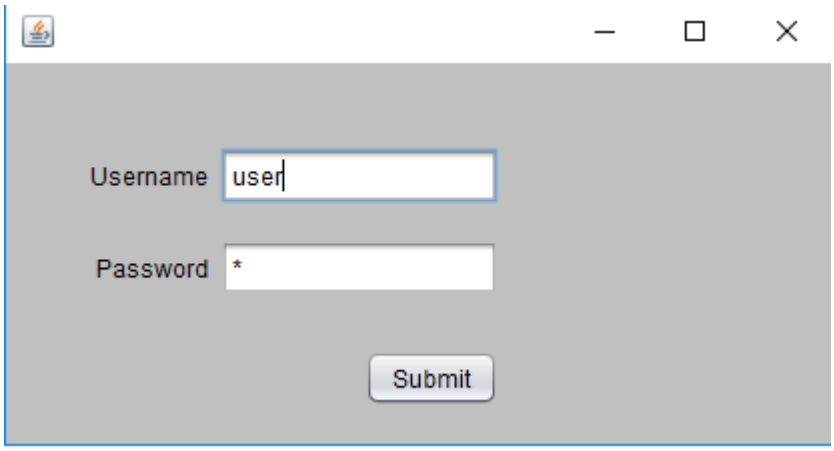
```
import javax.jws.WebService;
import javax.ejb.Stateless;
import javax.jws.Oneway;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author ub2232e
 */
@WebService(serviceName = "TaxiService")
@Stateless()
public class TaxiService {

    @EJB
    private HibernateInterface ejbRef;// Add
    // "Insert Code > Add Web Service Operat

    @WebMethod(operationName = "getID")
    public Log getID(@WebParam(name = "drive
        return ejbRef.getID(driverName);
    }

    @WebMethod(operationName = "getIDName")
    public Log getIDName(@WebParam(name = "d
        return ejbRef.getIDName(driverName,
    }

    @WebMethod(operationName = "updateLog")
    @Oneway
```

```
import java.time.temporal.ChronoField;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.Stateless;
import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;

/**
 *
 * @author ub2232e
 */
@Stateless
public class HibernateSingleton implements H

    private static HibernateSingleton firstI

    protected HibernateSingleton() {

    }

    public static HibernateSingleton getInst
        if (firstInstance == null) {
            firstInstance = new HibernateSin
        }
        return firstInstance;
    }

    @Override
    public void createDriver(Driver c) {
        Session session = HibernateUtil.getS
        session.beginTransaction();
```

Figure 24 shows the process database connection through SOAP on the Speedy Taxi Website.

Referring to Figure 24, this shows the connection the SOAP has on the Speedy Taxi Website. This shows that the database class on the right-hand side, HibernateSingleton, has been converted to a Stateless Bean for this to happen. Once this has been completed, a SOAP connection is automatically generated by NetBeans and this connection has made Taxi Service on the left-hand side of Figure 24. This SOAP connection is ready for the Web Service Client to use. Please note that all connections that have been made through the database has been all alongside the use of Hibernate. Any verification can be made by referring to the HibernateSingleton class.

Driver Desktop Application



Referring to Figure 26, this demonstrates the login for each driver to authenticate into the application. Each driver has different login that must be authorised. Please refer yourself over to Figure 9 as this demonstrates where each login needs to match.

Figure 26 shows the login for the driver desktop application.

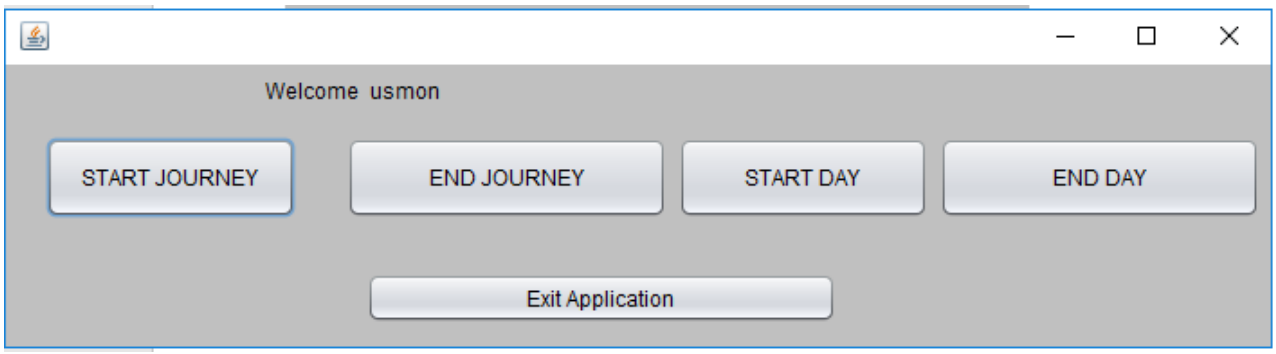


Figure 27 shows the four log components for each user.

Referring to Figure 27, as soon as the login is correct, this main screen will show. As you can see, the username, "user" logs into the system. This shows the user's driver name and tells us which driver has logged into the system. Four log components are ready. Figure 28 shows the that if END Journey or END Day has been selected before START, it wouldn't let the user do this before anything has happened. However, as soon as START and END has been pressed, it enters the database and informing the user that the information has been logged into the system.

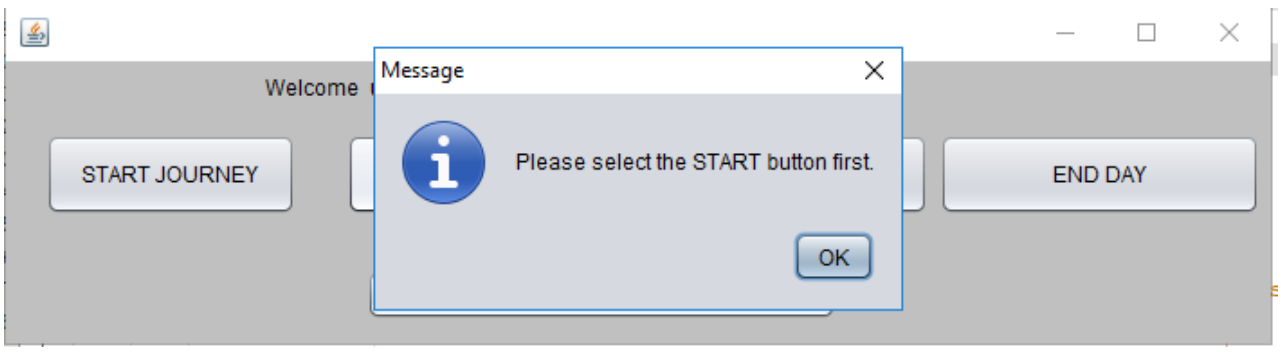


Figure 28 shows the error of pressing end first.

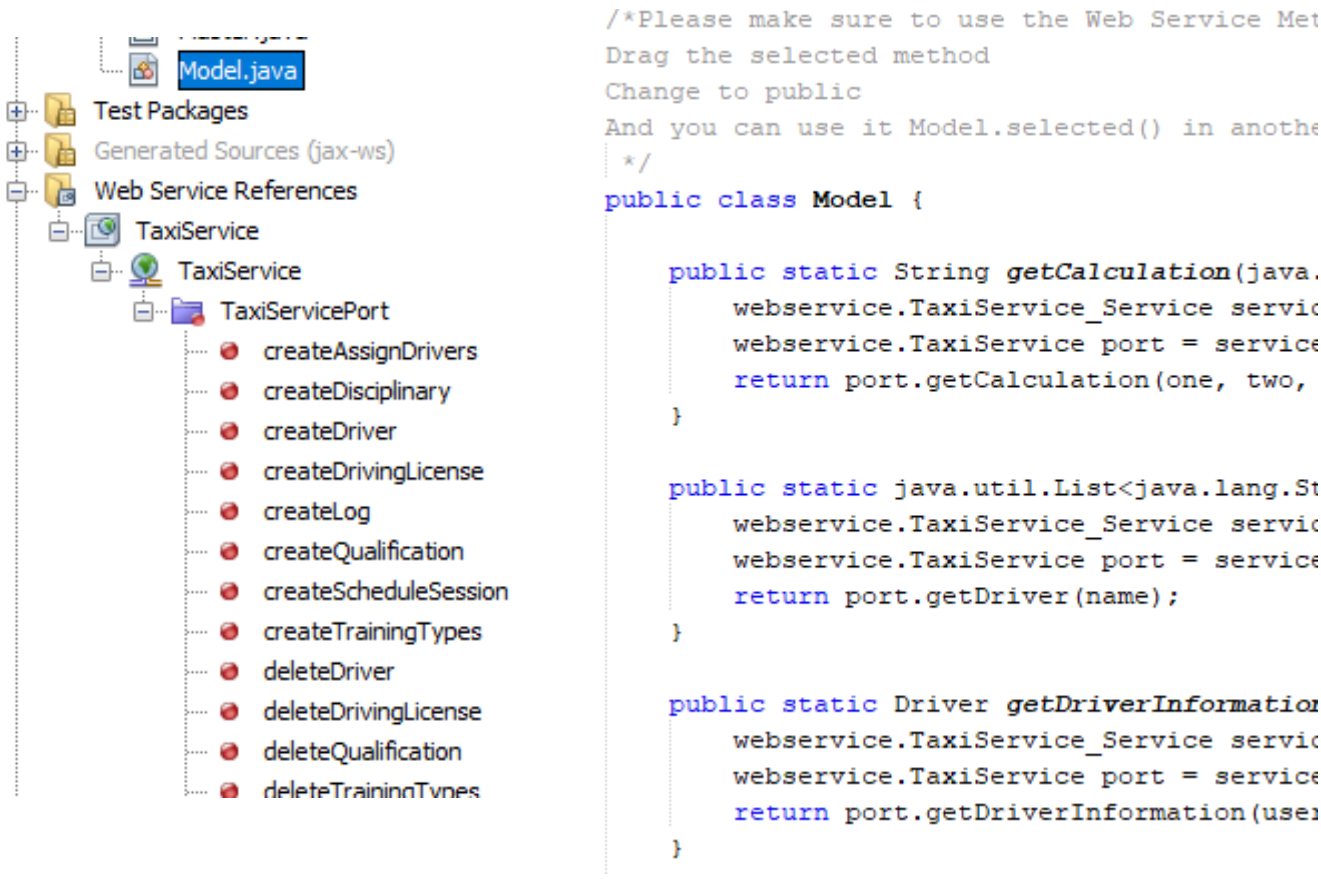
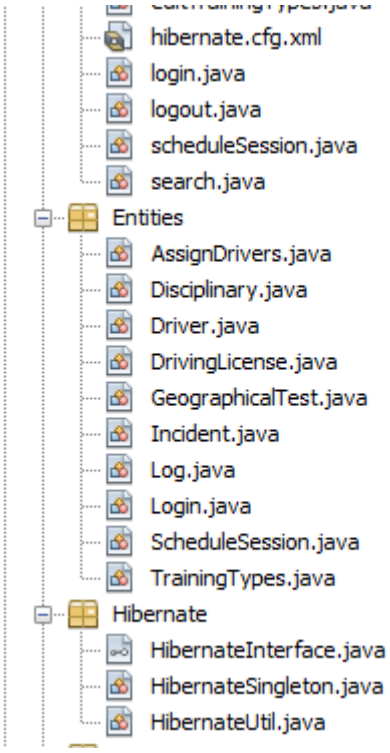


Figure 29 shows the Web Service Client being used in the Desktop Application.

Referring to Figure 29, this shows the Web Service Client has been used to communicate through SOAP to get through to all the information from the Speedy Taxi Website. This is auto-generated by NetBeans and the Model class is used to separate this from the user.



Referring to Figure 30, this demonstrates all the Hibernate database connection on Speedy Taxi Website. Hibernate.cfg.xml file is all the information regarding authentication of the database and mapping entities that have been used. Entities package shows how many are used for database. Lastly, HibernateSingleton is the class that uses all this information. This information inserts, updates and deletes from this class.

Figure 30 shows all the Hibernate information using entities

Evaluation

During this section, I will be thoroughly evaluating the evolution of the application. I will do a deep analysis on the implementation of the positive and negatives of the implementation. I will discuss any problems that have occurred during the system. Any alternative routes that could have been taken will be stated too. Lastly, I will be stating how I could have improved the final product for this application too.

Problems occurred

As Markus taught this course in Eclipse, my personal taste was to stick with NetBeans as this is the most comfortable software I can use. However, problems did occur when trying to send a JMS message. Referring to Figure 25, this shows the Connection Factories and Destination Resources that need to be created for the message to be sent in the queue. However, when clicked, "new" to create one; this gave me a runtime error stated that this is unable to create. However, I did find an alternative route to solve this issue. Command Prompt can use the GlassFish asadmin to create both of these. By changing the directory to "C:\Institutions\Gre\Apps\glassfish-4.1.1\glassfish\bin" and typing in asdamin enables the authority to do this. By typing in "create-jms-resource --restype javax.jms.QueueConnectionFactory jms/MyQueue1", enables a JMS Connection Factory to be created. Also, by typing in "create-jms-resource --restype javax.jms.Queue --property Name=PhysicalDestination jms/MyDest" creates the JMS Destination Resources to be created within both of these JMS Resources. As soon as this is successful for both, Figure 25 shows successfully that this has been created. One issue that has been faced is that this is only temporarily created. This perfectly works without error. However, I later found out that when I logout and close Glassfish Server, I realise that both of these have been deleted. As soon as this is closed, I have to follow these steps again for the JMS message to get sent. I do not realise what problem can occur. One issue that this may be is that the University is clearing this out. However, I do not currently know.

Another issue that has been faced on the Speedy Taxi Website is that when the Servlet, or Java Embedded in HTML, is changed and saved; an error is occurred stating, "*IllegalStateException: WELD-000227: Bean identifier index inconsistency detected - the distributed container probably does not work with identical applications*". This occurs when the application is running and refreshed. I had to close the application, deploy and run the application to prevent this from happening. This makes it a longer issue when making a small change just to close, deploy and run the application.

Positives

I will be stating the positives that both of the applications face. These are the following:

- Duplication Validation – Many drivers have different IDs for each driver. A duplication validation has been inserted to prevent this from happening. This is the same with the driving license too. Any editing can be done to prevent the same ID being submitted.
- SQL Injection is a code technique that can be used for data-driven applications. This is one of the techniques that have been inserted to prevent this from happening. Please refer yourself to Figure 31 as this shows the use of Hibernate. Hibernate uses Queries and sets the String separately to prevent this from happening. This is a good technique for security of the application. The second image of the first statement shows that this is SQL Injection.
- Singleton Pattern is used to restrict instantiation of the class to only once. This is an easy pattern that can used, and this would be good to use in a class that is called numerous times. I used this in the database class. Please refer yourself over to Figure 24 to see how Singleton Pattern has been used.
- Hibernate has been used for the database connection throughout this implementation. This is a good way to instantiate entities using Session Bean. Figure 24 and 30 shows this how Hibernate can be used
- JUnit Testing has been used only once in this implementation to show how JUnit Testing can used. Please refer yourself to Figure 31 as this shows only one test for createDriver. As you can see, this passed as 100% to be accurate. All three statements passed.
- Model and View is separate. Making sure that all of this is separate on all platforms is important.

```
@Test
public void driverTest() {
    int driverID = 3;
    String driverName = "test";
    String username = "test";
    String password = "test";
    HibernateInterface db = HibernateSingleton.getInstance();
    Driver c = new Driver(driverID, driverName, username, password);
    db.createDriver(c);

    assertTrue(driverName.equals(db.retrieveAllDriverInfo(driverID).getDriverName()));
    assertTrue(username.equals(db.retrieveAllDriverInfo(driverID).getUsername()));
    assertTrue(password.equals(db.retrieveAllDriverInfo(driverID).getPassword()));
```

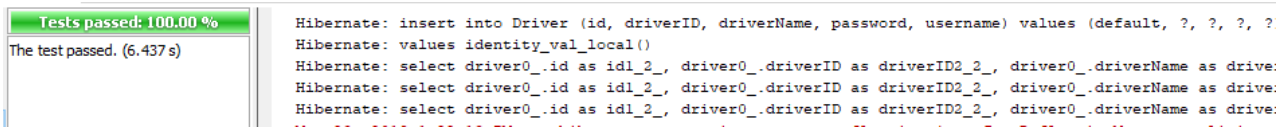


Figure 31 shows the JUnit Test for the application.

Negatives

- I noticed that throughout changes within SOAP, this must be deleted and refreshed. I find this very irritating as a small change occurs to deleting the whole SOAP and adding it onto the application again for this to be read. As soon as a small change is made for the client too, the application must "build" for the SOAP to integrate and refresh. A lot of refreshing is made.
- One negative that I would consider to be a negative is that the JMS Message has been sent through the Incident. However, I did not manage to get the retrieve working from the driver-client side of the implementation. I do consider this as a negative as the two implementations were missed and not been able to be included.

Suggested Alternatives

One suggested alternative that can be implemented is to make SOAP a different application for external use of the database to be connected and made use of. Currently, the SOAP is connected to the Speedy Taxi website as this works from the website. This could be another, or a better way to get the SOAP to work on different platforms too. Another alternative way that could have been done is to reduce the many JSP files that have been used. This has many pros and cons to it, but it could have been suggested to work on only file.

Improvements

One main improvement that I would do is make sure that the JMS Message Driven Bean in Advanced section is working with the driver application. I added the JMS Incident message to be sent to the JMS files. However, I did not manage to integrate this to be able to be received for the implementation. Another improvement that I could do is I failed to attempt to do the last Message Driven Bean for the 30 day expire to inform drivers. This was not attempted due to the failure of retrieving JMS Message Driven Bean for the Incident. A different way that Hibernate could have considered to be used is the use of foreign keys to be integrated with each entity. This can be used with started with the driver entity being linked with other qualification entities. This is a different way that can be integrated within Hibernate that has not been implemented. Another improvement that could be made is to make sure that the day log can be able to log out and still be able to log back in and log the information. Currently, the four logs must keep the application active. This was attempted and successfully working. However, I could not make the unexpected sequence of clicking the end log before the start. This is the only reason that I had to make sure that this is the improvement that could have been made at the end of this implementation. Another improvement that can be made in the Driver Desktop Application is the use of MVC to separate the Model and View together.

Another improvement that I would make sure is that assign drivers do not have the option of editing, nor deleting the selected assigned driver. This would be one improvement to make sure that this can be implemented in this application.

Java EE

Java Platform, Standard Edition (Java SE) supplies the core java programming functionality of the language. This defines everything from basic types to high-level classes that are used. Java SE uses various technologies that will be defined later in this report. Java Platform, Enterprise Edition is a platform that is built on top of Java SE platform (Oracle, 2012).
Java EE is a collection of hardware and independent platform on which enterprise applications can be run on. Java EE is an open, standard-based source making organisations to experiment on making many different applications (Prajapati and Dabhi, 2009). Many Enterprise Editions have been released since 1999. Since the release of Java EE 2, currently stands the most update edition of Java EE is currently on Edition 8. Java EE applications are composed of three layers that are assemblies of components. These are presentation layer, business layer, and database layer. Presentation layer includes all the JSP, JSF, Servlet and SOAP Engine. Business layer includes all the Session Bean, Entity Bean and Message-Driven Bean. Database layer includes all the connections relating to the database such as JDBC (Desertot, Donsez and Lalanda, 2006). All of these are a combination of technologies that are carried out in Java EE.
There are many different Java Platforms that have not been mentioned such as Java ME and Java FX that are going to be investigated further in this section of the report.

Many different languages have different frameworks for Java Editions. Java EE has many different Java Frameworks that going to be mentioned. As mentioned, Hibernate is one of the many Java EE frameworks. Hibernate is an Object-relational mapping framework is known for its high-quality option to deal with the database access. Hibernate has a clever way of incorporating communication to multiple databases (JavaPipe, 2019). JavaServer Faces (JSF) is a user interface framework for Java web applications. This is designed to simplify the construction of user interface using the MVC model that has been integrated in it (Burns and Kitain, 2006). JSF can be integrated within Java EE applications to separate this model and make this easier for users to implement. Google Web Toolkit (GWT) is an open source web development framework that allows high-end performance to be created by developers in Java (JavaPipe, 2019).

These that have been stated are Java EE frameworks that have been used. I am going to be naming some other frameworks in another language. Another framework used in C# is .NET framework. This is designed and supported to integrate different implementation alongside this. .NET uses Windows, ASP.NET and Base Class Library for different to be used (Altex, 2018). Meteor is a full stack framework that helps write application in Javascript. This is an MVC framework for Node.js (Kolisnyk, 2018). Entity Framework is an open source that is object-relational mapping that is used for ADO.NET. This has familiar to Hibernate in Java (Lombardo, 2016). Please note that the Table 1 demonstrates the comparison between the Java EE frameworks and other Frameworks that are available.

Java EE Framework	**Pros and Cons**	**Other Frameworks**	**Pros and Cons**
Hibernate	• Easy to configure and modify • Very fast • Powerful • Slow to start the application	.NET Framework	• Reliable • Cross-platform design • Flexible and easy to maintain
JavaServer Faces (JSF)	• Reusability of UI Components • Easily built custom UI components • Complex • Experienced is needed	Entity Framework	• Shortens code • Speed issues • Complicates command further
Google Web ToolKit	• Easy to use • Accessible to others • Slow	Meteor	• Very popular • Helps save time and energy • No SQL in database

Table 1 shows the pros and cons for Java EE frameworks and other frameworks

References

AltexSoft. (2018). *The Good and the Bad of .NET Framework Programming*. [online] Available at: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-net-framework-programming/ [Accessed 23 Mar. 2019].

Burns, E. and Kitain, R., 2006. JavaServer™ Faces Specification. [Accessed 23 Mar.2019]

Desertot, M., Donsez, D. and Lalanda, P., 2006, September. A dynamic service-oriented implementation for java ee servers. In *2006 IEEE International Conference on Services Computing (SCC'06)* (pp. 159-166). IEEE.

JavaPipe. (2019). *10 Best Java Web Frameworks to Use in 2019 (100% Future-Proof)*. [online] Available at: https://javapipe.com/blog/best-java-web-frameworks/ [Accessed 23 Mar. 2019].

Kolisnyk, M. (2018). *Meteor vs Angular: Their Pros and Cons*. [online] Diceus. Available at: https://diceus.com/meteor-vs-angular/ [Accessed 23 Mar. 2019].

Lombardo, C. (2016). *Pros And Cons of Entity Framework - Vision Launch*. [online] Vision Launch. Available at: http://visionlaunch.com/pros-and-cons-of-entity-framework/ [Accessed 23 Mar. 2019].

Oracle. (2012). *Document Information - Your First Cup: An Introduction to the Java EE Platform*. [online] Available at: https://docs.oracle.com/javaee/6/firstcup/doc/docinfo.html [Accessed 20 Mar. 2019].

Prajapati, H.B. and Dabhi, V.K., 2009, March. High quality web-application development on java EE platform. In *2009 IEEE International Advance Computing Conference* (pp. 1664-1669). IEEE. [Accessed on 19th March 2019].