MANCHESTER METROPOLITAN UNIVERSITY
School of Computing, Mathematics & Digital Technology

ASSIGNMENT COVER SHEET

Unit:	6G5Z1101 Advanced Programming
Assignment set by:	Dr Alan Crispin
Verified by:	Nick Whittaker
Moderated by:	Nick Whittaker
Assignment number:	1CWK50
Assignment title:	Vehicle Inventory System (Java)
Type: (GROUP/INDIVIDUAL)	Individual
Hand-in format and mechanism:	Submission is online, via Moodle. More information is available in the attached coursework specification.
Deadline:	As indicated on Moodle.

Unit Learning Outcomes being Assessed:

1) Design and implement object oriented applications and understand principles relating to interactive user interface development

2) Use tools and techniques and utilise existing classes and libraries in a systematic way to develop a complex software application

It is your responsibility to ensure that your work is complete and available for assessment by the date given on Moodle. If submitting via Moodle, you are advised to check your work after upload; and that all content is accessible. Do not alter after the deadline. You should make at least one full backup copy of your work.
Penalties for late hand-in: see Regulations for Undergraduate Programmes of Study:

http://www.mmu.ac.uk/academic/casqe/regulations/assessment-regulations.php

The timeliness of submissions is strictly monitored and enforced.

Plagiarism: Plagiarism is the unacknowledged representation of another person's work, or use of their ideas, as one's own. MMU takes care to detect plagiarism, employs plagiarism detection software, and imposes severe penalties, as outlined in the Student Handbook
http://www.mmu.ac.uk/academic/casqe/regulations/docs/policies_regulations.pdf
and Regulations for Undergraduate Programmes
http://www.mmu.ac.uk/academic/casqe/regulations/assessment.php
Exceptional Factors affecting your performance: see Regulations for Undergraduate Programmes of Study :

http://www.mmu.ac.uk/academic/casqe/regulations/assessment/docs/ug-regs.pdf

Assessment Criteria:	Indicated in the attached assignment specification.
Formative Feedback:	Formative feedback will be provided in laboratory sessions with an assignment check point
Summative Feedback	Written feedback in the form of a commented mark grid will be provided to each student and marks made available via Moodle
Weighting:	This Assignment is weighted at 50% of the total unit assessment.

Aim
This unit covers concepts relating to advanced object-oriented program design, the use of framework libraries, web server and mobile application development.

The aim of this assignment is to develop a web-based inventory system for a car sales business. The system will allow used car details (make, model, condition, price, etc.) to be added and removed from a back-end SQLite database. The web server front-end should allow users to browse available vehicles stored in the database providing search features based on different types of criteria such as make, model and price range.

The Brief
You are required to implement your own vehicle inventory system which allows a user to make CRUD (i.e. Create, Retrieve, Update and Delete) database operations on a back-end SQLite database storing fictitious vehicle data. The first stage of the assignment requires that you develop a menu based console application (see Fig. 1) to test CRUD operations on a back-end database called “vehicles.sqlite” which will be provided as part of the assignment brief. For attempting the advanced features of the assignment you will need to create your own database with additional fields. See the supporting Java CRUD screencast which shows how to create an SQLite database using the free graphical tool called DB Browser for SQLite.

```
-------------------------
Vehicle Inventory System
Choose from these options
-------------------------
1 - Retrieve all vehicles
2 - Search for vehicle
3 - Insert new vehicle into database
4 - Update existing vehicle details
5 - Delete vehicle from database
6 - Exit
Enter choice >
```

Fig. 1. Console vehicle inventory menu

Specifically your menu should provide options to retrieve all vehicles stored in the database, search for a specific vehicle based on a criterion such as vehicle ID or licence registration, insert a new vehicle into the database, update the details of an existing vehicle and delete a vehicle from the database.

The second stage of the assignment will require that you develop a web front end which will allow a user to browse and search for vehicles stored in the database. For the higher marks you will need to add additional functionality. Specifically, the web front-end should provide a login system to allow an administrator to log into the system to insert, edit (update) and delete vehicles stored in the database.

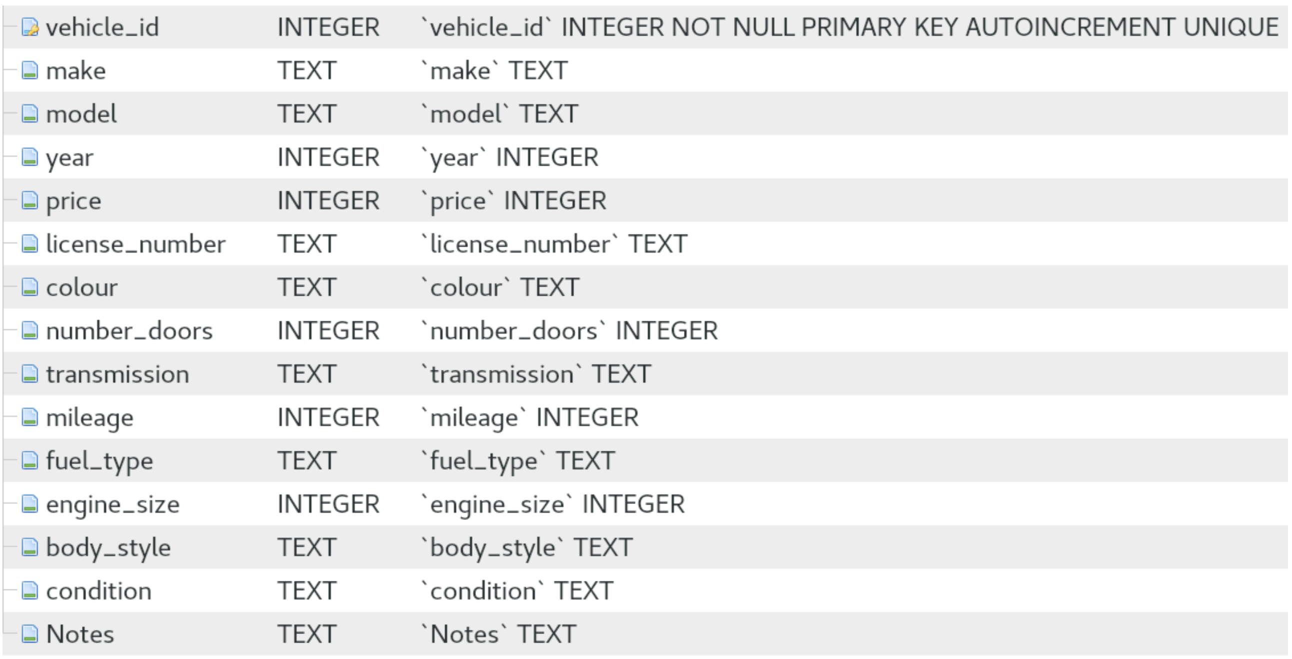
There are many other ways in which the system could be enhanced such as validating (using regular expressions) the licence plate string to check it is of the right format before inserting it into the database, modifying the web front-end to allow a user to search the inventory by make, model and price range, using password hashing as a security feature for administrator login and keeping a record of car sales. You will be rewarded for implementing additional features which sensibly enhance the system. In addition to the above your code should be fully documented using **Javadoc**.

The following provides a step by step guide to the assignment but you can tackle it in your own way if you so wish.

Step 1: Create the Vehicle class

The first step is to create a Vehicle class which maps the data stored in the vehicles.sqlite database. The database has a single table called vehicles which stores the vehicle details as shown in Fig. 2. The Vehicle.java class models the data entities in the database. Consequently it needs the following instance variables to be defined.

```
public class Vehicle {
   private int vehicle_id;
   private String make;
   private String model;
   private int year;
   private int price;
   private String license_number;
   private String colour;
   private  int number_doors;
   private String transmission;
   private int mileage;
   private String fuel_type;
   private int engine_size;
   private String body_style;
   private String condition;
   private String notes;
}
```



vehicle_id	INTEGER	\`vehicle_id\` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE
make	TEXT	\`make\` TEXT
model	TEXT	\`model\` TEXT
year	INTEGER	\`year\` INTEGER
price	INTEGER	\`price\` INTEGER
license_number	TEXT	\`license_number\` TEXT
colour	TEXT	\`colour\` TEXT
number_doors	INTEGER	\`number_doors\` INTEGER
transmission	TEXT	\`transmission\` TEXT
mileage	INTEGER	\`mileage\` INTEGER
fuel_type	TEXT	\`fuel_type\` TEXT
engine_size	INTEGER	\`engine_size\` INTEGER
body_style	TEXT	\`body_style\` TEXT
condition	TEXT	\`condition\` TEXT
Notes	TEXT	\`Notes\` TEXT

Fig. 2. Vehicles table

You need to create a constructor to allow these instance variables to be initialised which has the form shown below.

```java
public Vehicle(int vehicle_id, String make, String model, int
year, int price, String license_number, String colour, int
number_doors, String transmission, int mileage, String fuel_type,
int engine_size, String body_style, String condition, String
notes) {

//Initialise instance variables here..
}
```

You also need to create getter and setter methods for the class instance variables together with a toString() method.

Step 2: Create the VehicleDAO class

The next step is to develop the VehicleDAO class using the Java Database Connectivity (JDBC) API . This is the data access object (DAO) that provides an interface to the SQLite database and should contain the create, read, update and delete (CRUD) methods. The basic structure of the VehicleDAO class is shown in Fig. 3 and should have methods for each of the CRUD operations.

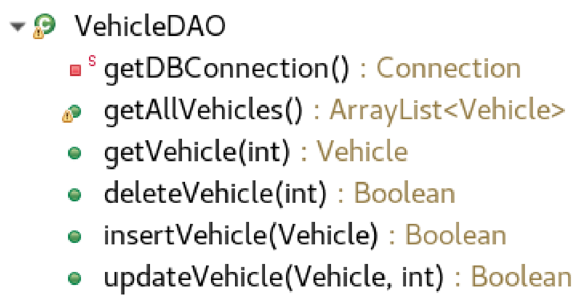

```
VehicleDAO
    getDBConnection() : Connection
    getAllVehicles() : ArrayList<Vehicle>
    getVehicle(int) : Vehicle
    deleteVehicle(int) : Boolean
    insertVehicle(Vehicle) : Boolean
    updateVehicle(Vehicle, int) : Boolean
```

Fig. 3. VehicleDAO class

The getAllVehicles() method uses the SQL SELECT statement to retrieve vehicle records from the database as shown below.

```
String query = "SELECT * FROM vehicles;";
```

For the getVehicle(int) method you need to specify a criterion for the data to be retrieved using the SQL WHERE clause. In the example below the vehicle_id number is used in the SQL SELECT statement.

```
String query = "SELECT * FROM vehicles WHERE vehicle_id =" +
vehicleID + ";";
```

Similarly for the deleteVehicle(int) method you will need to specify a criteria for the record to be deleted. In the example below the vehicle_id number is used.

```
String query = "DELETE FROM vehicles WHERE vehicle_id = "
+vehicle_id + ";";
```

The insertVehicle method will need to use the SQL statement INSERT INTO vehicles(...) VALUES (…) and the updateVehicle() method will need to use SQL UPDATE statement.

The w3Schools tutorial provides a refresher on SQL statements should you need it.
https://www.w3schools.com/sql/default.asp

You will need to add further methods to the VehicleDAO.java class which will be needed when implementing additional features. You will receive additional marks if your VehicleDAO uses SQL prepared statements (i.e. pre-compiled SQL statements with parameter placeholders). They have the following benefits. They help in preventing SQL injection attacks. They make it easier to set SQL parameters. Performance may be improved as statements are pre-compiled.

Step 3: Testing CRUD operation with console menu system

This stage of the assignment requires that you write a Controller class with a menu system to test each of the CRUD (i.e. Create, Retrieve, Update and Delete) database operations developed for the VehicleDAO class to check their functionality. You will need to use the Scanner class to receive input from the keyboard.

```
Scanner input = new Scanner(System.in);
```

Your menu system should look like that below so that when, for example, option 1 is selected all vehicles stored in the database are displayed as shown below.

```
-------------------------
Vehicle Inventory System
Choose from these options
-------------------------
1 - Retrieve all vehicles
2 - Search for vehicle
3 - Insert new vehicle into database
4 - Update existing vehicle details
5 - Delete vehicle from database
6 - Exit
Enter choice >
1
Retrieving all vehicles ....
Retrieving all vehicles ...
DBQuery = SELECT * FROM vehicles;
---------------------------
Vehicle ID = 1
Vehicle Make = Ford
Vehicle Model = Fiesta
Registration Year = 2014
Price = 5000
License Number = AB14CDE
Colour =Silver
Number of Doors = 5
Transmission = Manual
Mileage = 20000
Fuel Type = Petrol
Engine Size = 999
Body Style = Hatchback
Condition = Good
Notes = Eco Boost
---------------------------
Vehicle ID = 2
Vehicle Make = Toyota
Vehicle Model = Yaris
Registration Year = 2015
Price = 9000
License Number = AB15CDE
Colour =White
Number of Doors = 5
Transmission = Manual
Mileage = 14000
```

```
Fuel Type = Petrol
Engine Size = 1300
Body Style = Hatchback
Condition = Good
Notes = Icon
---------------------------
Vehicle ID = 3
Vehicle Make = Hyundai
Vehicle Model = ix35
Registration Year = 2013
Price = 8500
License Number = AB13CDE
Colour =White
Number of Doors = 5
Transmission = Manual
Mileage = 45000
Fuel Type = Diesel
Engine Size = 1600
Body Style = SUV
Condition = Good
Notes = 2WD
---------------------------
```

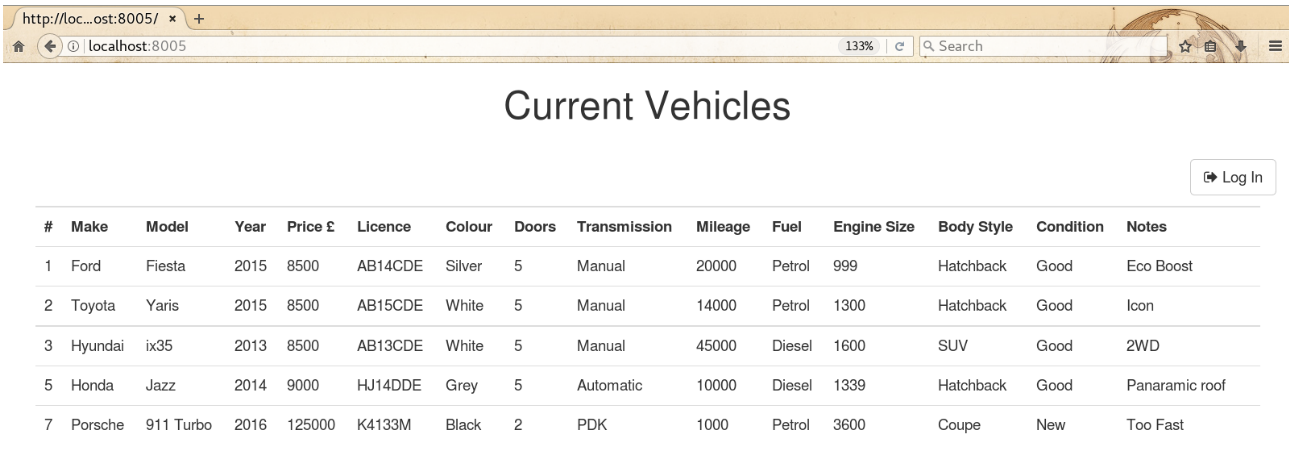
When selecting option 3 the user should be able to enter all details of the vehicle to be inserted into the database as shown below.

```
1 - Retrieve all vehicles
2 - Search for vehicle
3 - Insert new vehicle into database
4 - Update existing vehicle details
5 - Delete vehicle from database
6 - Exit
Enter choice >
3
Please enter new vehicle details..
Enter vehicle make: Honda
Enter vehicle model:   Jazz
Enter vehicle year  2015
Enter vehicle price:   9500
Enter vehicle license number:  HJ15ABC
Enter vehicle colour:  Silver
Enter number of doors: 5
Enter transmission type (manual/automatic):  Automatic
Enter mileage:10000
Enter fuel type (petrol.diesel,hybrid,electric):   Petrol
Enter engine size (cc):   1339
Enter body style(hatchback, estate, SUV, MVP coupe):  Hatchback
Enter condition(e.g. like new, good, fair):  Good
Enter notes (special features such as sat nav): Panaramic roof
```

Likewise when updating a vehicle the user should be provided with options to change any of the vehicle fields.

Step 4: Web Front-end (Displaying Vehicles)

This stage of the assignment requires that you write server code which displays vehicles stored in the database on a web page as shown in Fig. 4.



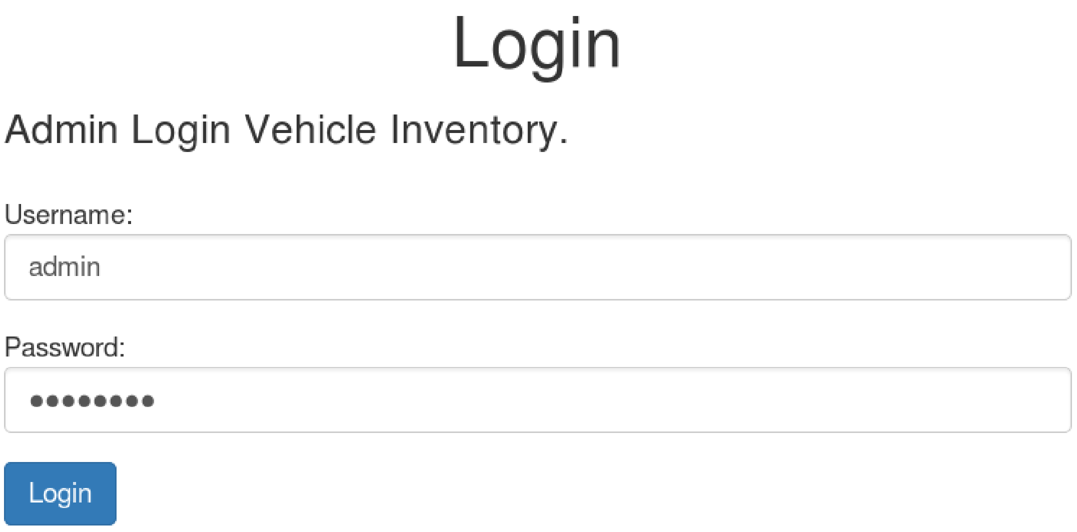
Current Vehicles

#	Make	Model	Year	Price £	Licence	Colour	Doors	Transmission	Mileage	Fuel	Engine Size	Body Style	Condition	Notes
1	Ford	Fiesta	2015	8500	AB14CDE	Silver	5	Manual	20000	Petrol	999	Hatchback	Good	Eco Boost
2	Toyota	Yaris	2015	8500	AB15CDE	White	5	Manual	14000	Petrol	1300	Hatchback	Good	Icon
3	Hyundai	ix35	2013	8500	AB13CDE	White	5	Manual	45000	Diesel	1600	SUV	Good	2WD
5	Honda	Jazz	2014	9000	HJ14DDE	Grey	5	Automatic	10000	Diesel	1339	Hatchback	Good	Panaramic roof
7	Porsche	911 Turbo	2016	125000	K4133M	Black	2	PDK	1000	Petrol	3600	Coupe	New	Too Fast

Fig. 4. Displaying vehicles stored in the database using an HTML table

Step 5: Administration Login/Logout

This stage of the assignment requires that you use the “users” table in the vehicles.sqlite database to provide a login system to allow an administrator to log into the system to insert, edit (update) and delete vehicles in the database.



Login

Admin Login Vehicle Inventory.

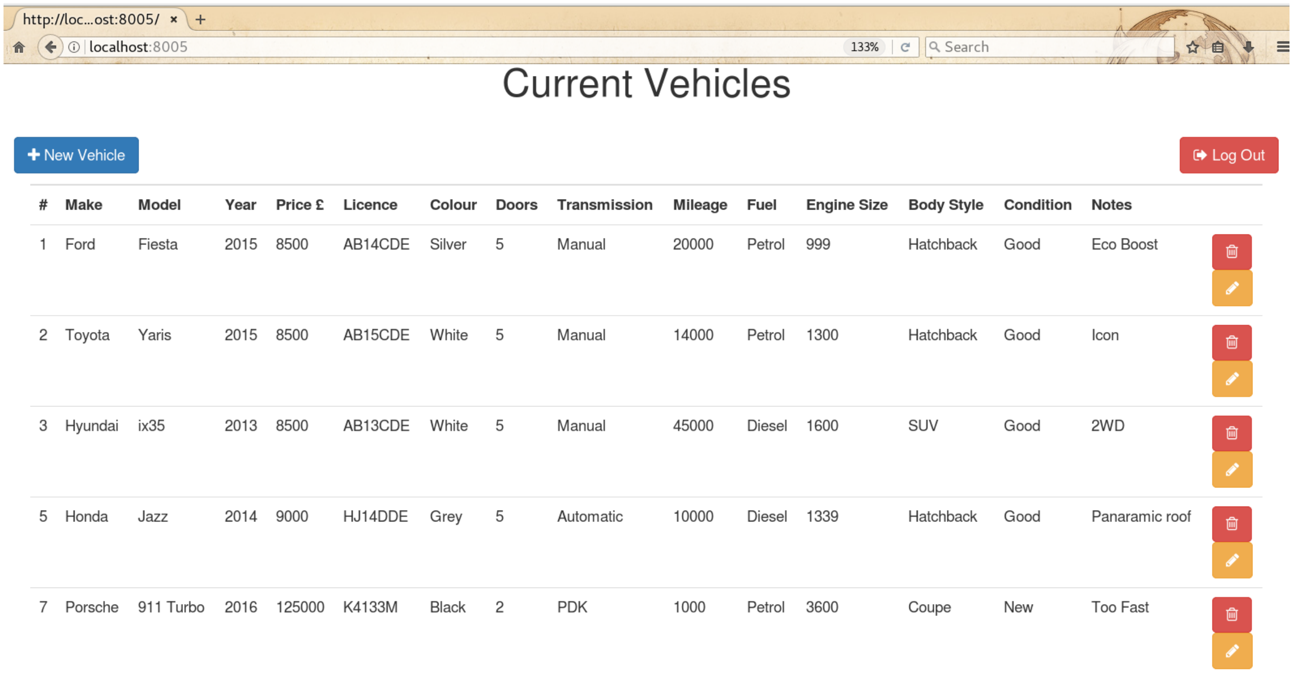
Username: admin

Password: ●●●●●●●●●

Fig. 5. Login form

Step 7: Web Interface (Administrator)

This stage of the assignment requires that you develop code to allow an administrator to insert (add) a new vehicle to the inventory, edit (update) existing information about a vehicle and delete a vehicle (e.g. when a vehicle is sold). See Fig 6.



Current Vehicles

#	Make	Model	Year	Price £	Licence	Colour	Doors	Transmission	Mileage	Fuel	Engine Size	Body Style	Condition	Notes
1	Ford	Fiesta	2015	8500	AB14CDE	Silver	5	Manual	20000	Petrol	999	Hatchback	Good	Eco Boost
2	Toyota	Yaris	2015	8500	AB15CDE	White	5	Manual	14000	Petrol	1300	Hatchback	Good	Icon
3	Hyundai	ix35	2013	8500	AB13CDE	White	5	Manual	45000	Diesel	1600	SUV	Good	2WD
5	Honda	Jazz	2014	9000	HJ14DDE	Grey	5	Automatic	10000	Diesel	1339	Hatchback	Good	Panaramic roof
7	Porsche	911 Turbo	2016	125000	K4133M	Black	2	PDK	1000	Petrol	3600	Coupe	New	Too Fast

Fig. 6. Administrator view to allow a new vehicle to be added to the database, details of an existing vehicle to be edited and a vehicle to be deleted

Step 8: Advanced Features

Finally modify your solution to add advanced features which should include:-

(i) a check that licence registrations are of the correct format (regular expressions)
(ii) server allows a user to search the inventory by make, model and price range
(ii) password hashing
(iii) a way of recording vehicle sales in the database

Your code should check that licence registrations are of the correct format. New registrations from 2001 should have the format XXNNXXX . Old registrations have a different format. You need to research the validation of car licence registration using regular expressions.

A password hash is an encrypted sequence of characters obtained after applying a hash algorithm (for the purposes of this assignment you can use the MD5 hashing algorithm as it is fast and easy to implement but be aware that it generates fairly weak hashes) on a text password. Text passwords which are very weak and easy to guess.

Once a password hash has been generated and stored in the database, you cannot convert it back to the original password. Each time an administrator logs into the vehicle system it is necessary to regenerate the password hash and match with the hash stored in the database. This means that if an administrator forgets their password, they have to be sent a temporary password and are asked to change it with this new password. You will need to research this area to complete this aspect of the assignment.

You will be rewarded for implementing additional features which sensibly enhance the system. In your assignment template form please specify any additional features that you have added to the inventory management system showing screenshots.

Hand In:

You are required to up-load your **project files** and a **completed assignment template form** (see below) indicating which features have been implemented successfully. You should submit a complete zipped copy of your source code (i.e. the bases classes, DAO class, menu tester, server code, SQLite database, jar files etc.) and your completed assignment template form with the format

surname_studentnumber.zip

Please ensure that all code submitted compiles and runs at the command line and your name and student number are included as comments in your source code files.

Assignment Template Form

Assignment Functionality Completed:

Vehicle class implemented
☐ Vehicle class fully implemented and tested

VehicleDAO CRUD methods implemented
☐ Method to connect to the vehicle database
☐ Method to return all vehicles stored in the database
☐ Method to return a single vehicle based on ID
☐ Method to insert vehicle into database
☐ Method to update details of a vehicle stored in the database based on ID
☐ Method to delete vehicle in database based on ID
☐ Use of prepared statements

Console menu system incorporated

☐ Application has a menu system to allow CRUD methods to be fully tested

Web Front-End

☐ Server connects to database and vehicles displayed using an HTML table
☐ Server incorporates administrator log-in
☐ Administrator can add new vehicle
☐ Administrator can update existing vehicle
☐ Administrator can delete vehicle

Advanced Features

☐ Check the validity of licence registrations (regular expressions)
☐ Server allows a user to search the inventory by make, model and price range
☐ Security features (password hashing, temporary password generation)
☐ Vehicle sales recorded

Additional Comments

Please indicate any features that have been partially implemented or any other issues that need to be drawn to the attention of the markers.

Screencasts: to support this assignment include those on Java syntax, classes and objects, inheritance and polymorphism, Java Database Connectivity (JDBC) and Java web programming.

Feedback: Marks will be made available via Moodle with a turnaround time in accordance with University guidelines. Written feedback will be provided to each student by tutors.

Vehicle Class		**Database Connectivity**		**Console Menu Testing**		**Web Front-End**		**Code Quality/Report**	
Attempt at getter, setter, constructor and toString methods	5 marks	Connection to database and CRUD method to return all vehicles	5 marks	Menu system allows all vehicles to be displayed	2 marks	Server connects to the SQLite database and vehicles displayed using an HTML table	5 marks	Assignment template form completed	2 marks
Vehicle class fully implemented and tested	10 marks	CRUD method to insert new vehicle	5 marks	Menu system allows a single vehicle to be selected based on vehicle ID	2 marks	The web front-end provides a login system to allow an administrator to log into the system to edit and delete vehicles stored in the database	15 marks	Code is **well** presented, with **good** Javadoc documentation and **sensible** comments	8 marks
		CRUD method to update vehicle	5 marks	Menu system allows a new vehicle to be inserted into database	2 marks				
		CRUD method to delete vehicle	5 marks	Menu system allows a vehicle to be selected so that details can be updated	2 marks	Additional functionality: • Search based on criteria such as make, year and price • Check on the validity of licence registrations • Password hashing • Recording sales	15 marks		
		CRUD method to return single vehicle based on ID	5 marks	Menu system allows a vehicle to be deleted from database	2 marks				
		Use of prepared statements	5 marks						
Section Total	15		30		10		35		10

Mark Scheme: Your work in this assignment will be graded in a number of areas, attracting a number of marks for each. Guidance on the assessment criteria for each area is included below. The marks given for each area will be combined to give an overall mark for 1CWK50, which is worth 50% of the final unit mark.

For example, a student earning 15 marks in the *Vehicle class section*, 20 marks in *Database Connectivity section*, 10 marks each in the *Console menu* and *Web front-end sections* and 5 marks for *Code Quality* would achieve an overall mark of 60% (15 + 20 + 10 + 10 + 5). Note that in each area you must work upwards in terms of features implemented to achieve the maximum mark for that area. If you need any help in understanding this assignment please talk to the tutor who takes you for your laboratory sessions or arrange to see either Dr Alan Crispin or Dr Mohammed Kaleem during their office hours.