ECS506U Software Engineering
Group Project 2017

Group 30
Basim Hassan Mehdi
ec15111
Vehicle Records

Test and Contribution Report Template

Version 1.0

3 March 2017

Prof. Martin Neil

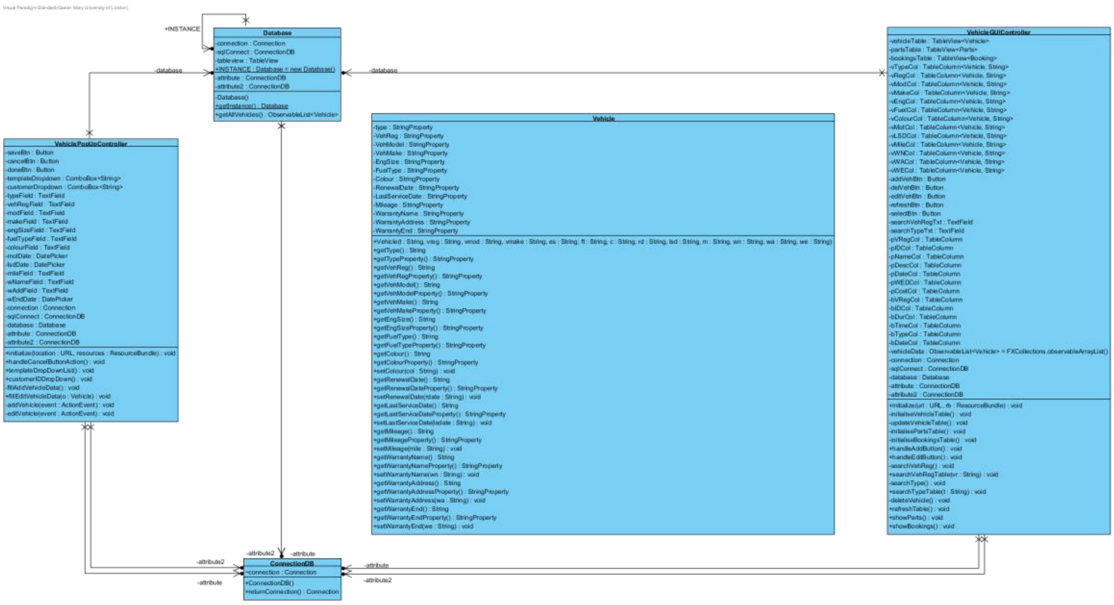
TABLE OF CONTENTS

1 Module Design

The layout of the Vehicles module has three tables taking up most of the screen and buttons and text fields on the right. The first table displays all the vehicles that are in the database, with this changing if the user searches type, vehicle registrations or adds or deletes an entry. The two tables below this table display the parts and bookings associated with a vehicle respectively. These tables show all parts and bookings in the system unless a vehicle is selected and ‘View Parts and Bookings’ is clicked.

The vehicle module has a main Vehicle.java class which declares variables and handles all the getter and setter methods. This works with the VehicleGUIController.java class which handles all the GUI functionalities like tables, buttons etc. I used the Database.java class for the SQL of the vehicles table like the observableList and selecting data for filling the table with. The vehicle table is initialised and displayed in the VehicleGUIController.java class. The parts and bookings table have their SQL in their methods in VehicleGUIController.java, as well as the display table code.

If ‘Add’ is clicked, a popup opens, this is another FXML file called addVehicle.fxml. There is also an editVehicle.fxml file that opens as a popup when the user clicks ‘Edit’. These two popups are handled in the VehiclePopUpController.java class. This class involves all the code to fill dropdown lists and to fill the text fields in these pop ups if a template or a vehicle is selected. It also includes UPDATE, DELETE and SELECT SQL queries for their respective functions.

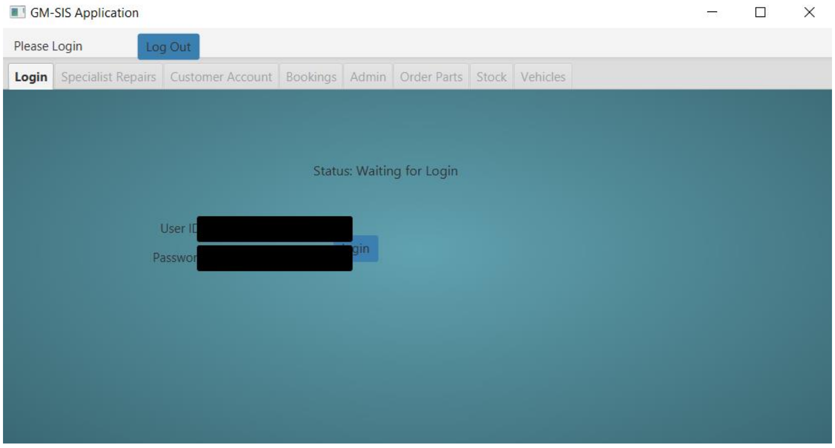

Visual Paradigm Standard(Queen Mary University of London)
+INSTANCE
-database
-database
Database
-connection : Connection
-sqlConnect : ConnectionDB
-tableview : TableView
+INSTANCE : Database = new Database()
-attribute : ConnectionDB
-attribute2 : ConnectionDB
-Database()
+getInstance() : Database
+getAllVehicles() : ObservableList<Vehicle>
VehiclePopUpController
-saveBtn : Button
-cancelBtn : Button
-doneBtn : Button
-templateDropdown : ComboBox<String>
-customerDropdown : ComboBox<String>
-typeField : TextField
-vehRegField : TextField
-modField : TextField
-makeField : TextField
-engSizeField : TextField
-fuelTypeField : TextField
-colourField : TextField
-motDate : DatePicker
-lsdDate : DatePicker
-mileField : TextField
-wNameField : TextField
-wAddField : TextField
-wEndDate : DatePicker
-connection : Connection
-sqlConnect : ConnectionDB
-database : Database
-attribute : ConnectionDB
-attribute2 : ConnectionDB
+initialize(location : URL, resources : ResourceBundle) : void
+handleCancelButtonAction() : void
+templateDropDownList() : void
+customerIDDropDown() : void
-fillAddVehicleData() : void
+fillEditVehicleData(o : Vehicle) : void
-addVehicle(event : ActionEvent) : void
-editVehicle(event : ActionEvent) : void
Vehicle
-type : StringProperty
-VehReg : StringProperty
-VehModel : StringProperty
-VehMake : StringProperty
-EngSize : StringProperty
-FuelType : StringProperty
-Colour : StringProperty
-RenewalDate : StringProperty
-LastServiceDate : StringProperty
-Mileage : StringProperty
-WarrantyName : StringProperty
-WarrantyAddress : StringProperty
-WarrantyEnd : StringProperty
+Vehicle(t : String, vreg : String, vmod : String, vmake : String, es : String, ft : String, c : String, rd : String, lsd : String, m : String, wn : String, wa : String, we : String)
+getType() : String
+getTypeProperty() : StringProperty
+getVehReg() : String
+getVehRegProperty() : StringProperty
+getVehModel() : String
+getVehModelProperty() : StringProperty
+getVehMake() : String
+getVehMakeProperty() : StringProperty
+getEngSize() : String
+getEngSizeProperty() : StringProperty
+getFuelType() : String
+getFuelTypeProperty() : StringProperty
+getColour() : String
+getColourProperty() : StringProperty
+setColour(col : String) : void
+getRenewalDate() : String
+getRenewalDateProperty() : StringProperty
+setRenewalDate(rdate : String) : void
+getLastServiceDate() : String
+getLastServiceDateProperty() : StringProperty
+setLastServiceDate(ladate : String) : void
+getMileage() : String
+getMileageProperty() : StringProperty
+setMileage(mile : String) : void
+getWarrantyName() : String
+getWarrantyNameProperty() : StringProperty
+setWarrantyName(wn : String) : void
+getWarrantyAddress() : String
+getWarrantyAddressProperty() : StringProperty
+setWarrantyAddress(wa : String) : void
+getWarrantyEnd() : String
+getWarrantyEndProperty() : StringProperty
+setWarrantyEnd(we : String) : void
VehicleGUIController
-vehicleTable : TableView<Vehicle>
-partsTable : TableView<Parts>
-bookingsTable : TableView<Booking>
-vTypeCol : TableColumn<Vehicle, String>
-vRegCol : TableColumn<Vehicle, String>
-vModCol : TableColumn<Vehicle, String>
-vMakeCol : TableColumn<Vehicle, String>
-vEngCol : TableColumn<Vehicle, String>
-vFuelCol : TableColumn<Vehicle, String>
-vColourCol : TableColumn<Vehicle, String>
-vMotCol : TableColumn<Vehicle, String>
-vLSDCol : TableColumn<Vehicle, String>
-vMileCol : TableColumn<Vehicle, String>
-vWNCol : TableColumn<Vehicle, String>
-vWACol : TableColumn<Vehicle, String>
-vWECol : TableColumn<Vehicle, String>
-addVehBtn : Button
-delVehBtn : Button
-editVehBtn : Button
-refreshBtn : Button
-selectBtn : Button
-searchVehRegTxt : TextField
-searchTypeTxt : TextField
-pVRegCol : TableColumn
-pIDCol : TableColumn
-pNameCol : TableColumn
-pDescCol : TableColumn
-pDateCol : TableColumn
-pWEDCol : TableColumn
-pCostCol : TableColumn
-bVRegCol : TableColumn
-bIDCol : TableColumn
-bDurCol : TableColumn
-bTimeCol : TableColumn
-bTypeCol : TableColumn
-bDateCol : TableColumn
-vehicleData : ObservableList<Vehicle> = FXCollections.observableArrayList()
-connection : Connection
-sqlConnect : ConnectionDB
-database : Database
-attribute : ConnectionDB
-attribute2 : ConnectionDB
+initialize(url : URL, rb : ResourceBundle) : void
-initialiseVehicleTable() : void
-updateVehicleTable() : void
-initialisePartsTable() : void
-initialiseBookingsTable() : void
+handleAddButton() : void
+handleEditButton() : void
-searchVehReg() : void
+searchVehRegTable(vr : String) : void
-searchType() : void
+searchTypeTable(t : String) : void
-deleteVehicle() : void
+refreshTable() : void
+showParts() : void
+showBookings() : void
-attribute2
-attribute
-attribute2
-attribute
-attribute
-attribute2
ConnectionDB
~connection : Connection
+ConnectionDB()
+returnConnection() : Connection

2 Team Performance

Group Member	Grade	Justification
Musab Adam	E	He was the group leader of sorts for our group, integrated our modules and made sure things were running smoothly. Would follow up on people if something was broken or an explanation was needed.
Qasim Hasan Ziauddin	M	
Loui Rajabi	M	
Oluwaromade Oluwarotimi Agonsoye	M	

3 Test Cases and Results

Login Screen

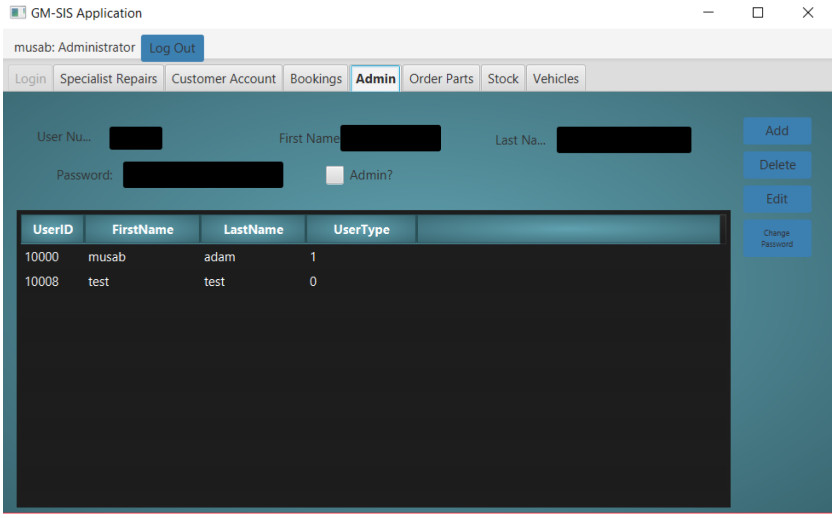


Status: Waiting for Login

User I [redacted]

Passwor [redacted]

Admin Tab as Administrator



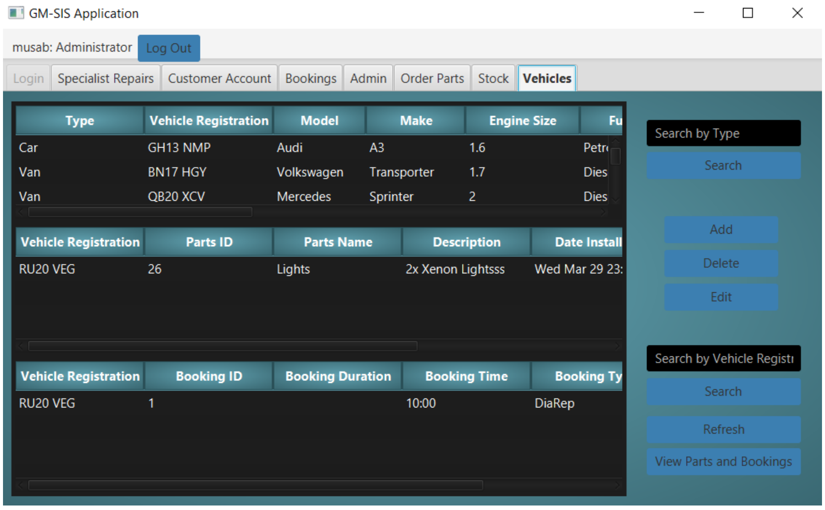
musab: Administrator

User Nu... [redacted] First Name [redacted] Last Na... [redacted]

Password: [redacted] □ Admin?

UserID	FirstName	LastName	UserType
10000	musab	adam	1
10008	test	test	0

Vehicles Tab



musab: Administrator

Type	Vehicle Registration	Model	Make	Engine Size	Fu
Car	GH13 NMP	Audi	A3	1.6	Petr
Van	BN17 HGY	Volkswagen	Transporter	1.7	Dies
Van	QB20 XCV	Mercedes	Sprinter	2	Dies

Vehicle Registration	Parts ID	Parts Name	Description	Date Install
RU20 VEG	26	Lights	2x Xenon Lightsss	Wed Mar 29 23:

Vehicle Registration	Booking ID	Booking Duration	Booking Time	Booking Ty
RU20 VEG	1		10:00	DiaRep

Module	Test Case	Pre-conditions	Actions	Result
Authentication	Logon as existing system administrator – provide id and password.	Program should be up and running.	Inputting the user ID and the password (‘10000’ and ‘newblarg’), and clicking enter or the ‘Login’ button.	Pass
	View list of existing day-to-day users.	Logged in as admin already.	Click on the ‘Admin’ tab to take you to the admin section, which displays a table with all the users.	Pass
	Change password of existing day-to-day user.	Logged in as admin already.	Select a day-to-day user from the table, enter the old password in the password text field and then click on the ‘Change Password’ button. A prompt will open asking to enter the new password.	Pass
	Create a new day-to-day user.	Logged in as admin already.	Enter details into text fields and then click ‘Add’. New user should show up in table.	Pass
	Logout existing administrator.	Logged in as admin already.	Click on logout button on the top left next to admin name. This takes you to login page again.	Pass
	Login as the new day-to-day user.	Program should be up and running.	Enter in the details for new day-to-day user and click enter or the ‘Login’ button. This takes you to the program, but ‘Admin’ tab is now blurred out.	Pass
	Logout as new day-today user.	Logged in as day-to-day user already.	Click on logout button on the top left next to day-to-day user name. This takes you to login page again.	Pass

	Attempt login as new day-to-day user with incorrect authentication.	Program should be up and running.	Enter in the wrong details for new day-to-day user and click enter or the ‘Login’ button. This takes you to the login page again, but with the fields empty for the user to try again.	Pass
	Logon as existing system administrator.	Program should be up and running.	Enter in the details for the existing admin and click enter or the ‘Login’ button. This takes you to the program, but ‘Admin’ tab is now available.	Pass
	Delete newly created day-to-day user.	Logged in as admin already.	Select the day-to-day user from the table and click on the delete button.	Pass
	Login as existing day-to-day user	Program should be up and running.	Enter in the details for the existing day-to-day user and click enter or the ‘Login’ button. This takes you to the program, but ‘Admin’ tab is blurred out.	Pass
Vehicles	Search for vehicles by type (car, truck, van).	Program should be up and running and user should be logged in.	Type a vehicle ‘Type’ in the type search field and click on the ‘Search’ button below it. This will update the vehicle table, displaying only the vehicles with the type entered.	Pass
	Select vehicle and show parts used, past and future booking dates, and the total cost per booking (warranty and non-warranty).	Program should be up and running and user should be logged in.	Select a vehicle from the vehicle table and click on the ‘View Parts and Bookings’ button on the bottom left. This will update the parts and bookings tables with the details associated to the selected vehicle.	Pass

	Search for vehicles by vehicle template.	Program should be up and running and user should be logged in.	Type a vehicle registration in the type search field and click on the 'Search' button below it. This will update the vehicle table, displaying only the vehicles with the vehicle registration entered.	Fail
	Select vehicle and show details.	Program should be up and running and user should be logged in.	Select a vehicle from the vehicle table and scroll to the right to see the vehicles details.	Pass
	Edit an existing vehicle record.	Program should be up and running and user should be logged in.	Select a vehicle from the vehicle table and click on the 'Edit' button. This will open a popup with the text fields already filled in from the vehicle selected.	Pass
	Edit the warranty details of a vehicle under warranty.	Program should be up and running and user should be logged in.	Select a vehicle from the vehicle table and click on the 'Edit' button. This will open the popup and warranty details can be edited on the bottom of the popup.	Pass
	Delete a vehicle record.	Program should be up and running and user should be logged in.	Select a vehicle from the vehicle table and click on the 'Delete' button. This will open a prompt which will ask the user to confirm.	Pass
	Create a new vehicle record.	Program should be up and running and user should be logged in.	Click on the 'Add' button. This will open a popup which the user can enter in the details and save to the database.	Pass
	Logout, close application	Program should be up	Logged out using button on the top left.	Pass

	and log back in. Show created record is persistently stored.	and running and user should be logged in.	Logged back in and went to the ‘Vehicles’ tab. Scrolled down in the vehicles table to find the new record.	