Hire_From_Us Car Hire System Technical documentation

Author: Aaron Edge (113612)

Date: 16/05/2018

Overview

This system is a custom software solution specifically designed for Hire_From_Us to aid the sales assistants with the handling and processing of administration documentation involved in hiring out a car to the customer.

The system is capable of storing data on all of the cars and previous customers, using that data to calculate the cost of the rental term and print of an invoice.

The system consists of a user facing front end built in java and a sql database running on a derby database server.

System Installation

Installing the client side software

Copy all of the files from the installation disk to a suitable folder.
To run the application double click on the file called Hire_From_Us.jar.
For convenience a shortcut to this file can be made on the desktop.

Setting up the database

First you would need to install derby on the server. Derby is an open source relational database management system based on java technologie and SQL. For more information visit the derby website at http://db.apache.org/derby.

The following is a summary of the guide to installing derby that can be found on the derby website mentioned (Db.apache.org, 2018).

To install derby:

1. Download the latest derby distribution, there are many different types of distribution available, make sure you download the “bin” distribution as that is the version that this guide will be using.
2. Extract the package to the server's hard drive.

The extracted installation contains several subdirectories:

“Demo”	-	contains demonstration programs.
“Bin”	-	contains the scripts for executing utilities and setting up the

environment.

“Javadoc”	-	contains the api documentation that was generated from the comments in the source code.
“Docs”	-	contains the derby help documentation
“Lib”	-	contains the derby .jar files
“Test”	-	contains regression tests for derby

To start the database:

1. open the terminal on the server.
2. Navigate to the derby installation folder using the ”cd” command:
   ```
   C:\ > cd
   ```
3. Then run this command to start the database:

   ```
    C:\Apache\db-derby-10.4.1.3-bin\lib> java -jar derbyrun.jar server start
   ```

Software Dependencies

-Client

  Jfoenix

  jdk

Server

  derbyClient

Structure of the database

Database tables

Users The users table is used to store the systems user details and can be interrogated to find out what permission level the user has access to.		
ID - PK	Int	This is the primary key and is used to identify the user.
USERNAME	String	The USERNAME field is used to store the users chosen username. The user will enter

		this string into the login system.
PERMISSION LEVEL	Int	The PERMISSIONLEVEL field is used to store an integer indicating the level of access the user has to specific parts of the system.
USERPASSWORD	String	The USERPASSWORD field stores the users password string. Compare the users input string to the string contained in this field to allow access to the system.

Customers The customers table is used to store all the information about the customers including payment details and licence details.		
ID - PK	Int	This is the primary key and is used to identify the customer.
FIRSTNAME	String	The FIRSTNAME field is used to store the customer’s first name.
LASTNAME	String	The LASTNAME field is used to store the customer's last name.
DATEOFBIRTH	Date	The DATEOFBIRTH field is used to store the customer’s date of birth as it appears on the driving licence.
PHONENUMBER	Int	The PHONENUMBER field is used to store the customer’s contact telephone number.
ADDRESS	String	The ADDRESS field is used to store the customer’s home address. The address should follow the format: Address line 1 TAB Address line 2 TAB Street TAB Town / City TAB County TAB Country TAB Postcode

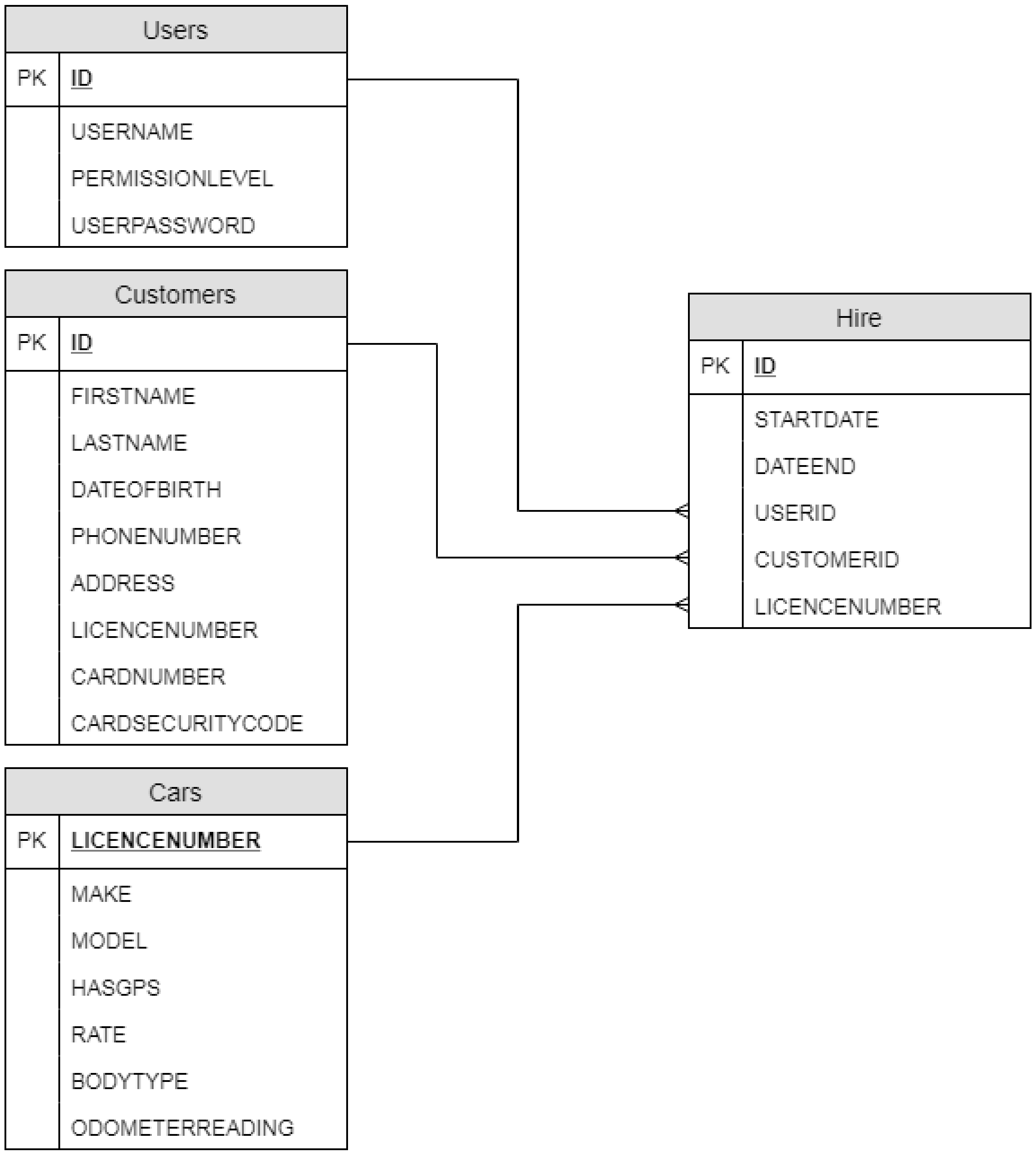
LICENCENUMBER	String	The LICENCENUMBER field is used to store the customer’s driving licence number as it appears on their driving licence.
CARDNUMBER	Int	The CARDNUMBER field is used to store the long number that appears on the customers chosen payment debt / credit card.
CARDSECURITYCODE	Int	The CARDSECURITYCODE field is used to store the three digit security code found on the back of a debt / credit card.

Cars The cars table is used to store all of the information about the hire company's fleet of cars.		
LICENCENUMBER - PK	Int	The LICENCENUMBER field is used to store the license plate number as it appears on the car. This is the primary key and is used to identify the car.
MAKE	String	The MAKE field is used to store the car manufacturers name.
MODEL	String	The MODEL field is used to store the cars model name.
HASGPS	Boolean	The HASGPS field is used to identify if the car is equipped with gps or not. True - the car is equipped with GPS False - The car is not equipped with GPS
RATE	Decimal	The RATE field is used to store the daily rate that the car can be rented out at in currency value. This rate is set by a user with management permission level access.
BODYTYPE	String	The BODYTYPE field is used to store the body shape of the car.
ODOMETERREADING	Int	The ODOMETERREADING field is used to store the current odometer reading of the

		car. It is updated when the car is returned from a hire.

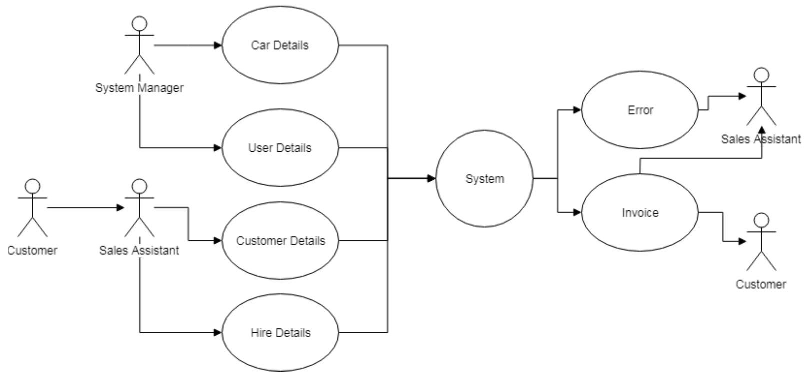
Hires The hires table is used to store all of the data related to the hiring process. It links the users data, customer data and car data together to form a hire record.		
ID - PK	Int	This is the primary key and is used to identify the hire.
STARTDATE	Date	The STARTDATE field is used to store that date that the hire started.
DATEEND	Date	The DATEEND field is used to store that date that the hire ended.
USERID	Int	The USERID field is used to store a reference to the id of the user that made the hire.
CUSTOMERID	Int	The CUSTOMERID field is used to store a reference to the id of the customer that is hiring the car.
LICENCENUMBER	String	The LICENCENUMBER field is used to store a reference to the licence plate number of the car that the customer is hiring.

Entity relationship diagram

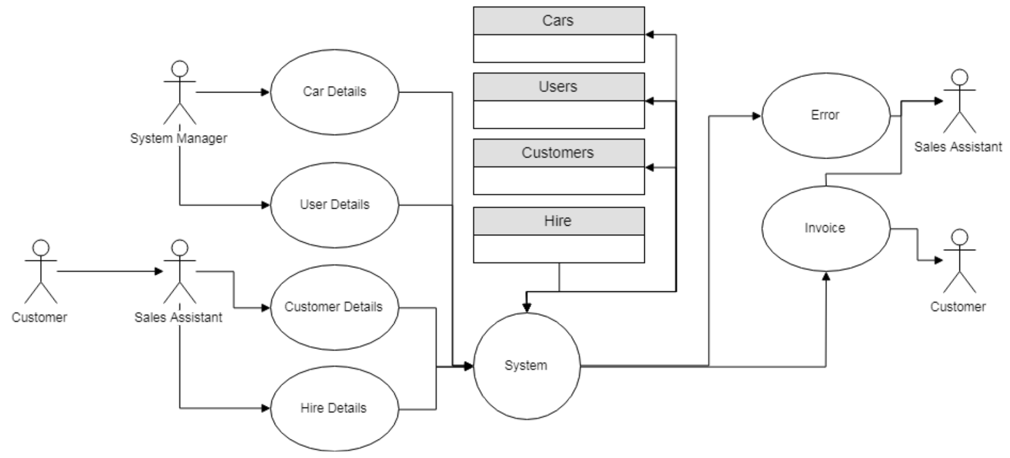


Flow of data

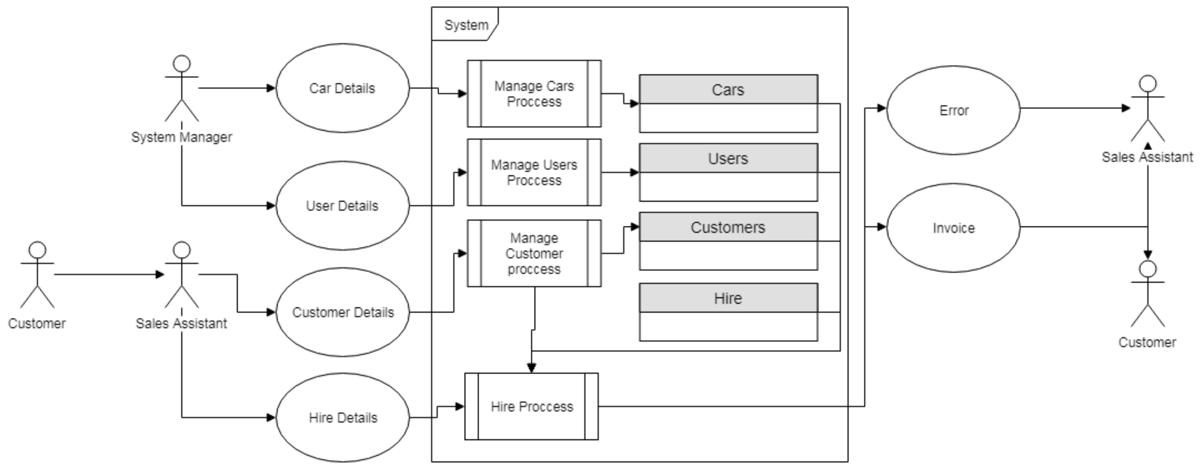
Context diagram



Data flow diagram level 1



Data flow diagram level 2



Data dictionary

Name	Type	Scope	Purpose
myStageManager	StageManager	public	This variable is used to store a reference to the stage manager object created when the application is first started
mySystemManager	SystemManager	public	This variable is used to store a reference to the system manager object created when the application is first started
instance	MainClass	private	This variable is used to store a reference to the main class object. This variable should not be used directly. Instead use the getInstance() method to reference the main object.

Class list

Class Name	System Manager
Overview	The system manager contains utility methods that can be used anywhere in the application using a reference to the mySystemManager object in the main class. The system manager acts as an interface to the main variables stored in the system. To access the utility methods contained within the system manager class use this code: `MainClass.getInstance().mySystemManager`
Properties	loginErrorString - Used to get the specific error response from a login request. LoggedInUserName - Use method: GetLoggedInUser() to get the name of the currently logged in user. LoggedInUserID - Use the method: GetLoggedInUserID() to get the ID of the currently logged in user.

	CurrentPermissionLevel - Use method: GetPermissionLevel() to get the currently logged in users permission level.

Methods		
Name	Scope	Purpose
doConnect()	Public	Used to make a connection to the database.
Logout()	Public	Used to logout the currently logged in user.
Login()	Public	Used to login a user with a username and password.
MakeNewHire()	Public	This method is used to input all of the objects stored in memory into the database.
ClearSelection()	Public	Clears all the selected objects in memory.
GetLoggedInUser()	Public	Get the user name of the user currently logged in.
GetLoggedInUserID()	Public	Get the user ID of the user currently logged in.
GetPermissionLevel()	Public	Get the permission level of the currently logged in user.
GetColNames()	Public	Get the column names of a given table.
GetAllMakes()	Public	Returns an array of string with all of the names of the makes of cars stored in the database.
GetAllModels()	Public	Returns an array of string with all of the names of the models of cars stored in the database.
convertStringToDate()	Public	Convert a date from string format to date format.
convertDateToStringForSQL()	Public	Converts a data variable to a format usable by the database.

getLocalDateFromDate()	Public	Helper function to convert date to a localdate.
getDateFromLocalDate()	Public	Helper function to convert localdate to a date.
getAllCustomersList()	Public	Get a list of all the customers in the customer table.
getAllCarsList()	Public	Get a list of all the cars in the cars table.
getAllUsersList()	Public	Get a list of all the users in the users table.
getAllHiresList()	Public	Get a list of all the hires in the hires table.
AddNewCustomer()	Public	Add a new customer to the customer table.
AddNewCar()	Public	Add a new car to the cars table.
AddNewUser()	Public	Add a new user to the users table.
AddNewHire()	Public	Add new hire to the hire table.
DeleteCustomer()	Public	Delete a customer record from the database.
DeleteCar()	Public	Delete a car record from the database.
DeleteUser()	Public	Delete a user record from the database.
DeleteHire()	Public	Delete a hire record from the database.
CalculateCostOfHire()	Public	Calculates the cost of hire for a given hire id.
CalculateCostOfHire()	Public	Calculates the cost of hire using data in memory
CalculateCommission()	Public	Calculates the commission for all of the users that have the Sales assistant permission level and returns a string array of of each users id and commission earned.
UpdateCustomer()	Public	Update the specified fields for a customer.
UpdateCar()	Public	Update the specified fields for a car.
UpdateUser()	Public	Update the specified fields for a user.

UpdateHire()	Public	Update the specified fields for a hire.
UpdateCarOdometer()	Public	Update the odometer reading of a specific car stored in the database.
GetMostHiredCarName()	Public	Returns the name of the car that has been hired out the most.
GetBestSalesPerson()	Public	Returns the sales assistant that has made the most hires.

Class Name	Stage Manager
Overview	The stage manager is used to manage what is being displayed by the GUI.
Properties	Stage - Used to store a reference to the stage object. Use the getter GetStage() to get access to this propertie.

Methods		
Name	Scope	Purpose
InitManager()	Public	Used once when the application is first run to setup the stage.
GoBack()	Public	Used to display the previous scene.
GoToWindow()	Public	This method is used to change the scene that is currently being displayed to the specified scene.
GoToLogin()	Public	This method is used to display the login scene.
replaceSceneContent()	Private	This is an internal method that handles the switching of scenes. Do not use this method directly to switch scenes, use GoToWindow() instead.
GetStage()	Public	Returns the currently loaded stage.

Class Name	Car
Overview	This class is used to store all of the data relating to a car in the system.
Properties	LicenceNumber - String used to sore the cars licenceplate number. Make - String used to store the make of the car Model - String used to store the model of the car HASGPS - Boolean used to identify if the car has a GPS installed or not. Rate - Double used to store the rental rate of the car. BodyType - String used to store the body type of the car. OdometerReading - Int used to store the current odometer reading of the car.

Methods		
Name	Scope	Purpose
getLicenceNumber()	Public	Returns the LicenceNumber.
getMake()	Public	Returns the Make.
getModel()	Public	Returns the Model.
getHASGPS()	Public	Returns true if the car is equipped with a GPS false if not.
getRate()	Public	Returns the rental rate.
getBodyType()	Public	Returns the body type.
getOdometerReading()	Public	Returns the current odometer reading.

Car()	Public	Constructor used to instantiate a new car object.

Class Name	Customer
Overview	This class is used to store all of the data relating to a customer in the system.
Properties	ID - Int to store the customers database ID. FirstName - String to store the customers first name. LastName - String to store the customers last name. DOB - String to store the customers date of birth. PhoneNumber - String to store the customers contact phone number. Address - String to store the customers current address. LicenceNumber - String to store the customers driving licence number.

Methods		
Name	Scope	Purpose
Customer()	Public	Constructor used to instantiate a new customer object.
Property Getters	Public	Each of the properties can be retrieved by using the prefix get<property name> ()
Property setters	Public	Each of the properties can be set by using the set<property name>() prefix.

Class Name	User
Overview	This class is used to store all of the data relating to a user in the system.
Properties	ID - Int to store the users database ID. UserName - String used to store the username PermissionLevel - Int to store the users permission level, 0 being the highest permission level. Password - Used to store the users password.

Methods		
Name	Scope	Purpose
Property Getters	Public	Each of the properties can be retrieved by using the prefix get<property name> ()
Property setters	Public	Each of the properties can be set by using the set<property name>() prefix.
User()	Public	Constructor used to instantiate a new user object.

Class Name	Hire
Overview	This class is used to store all of the data relating to a hire in the system.
Properties	ID - Int to store the hire database ID. StartDate - String used to store the the date that the car will be hired out from. EndDate - String used to store the date that the car will be returned.

	CustomerID - Int to store the database ID of the customer that is hiring the car. UserID - Int to store the database ID of the user that conducted the hire. LicenceNumber - String used to store the license plate number of the car that is being hired out.

Methods		
Name	Scope	Purpose
Property Getters	Public	Each of the properties can be retrieved by using the prefix get<property name> ()
Property setters	Public	Each of the properties can be set by using the set<property name>() prefix.
Hire()	Public	Constructor used to instantiate a new hire object.

Controller Classes

Controller Class Name	AddNewCarController
Description	This is the controller class for the AddNewCar window
Trigger functions	SaveCar() - Save the car to the database GoBack() - go to the previous window

Controller Class Name	AddNewCustomerController
Description	This is the controller class for the AddNewCustomer window
Trigger functions	SaveCustomer() - Save the customer to the database

	GoBack() - go to the previous window

Controller Class Name	AddNewUserController
Description	This is the controller class for the AddNewUser window
Trigger functions	SaveUser() - Save the user to the database GoBack() - go to the previous window

Controller Class Name	CarSelectionController
Description	This is the controller class for the CarSelection window
Trigger functions	GoNext() - Save the selected car to the system manager object GoBack() - go to the previous window

Controller Class Name	CustomerSelectionController
Description	This is the controller class for the CustomerSelection window
Trigger functions	GoNext() - Save the selected customer to the system manager object GoBack() - go to the previous window AddNewCustomer() - go to the AddNewCustomer window

Controller Class Name	DataAnalysisController
Description	This is the controller class for the DataAnalysis window
Trigger functions	GoBack() - go to the previous window

Controller Class Name	DateSelectionController

Description	This is the controller class for the DateSelection window
Trigger functions	GoBack() - go to the previous window GoNext() - Save the selected dates to the system manager object

Controller Class Name	EditHireController
Description	This is the controller class for the EditHire window
Trigger functions	Save() - Saves the edits to the hire Cancel() - returns to the previous window

Controller Class Name	ExistingCustomerController
Description	This is the controller class for the ExistingCustomer window
Trigger functions	DeleteCustomer() - Deletes the selected customers database record EditCustomer() - Saves the edits to the customer GoBack() - returns to the previous window

Controller Class Name	HomeController
Description	This is the controller class for the Home window
Trigger functions	MakeNewHire() - Tells the system manager to start the hire process and goes to the customer selection window. EditExistingHire() - Goes to the edit hire window EditExistingCustomer() - Goes to the edit customer window. ManageUsers() - Goes to the user management window. ManageCars() - Goes to the car management window.

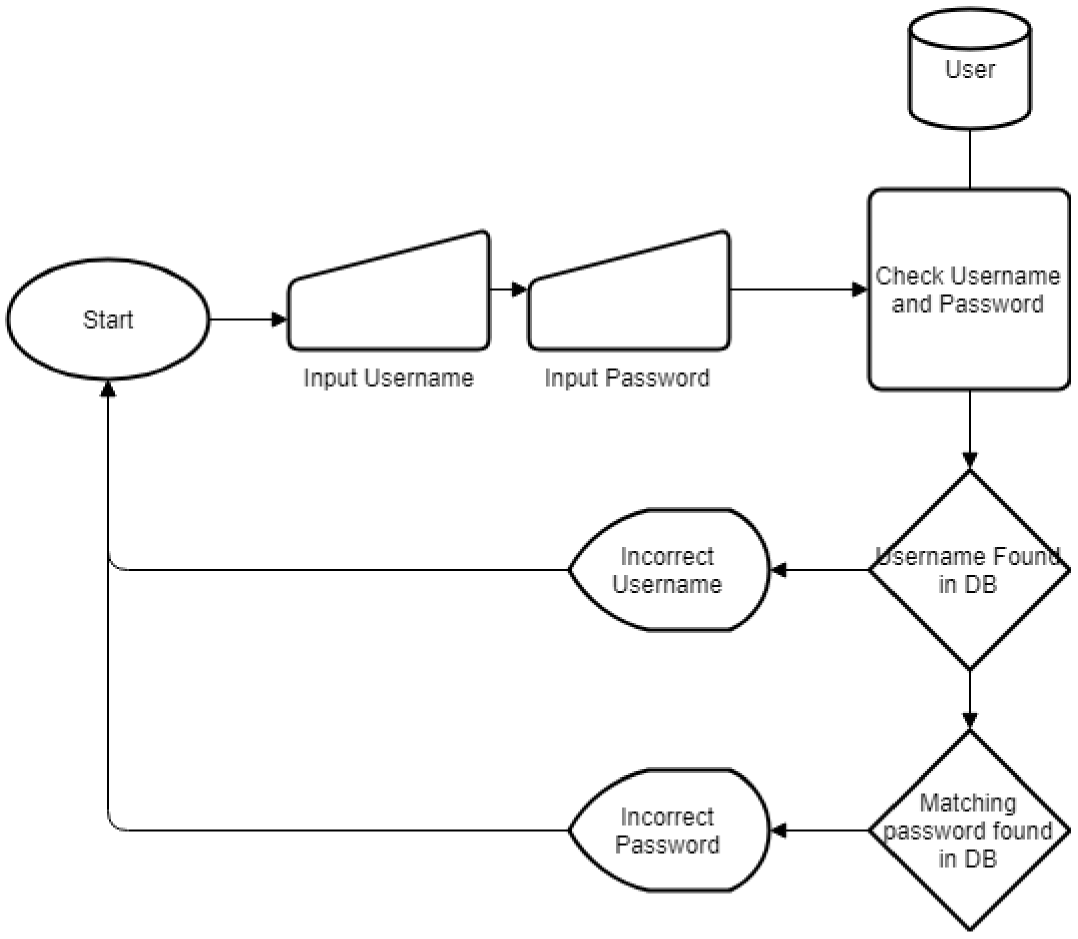
	ViewData() - Goes to the data analysis window. OpenSettings() - Shows the settings menu. CloseSettings() - Hides the settings menu. ReturnCar() - Tells the system manager to start the returns proccess and goes to the return car window.

Controller Class Name	LoginController
Description	This is the controller class for the login window
Trigger functions	UserLogin() - Attempts to login the user using the provided details. LogOut() - Tells the system manager to log the current user out.

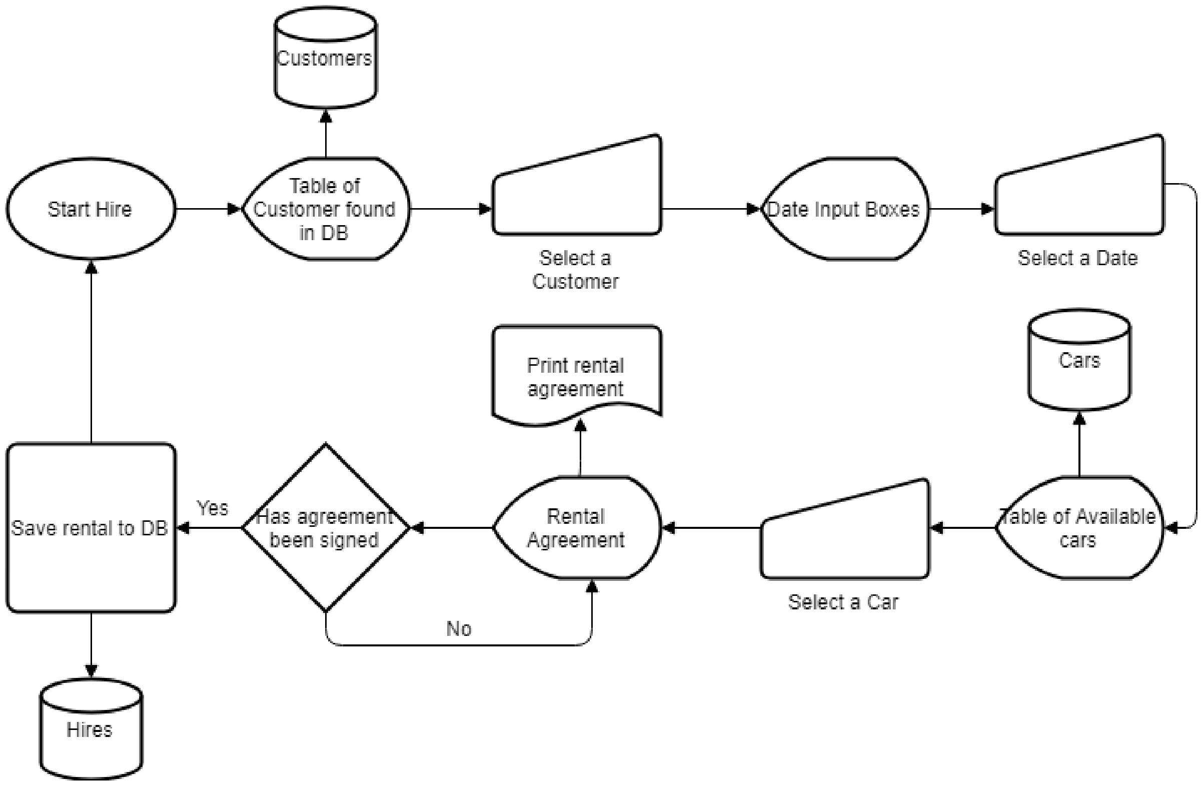
Processes

Flow charts

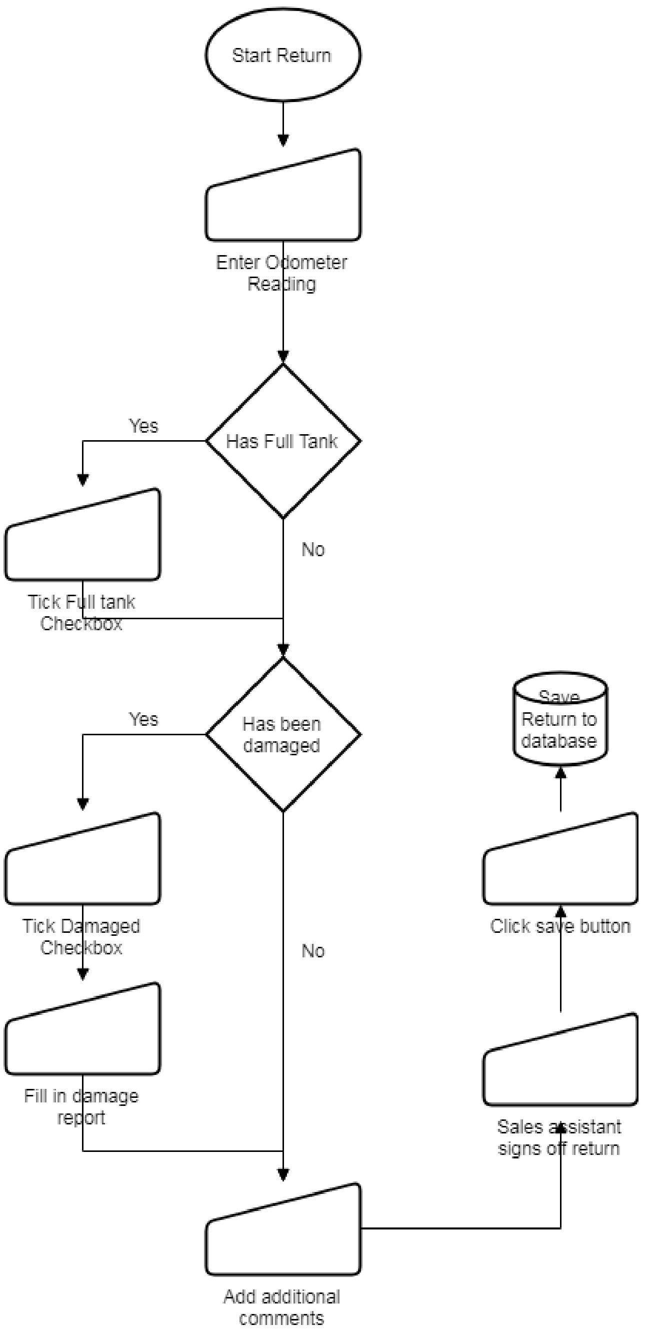
-Login



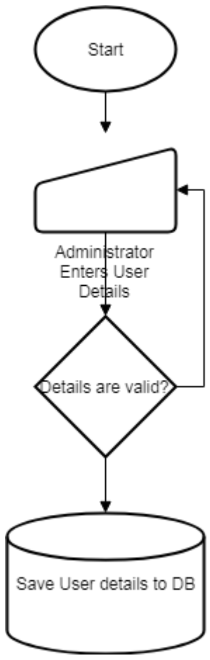
-Car Hire


Customers
Start Hire
Table of Customer found in DB
Select a Customer
Date Input Boxes
Select a Date
Cars
Print rental agreement
Save rental to DB
Yes
Has agreement been signed
Rental Agreement
Table of Available cars
Select a Car
No
Hires

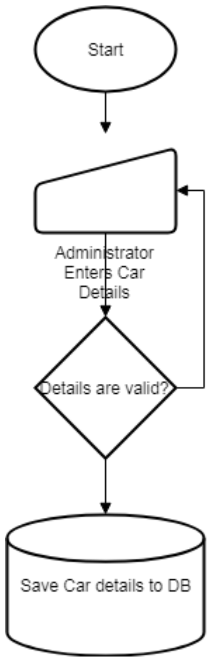
-Car Return


Start Return
Enter Odometer Reading
Has Full Tank
Yes
Tick Full tank Checkbox
No
Has been damaged
Yes
Tick Damaged Checkbox
Fill in damage report
No
Add additional comments
Sales assistant signs off return
Click save button
Save Return to database

-Add User


Start
Administrator Enters User Details
Details are valid?
Save User details to DB

-Add Car


Start
Administrator Enters Car Details
Details are valid?
Save Car details to DB

Design of GUI

Scene Builder
The elements of the UI can be built using the Scene Builder application with a jfoenix plugin for the material design styling.

Fxml
The fxml file type and language are used to store the structure of the GUI.

CSS
The css file type and language are used to store the styling of the GUI elements.

Main.css
The Main.css file contains all of the styles for the buttons.

List of button styles:
.ConfirmButton
.MenuButton
.CancelButton
.BackButton

Alert.css
The Alert.css file contains all of the styling for the alert popup.

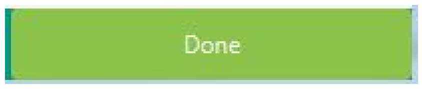
List of styles:
.root - for the window style
.content - for the message content style
.okButton - for the confirmation button style

Color scheme

Button Design

The man interface users will be using to interact with the system will be buttons. The button found within this application have been color coded to indicate their function.

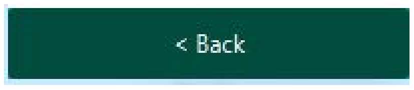
Bright Green button - Indicated a positive action, like adding data to the database running a process or continuing to the next part of the process.



Red button - Indicates a negative action, like canceling the current process or deleting data from the database.

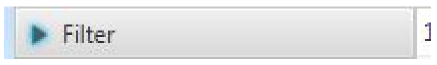
Blue button - Indicates an option, the label on the button indicates what process will be started when clicked.

Dark Green button - Indicates a negative navigation option like going back to the previous screen.



Options bar

Additional options are sometimes hidden underneath options bars (fig 7).



The arrow to the left of the bar indicates that there are extra options hidden, left click the bar to reveal the hidden options.

Reference

http://db.apache.org/derby/docs/10.14/getstart/getstartderby.pdf