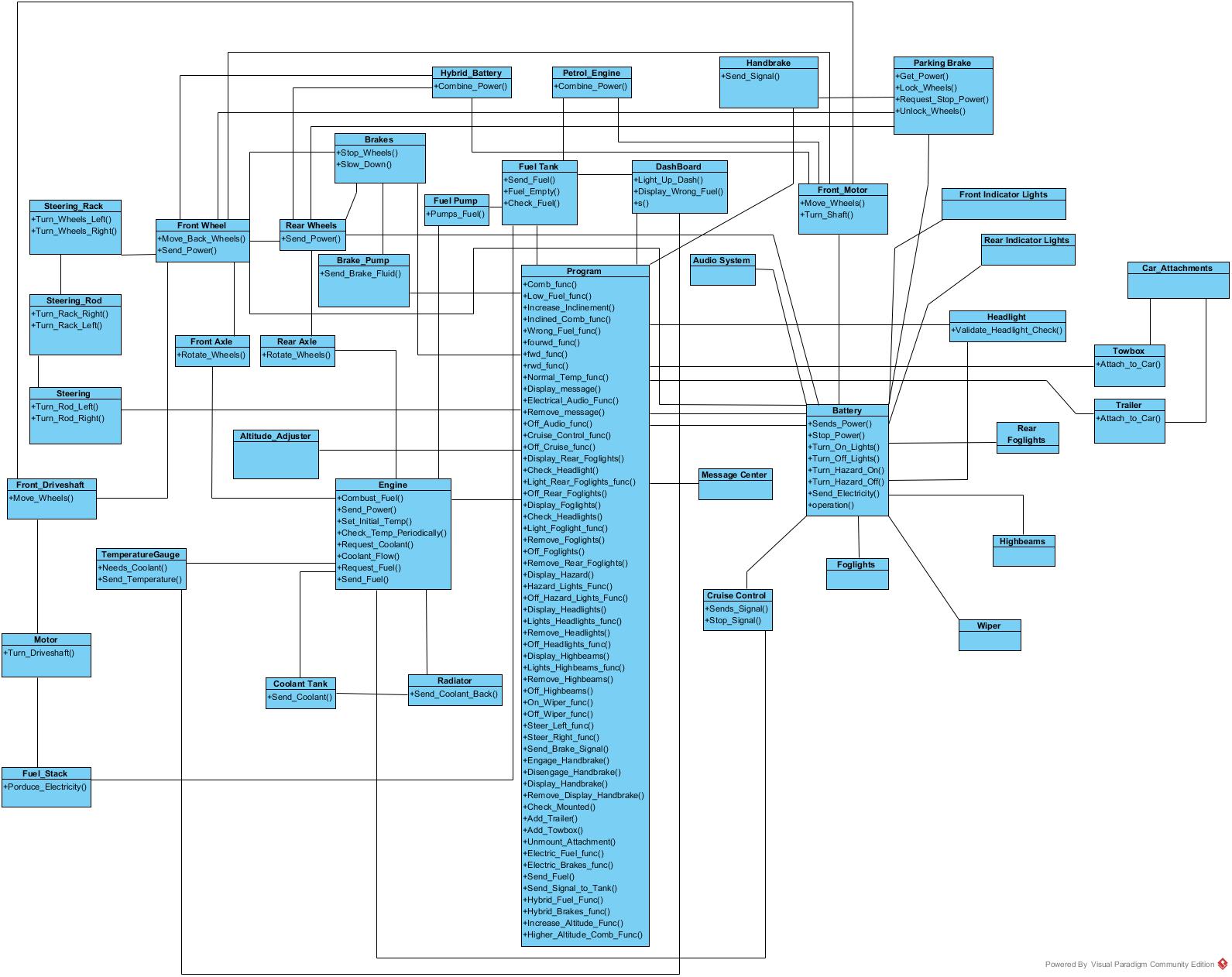
Class Diagram

The following page shows us the class diagram of our application. Note that this class diagram is not final, that is, some form of refactoring would take place throughout the development process.

In actual fact, over the last 6 weeks, this is actually a revised version of the class diagram. The initial class diagram is also included, to show the progress and the refactoring that was done to make the first revision of the class diagram.



new class diagram to go here

Use Cases

Over the last few weeks of discussion and brainstorming the functionalities of our application, we have come up with a number of use cases that should be present in the application we are developing.

Use case diagrams help as a quick checklist to ensure all functionalities have been implemented. In addition, the use case descriptions also assist in the development process as they act as a step-by-step guide on how each use case should run. This would help in the development process as it acts like pathway that a user needs to take to do something, hence it becomes like pathway on how a function should work.

The following pages show the use case diagram for our application, in addition to the use case descriptions. As this is a draft manual, there are some changes that can be expected to this section, as the development of the application is under way.

Macintosh HD:Users:kapilharesh:Dropbox:UOW:UOW February 2015:Project:CSCI321 Project:Weeeeeee:321 use case FOR MAC VIEWERS.pdf

# Use Case Descriptions

**Infographics of car**

|  |  |
| --- | --- |
| **Name:** Change perspective of car | **ID:** 1 |
| **Stakeholders and Goals:** User – to view different parts of the car | |
| **Description:** A user wants to view different parts of the car | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to view different parts of the car. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Infographics” button. Upon clicking on Infographics, the user will be brought to the infographics page where the user will be able to view the picture from the corner perspective (the default view). 2. To change to a different perspective (i.e to see a different part of a car), the user will then click on an arrow button which will be located above the description box, either to the left or to the right. If the user wishes to zoom in a perspective the up and down buttons can be used. The user will be able to view different components from different perspectives of the car. Different components will surround the picture of the different perspective of the car (View user manual for an example of interface). 3. Step 2 is repeated if the user wishes to see another perspective of the car. 4. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:** None | |

|  |  |
| --- | --- |
| **Name:** Obtain more information on part | **ID:** 2 |
| **Stakeholders and Goals:** User – to get more information on a component | |
| **Description:** A user wants to know more about a specific component of the car | |
| **Actors:** User | |
| **Trigger:** User runs the application and wants to know more about a specific part of the car | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Infographics” button. Upon clicking on Infographics, the user will be brought to the infographics page where the user will be able to view the corner perspective of the car (the default view). 2. The user will choose a perspective of the car he/she wishes to view. 3. Upon selecting a perspective that users want (except the corner perspective), to obtain more information on a particular component, the user clicks on the name of the component. The user can also click on the part on the picture. 4. The system will then display the description of the component on the description box. 5. Step 2 to 4 are repeated if the user wishes to see another component of the car. 6. Upon completion of this use case (Use Case ID: 3), the next use case **can/will** be triggered (\*\*The reason behind “can/will” is due to the ability of the user to choose if he/she wants to run the use case or not). | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:** None | |

|  |  |
| --- | --- |
| **Name:** View videos | **ID:** 3 |
| **Stakeholders and Goals:** User – to view videos on a component | |
| **Description:** A user wants to gain more information on a component by viewing a video prepared by the developers. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to view a video on a specific component of the car | |
| **Normal Flow:**   1. This use case is continued from the previous use case. Under the description box, the user will be able to click on a video link at the end. 2. The system will then pop up an overlay with the video but will not start playing the video 3. The user clicks “Play”. 4. The system starts playing the video. 5. Upon finishing the video or in the middle of watching the video, the user will be able to click the outer surroundings of the video overlay to go back to the Infographics screen. 6. If the user wishes to view a video for a different component, the user will have to run the previous use case again (Use Case ID: 2). 7. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (IF overlay cannot be done on the software itself)  2. The system will then prompt the user if he/she wishes to launch his/her default media player.  3. The user enters his/her input  4. The video will be played in an external player and upon completion the media player will be closed and the user will be brought back to the Infographics page of the system. | |

**Simulation of car**

1. **Persistent dashboard**

|  |  |
| --- | --- |
| **Name:** Start/Stop engine | **ID:** 4 |
| **Stakeholders and Goals:** User – to start/stop the engine | |
| **Description:** A user wants to start or stop an engine | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to start or stop the engine | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 3. The system will show that the engine has started, by showing different components being lit up on the dashboard. 4. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (The user wishes to turn off the engine instead)   1. On the persistent dashboard the user will be able to see the “Stop Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to stop the engine. 2. The system will show that the engine has stopped, by showing different components being dim down on the dashboard.   (The user wishes to turn off the engine while the car is still moving)   1. On the persistent dashboard the user will be able to see the “Stop Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to stop the engine. 2. The system will provide a warning to the user that the car is still moving and the engine cannot be stopped until the car has been turned off | |

|  |  |
| --- | --- |
| **Name:** Increase/Decrease incline | **ID:** 5 |
| **Stakeholders and Goals:** User – to increase/decrease the incline | |
| **Description:** A user wants to increase or decrease the degree of incline of the car. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to increase/decrease the incline | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 3. The system will show that the engine has started, by showing different components being lit up on the dashboard. 4. To increase the degree of incline, the user simply clicks on the “+” symbol under the “Incline” section. 5. The system will then display an increase in the degree of incline. 6. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (The user is trying to decrease the degree of incline)   1. To decrease the degree of incline, the user simply clicks on the “-” symbol under the “Incline” section. 2. The system will then display a decrease in the degree of incline. | |
| **Name:** Increase/Decrease speed | **ID:** 6 |
| **Stakeholders and Goals:** User – to increase/decrease the speed | |
| **Description:** A user wants to increase or decrease the speed of the car. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to increase/decrease the speed of the car. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 3. The system will show that the engine has started, by showing different components being lit up on the dashboard. 4. To increase the speed of the car, the user simply clicks the “+” symbol under the “Speed” section. 5. The system will then display an increase in speed of the car on the dashboard. 6. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (The user is trying to decrease the degree of incline)   1. To decrease the speed of the car, the user simply clicks on the “-” symbol under the “Speed” section. 2. The system will then display a decrease in the speed of the car on the dashboard. | |

|  |  |
| --- | --- |
| **Name:** Increase/Decrease altitude | **ID:** 7 |
| **Stakeholders and Goals:** User – to increase/decrease the altitude | |
| **Description:** A user wants to increase or decrease the altitude that the car is currently at. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to increase/decrease the altitude that the car is currently at. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 3. The system will show that the engine has started, by showing different components being lit up on the dashboard. 4. To increase the altitude that the car is currently at, the user simply clicks the “+” symbol under the “Altitude” section. 5. The system will then display an increase in altitude that the car is currently at. 6. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (The user is trying to decrease the degree of incline)   1. To decrease the altitude that the car is currently at, the user simply clicks on the “-” symbol under the “Altitude” section. 2. The system will then display a decrease in altitude that the car is currently at. | |

|  |  |
| --- | --- |
| **Name:** View car’s temperature, fuel and speed | **ID:** 8 |
| **Stakeholders and Goals:** User – to view the car temperature, fuel and speed | |
| **Description:** A user wants to view the temperature, fuel level and speed of the car. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to view the current car temperature, fuel level and speed. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 3. The system will show that the engine has started, by showing different components being lit up on the dashboard. 4. The user will also be able to see on the car dashboard, from the left and going in order, the first values represent the fuel level of the car, the second value represent the speed of the car and the final values represent the temperature of the car. 5. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:** None | |

1. **Fuel system simulation**

|  |  |
| --- | --- |
| **Name:** Simulate normal consumption | **ID:** 9 |
| **Stakeholders and Goals:** User – to simulate normal consumption of fuel of the car | |
| **Description:** A user wants to view the consumption of fuel of the car under normal circumstances. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to view the normal consumption of fuel of the car. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Fuel system”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view different components on the screen, which includes the fuel tank, fuel pump and the engine. There will also be a few buttons available for the user which includes “Simulate wrong fuel”, “Simulate filling up fuel” and “Simulate low fuel”. 3. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 4. The system will show that the engine has started, by showing different components being lit up on the dashboard. 5. The user will then be able to see the fuel flowing from the fuel tank to the fuel pump and to the engine. 6. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:** None | |

|  |  |
| --- | --- |
| **Name:** Simulate low fuel | **ID:** 10 |
| **Stakeholders and Goals:** User – to simulate low fuel | |
| **Description:** A user wants to view how the car responds when the fuel level is low. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to view the response of the car when the level of fuel is low. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Fuel system”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view different components on the screen, which includes the fuel tank, fuel pump and the engine. There will also be a few buttons available for the user which includes “Simulate wrong fuel”, “Simulate filling up fuel” and “Simulate low fuel”. 3. The user will then click on “Simulate low level” button in the simulation window. 4. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 5. The system will show that the engine has started, by showing different components being lit up on the dashboard. 6. The user will then be able to see the fuel flowing from the fuel tank to the fuel pump and to the engine. 7. The system will then light up the fuel symbol on the persistent dashboard. The fuel symbol lighting up on the dashboard represents the fuel level is low. 8. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:** None | |

|  |  |
| --- | --- |
| **Name:** Simulate wrong fuel | **ID:** 11 |
| **Stakeholders and Goals:** User – to simulate wrong fuel | |
| **Description:** A user wants to view the effects of adding the wrong fuel into the car. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to simulate wrong fuel being added to the car. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Fuel system”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view different components on the screen, which includes the fuel tank, fuel pump and the engine. There will also be a few buttons available for the user which includes “Simulate wrong fuel”, “Simulate filling up fuel” and “Simulate low fuel”. 3. The user then clicks on “Simulate wrong fuel” on the simulation window. 4. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 5. The system will show that the engine has started, by showing different components being lit up on the dashboard. 6. The user will then be able to see the fuel flowing from the fuel tank to the fuel pump and to the engine. 7. After 3 seconds, the system will then shut off the engine, by turning off the lights on the persistent dashboard. This is to show that the engine has broken down from adding the wrong fuel into the car. 8. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:** None | |

|  |  |
| --- | --- |
| **Name:** Simulate fuel consumption in different altitude | **ID:** 12 |
| **Stakeholders and Goals:** User – to simulate fuel consumption in different altitude | |
| **Description:** A user wants to view the difference in the rate of fuel consumption in different altitudes. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to simulate fuel consumption in different altitude. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Fuel system”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view different components on the screen, which includes the fuel tank, fuel pump and the engine. There will also be a few buttons available for the user which includes “Simulate wrong fuel”, “Simulate filling up fuel” and “Simulate low fuel”. 3. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 4. The system will show that the engine has started, by showing different components being lit up on the dashboard. 5. The user will then be able to see the fuel flowing from the fuel tank to the fuel pump and to the engine. 6. The user will then increase the current altitude by clicking on the “+” symbol under the “Altitude” section (This is closely related to the increase/decrease altitude use case, Use case ID: 7). 7. The system will then show the user the increased rate of consumption of fuel by showing more arrows flowing from the fuel tank, to the fuel pump and to the engine. 8. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (The user decides to decrease the current altitude instead)   1. The user will then decrease the current altitude by clicking on the “-” symbol under the “Altitude” section (This is closely related to the increase/decrease altitude use case, Use case ID: 7). 2. The system will then show the user the decreased rate of consumption of fuel by showing less arrows flowing from the fuel tank, to the fuel pump and to the engine. | |

|  |  |
| --- | --- |
| **Name:** Simulate fuel consumption in different incline | **ID:** 13 |
| **Stakeholders and Goals:** User – to simulate fuel consumption in different degree of incline | |
| **Description:** A user wants to view the difference in the rate of fuel consumption in different degrees of incline. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to simulate fuel consumption in different degrees of incline | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Fuel system”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view different components on the screen, which includes the fuel tank, fuel pump and the engine. There will also be a few buttons available for the user which includes “Simulate wrong fuel”, “Simulate filling up fuel” and “Simulate low fuel”. 3. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 4. The system will show that the engine has started, by showing different components being lit up on the dashboard. 5. The user will then be able to see the fuel flowing from the fuel tank to the fuel pump and to the engine. 6. The user will then increase the degree of incline by clicking on the “+” symbol under the “Incline” section (This is closely related to the increase/decrease incline use case, Use case ID: 5). 7. The system will then show the user the increased rate of consumption of fuel by showing more arrows flowing from the fuel tank, to the fuel pump and to the engine. 8. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (The user decides to decrease the current degree of incline instead)   1. The user will then decrease the degree of incline by clicking on the “-” symbol under the “Incline” section (This is closely related to the increase/decrease altitude use case, Use case ID: 5). 2. The system will then show the user the decreased rate of consumption of fuel by showing less arrows flowing from the fuel tank, to the fuel pump and to the engine. | |

|  |  |
| --- | --- |
| **Name:** Simulate filling up fuel | **ID:** 14 |
| **Stakeholders and Goals:** User – to simulate filling up fuel | |
| **Description:** A user wants to view the process of filling up fuel for the car | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to simulate filling up fuel for the car. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Fuel system”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view different components on the screen, which includes the fuel tank, fuel pump and the engine. There will also be a few buttons available for the user which includes “Simulate wrong fuel”, “Simulate filling up fuel” and “Simulate low fuel”. 3. The user then clicks on “Simulate filling up fuel” on the simulation window. 4. The system will then show arrows pointing upwards on the fuel tank to simulate the addition of fuel into the car (i.e the level of petrol in the car is increasing). 5. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User tries to fill up fuel when the car is NOT turned off, car is turned off when the engine is not started)   1. The system will then display to the user that the car should first be turned off before filling up fuel for the car. 2. On the persistent dashboard the user will be able to see the “Stop Engine” button. The user will then click on it to stop the engine. 3. The user then clicks on “Simulate filling up fuel” on the simulation window. 4. The system will then show arrows pointing upwards on the fuel tank to simulate the addition of fuel into the car (i.e the level of petrol in the car is increasing). | |

1. **Wheel drive system simulation**

|  |  |
| --- | --- |
| **Name:** View difference among different wheel drive system | **ID:** 15 |
| **Stakeholders and Goals:** User – to view the difference among different wheel drive system | |
| **Description:** A user wants to understand the difference between four wheel drive, rear wheel drive and front wheel drive | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to view difference among different wheel drive system | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. The user clicks on “Wheel drive system”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then able to see different components on the screen which consists of the Engine, Axle and Wheels. There will also be 3 buttons available for the user to click which is “4 wheel drive”, “Front wheel drive (2 wheel drive)” and “Rear wheel drive (2 wheel drive). 3. The user clicks on the “4 wheel drive” button. 4. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 5. The system will show that the engine has started, by showing different components being lit up on the dashboard. 6. The user will then be able to see the power of the engine flowing towards the front and back axle of the car and towards the wheels. 7. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (The user wishes to view a front wheel drive car instead)   1. The user clicks on “Front wheel drive (2 wheel drive)” button. 2. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 3. The system will show that the engine has started, by showing different components being lit up on the dashboard. 4. The user will then be able to see the power of the engine flowing only flowing towards the front axle of the car and towards the wheels.   (The user wishes to view a rear wheel drive car instead)   1. The user clicks on the “Rear wheel drive (2 wheel drive)” button. 2. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 3. The system will show that the engine has started, by showing different components being lit up on the dashboard. 4. The user will then be able to see the power of the engine flowing only flowing towards the back axle of the car and towards the wheels. | |

1. **Cooling system simulation**

|  |  |
| --- | --- |
| **Name:** Simulate normal temperature | **ID:** 16 |
| **Stakeholders and Goals:** User – to simulate cooling system under normal temperature | |
| **Description:** A user wants to simulate the cooling system of the car under normal temperature | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to view the cooling system under normal temperature. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Cooling system”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view different components on the screen, which includes the radiator, coolant tank and the engine. There will also be a few buttons available for the user which includes “Simulate normal temperature” and “Simulate high temperature”. 3. The user will then click on “Simulate normal temperature” button in the simulation window. 4. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 5. The system will show that the engine has started, by showing different components being lit up on the dashboard. 6. The system will then show the user the flow of the coolant from the coolant tank to the engine and to the radiator in a continuous loop. 7. The system will also be updating the temperature of the car dynamically. 8. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:** None | |

|  |  |
| --- | --- |
| **Name:** Simulate overheating (high temperature) | **ID:** 17 |
| **Stakeholders and Goals:** User – to simulate cooling system overheating or under high temperature | |
| **Description:** A user wants to simulate the cooling system of the car under high temperature and overheating | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to view the cooling system under high temperature. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Cooling system”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view different components on the screen, which includes the radiator, coolant tank and the engine. There will also be a few buttons available for the user which includes “Simulate normal temperature” and “Simulate high temperature”. 3. The user will then click on “Simulate high temperature” button in the simulation window. 4. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 5. The system will show that the engine has started, by showing different components being lit up on the dashboard. 6. The system will then show the user the flow of the coolant from the coolant tank to the engine and to the radiator in a continuous loop (The rate that the coolant is flowing though the different components would be much slower than the rate of flow of coolant of the previous use case, “Simulate normal temperature”, use case ID: 16). 7. The system will also be updating the temperature of the car dynamically until it reaches 140 degrees Celsius. 8. Within the next 5 seconds, the system will then display to the user that the car is overheating and the engine will shut down. 9. After 5 seconds, the system will then shut off the engine, by turning off the lights on the persistent dashboard. The only lights on the dashboard that will remain on is the **check engine light** and the **high temperature light**. This is to show that the engine has overheated. 10. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:** None | |

1. **Electronic system simulation**
2. **Lights**

|  |  |
| --- | --- |
| **Name:** Turning on/off headlights | **ID:** 18 |
| **Stakeholders and Goals:** User – to simulate turning on/off the headlights | |
| **Description:** A user wants to simulate turning on/off the headlights | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to turn on/off the headlights. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Electrical System”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view the electrical components of the car. There will also be a few buttons available for the user which includes “Lights”, “Audio”, “Wipers”, “Cruise control” and “Defrost rear”. 3. The user will then click on “Lights” button in the simulation window. 4. The system will then display to the user the different kinds of light available by showing more buttons to the user, which includes “Head lights”, “High beam”, “Front fog light”, “Rear fog light” and “Hazard light”. 5. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 6. The system will show that the engine has started, by showing different components being lit up on the dashboard. 7. The user will then click on the “Headlights on” button. 8. The system will then show the flow of power from the battery to the headlights. 9. The system will also update the persistent dashboard to light up the **headlights** symbol. 10. **IF** the user wishes to turn on the other lights (i.e high beam, front fog light or the rear fog lights), it will continue from this point of this use case. | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to turn off the headlights instead, we are assuming the headlights are already on)   1. The user will then click on the “Headlights off” button. 2. The system will then stop the flow of power from the battery to the headlights. 3. The system will also update the persistent dashboard to turn off the headlights symbol. 4. At the same time, if there are any other lights on (e.g high beam, front fog light and rear fog light), the system will also turn them off. | |

|  |  |
| --- | --- |
| **Name:** Turning on/off high beam | **ID:** 19 |
| **Stakeholders and Goals:** User – to simulate turning on/off high beam lights | |
| **Description:** A user wants to simulate turning on/off the high beam lights | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to turn on/off the high beam lights | |
| **Normal Flow:**   1. This use case is continued from a use case detailed before, “Turning on/off headlights”, use case ID: 18. We continue from point number 9 of the normal flow of the use case. At this point, the headlights are already ON. The reason why this use case is continued from before is because the user must turn on the headlights first before being able to turn on the other lights (i.e high beam, front fog light or the rear fog lights). 2. The user will then click on the “High beam on” button. 3. The system will then show the flow of power from the battery to the high beam lights 4. The system will also update the persistent dashboard to light up the **high beam light** symbol. 5. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to turn off the high beam lights instead, we are assuming the high beam lights are already on)   1. The user will then click on the “High beam off” button. 2. The system will then stop the flow of power from the battery to the high beam light. 3. The system will also update the persistent dashboard to turn off the high beam light symbol.   (User HOLDS on high beam lights while the headlights are off. In a car, you usually use the high beam at times to provide a signal to another driver (e.g to move away, or allowing the driver to make a turn)   1. The user will HOLD on the ”High beam on” button. 2. The system will then show the flow of power from the battery to the high beam lights. 3. The system will also update the persistent dashboard to light up the high beam light symbol 4. Once the user stops holding the “High beam on” button, the system will then stop the flow of power from the battery to the high beam lights. 5. The system will also update the persistent dashboard to turn off the high beam light symbol.   (User attempts to turn on the high beam lights without having the headlights turned on)   1. The user attempts to click on the “High beam on” button. 2. The system displays to the user that the headlights should be turned on first before the high beam light can be turned on. 3. The process of turning on the high beam lights are detailed in use case “Turning on/off headlights (Use case ID: 18)” and “Turning on/off high beam (Use case ID: 19)”. | |

|  |  |
| --- | --- |
| **Name:** Turning on/off front fog lights | **ID:** 20 |
| **Stakeholders and Goals:** User – to simulate turning on/off front fog lights | |
| **Description:** A user wants to simulate turning on or off the front fog lights of the car | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to turn on/off the front fog light. | |
| **Normal Flow:**   1. This use case is continued from a use case detailed before, “Turning on/off headlights”, use case ID: 18. We continue from point number 9 of the normal flow of the use case. At this point, the headlights are already ON. The reason why this use case is continued from before is because the user must turn on the headlights first before being able to turn on the other lights (i.e high beam, front fog light or the rear fog lights). 2. The user will then click on the “Front fog light on” button. 3. The system will then show the flow of power from the battery to the front fog light 4. The system will also update the persistent dashboard to light up the **front fog light** symbol. 5. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to turn off the front fog light instead, we are assuming that the front fog lights are already turned on)   1. The user will then click on the “Front fog light off” button. 2. The system will then stop the flow of power from the battery to the front fog light. 3. The system will also update the persistent dashboard to turn off the front fog light symbol.   (User attempts to turn on the front fog light without having the headlights turned on)   1. The user attempts to click on the “Front fog light on” button. 2. The system displays to the user that the headlights should be turned on first before the front fog light can be turned on. 3. The process of turning on the front fog lights are detailed in use case “Turning on/off headlights (Use case ID: 18)” and “Turning on/off front fog lights (Use case ID: 20)”. | |

|  |  |
| --- | --- |
| **Name:** Turning on/off rear fog lights | **ID:** 21 |
| **Stakeholders and Goals:** User – to simulate turning on/off rear fog lights | |
| **Description:** A user wants to simulate turning on or off the rear fog lights of the car | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to turn on/off the rear fog light. | |
| **Normal Flow:**   1. This use case is continued from a use case detailed before, “Turning on/off headlights”, use case ID: 18. We continue from point number 9 of the normal flow of the use case. At this point, the headlights are already ON. The reason why this use case is continued from before is because the user must turn on the headlights first before being able to turn on the other lights (i.e high beam, front fog light or the rear fog lights). 2. The user will then click on the “Rear fog light on” button. 3. The system will then show the flow of power from the battery to the rear fog light 4. The system will also update the persistent dashboard to light up the **rear fog light** symbol. 5. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to turn off the rear fog light instead, we are assuming that the rear fog lights are already turned on)   1. The user will then click on the “Rear fog light off” button. 2. The system will then stop the flow of power from the battery to the rear fog light. 3. The system will also update the persistent dashboard to turn off the rear fog light symbol.   (User attempts to turn on the rear fog light without having the headlights turned on)   1. The user attempts to click on the “Rear fog light on” button. 2. The system displays to the user that the headlights should be turned on first before the rear fog light can be turned on. 3. The process of turning on the rear fog lights are detailed in use case “Turning on/off headlights (Use case ID: 18)” and “Turning on/off rear fog lights (Use case ID: 21)”. | |

|  |  |
| --- | --- |
| **Name:** Turning on/off hazard lights | **ID:** 22 |
| **Stakeholders and Goals:** User – to simulate turning on/off hazard lights | |
| **Description:** A user wants to simulate turning on or off the hazard lights of the car | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to turn on/off the hazard light. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Electrical System”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view the electrical components of the car. There will also be a few buttons available for the user which includes “Lights”, “Audio”, “Wipers”, “Cruise control” and “Defrost rear”. 3. The user will then click on “Lights” button in the simulation window. 4. The system will then display to the user the different kinds of light available by showing more buttons to the user, which includes “Head lights”, “High beam”, “Front fog light”, “Rear fog light” and “Hazard light”. 5. The user will then click on the “Hazard light on” button. 6. The system will then show the flow of power from the battery to the hazard lights 7. The system will also update the persistent dashboard to light up the hazard light symbol. 8. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to turn off the hazard lights instead, we are assuming that the hazard lights are already turned on)   1. The user will then click on the “Hazard light off” button. 2. The system will then stop the flow of power from the battery to the hazard lights. 3. The system will also update the persistent dashboard to turn off the hazard light symbol. | |

1. **Wipers**

|  |  |
| --- | --- |
| **Name:** Turning on/off wipers | **ID:** 23 |
| **Stakeholders and Goals:** User – to simulate turning on/off wipers | |
| **Description:** A user wants to simulate turning on or off the wipers of the car | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to turn on/off the wipers | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Electrical System”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view the electrical components of the car. There will also be a few buttons available for the user which includes “Lights”, “Audio”, “Wipers”, “Cruise control” and “Defrost rear”. 3. The user will then click on “Wiper” button in the simulation window. 4. The system will then display to the user the “Turn wiper on” button (what user see may differ depending on the status of the wiper) and the speed bar (for changing the speed of the wiper, similar to the bar we use to increase/decrease altitude/speed/incline). 5. The user will then click on the “Turn wiper on” button. 6. The system will then show the flow of power from the battery to the wipers to show that the wipers are on 7. The system will also tell the user that the wiper has been turned on by displaying “Wiper is now on!” (How? In “functionality pathway” document, mentioned something called message centre). 8. **IF** the user wishes to increase/decrease the speed of the wipers, it is continued in the next use case “Increasing/Decreasing wiper speed”, Use case ID: 24. | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to turn off the wipers instead, we are assuming that the wipers are already turned on)   1. The user will then click on the “Wipers off” button. 2. The system will then stop the flow of power from the battery to the wipers. 3. The system will also tell the user that the wiper has been turned off by displaying “Wiper is now off!” | |

|  |  |
| --- | --- |
| **Name:** Increasing/decreasing wiper speed | **ID:** 24 |
| **Stakeholders and Goals:** User – to simulate increase/decrease of wiper speed | |
| **Description:** A user wants to simulate increasing or decreasing the speed of the wiper | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to increase/decrease the wiper speed | |
| **Normal Flow:**   1. This use case is continued from a use case detailed before, “Turning on/off wipers”, use case ID: 23. We continue from point number 7 of the normal flow of the use case. At this point, the wipers are already ON. The reason why this use case is continued from before is because the user must turn on the wipers first before being able to increase/decrease the speed of the wipers 2. The user will then increase the speed of the wiper by clicking on the “+” symbol on the speed bar. 3. The system will then show the increase of flow of power from the battery to the wiper by increasing the number of arrows to the flow. 4. The system will also tell the user that the speed of the wiper has been successfully changed via the message centre. 5. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to reduce the speed of the wipers)   1. The user will reduce the speed of the wiper by clicking on the “-” symbol on the speed bar. 2. The system will then show the decrease of flow of power from the battery to the wiper by reducing the number of arrows to the flow. 3. The system will also tell the user that the speed of the wiper has been successfully changed via the message centre.   (User attempts to increase/decrease the speed of the wipers without having the wipers on)   1. The user attempts to click on the “+”/“-” on the speed bar. 2. The system displays to the user that the wipers should be turned on first before being able to increase/decrease the speed of the wipers. 3. The process of increasing/decreasing the speed of the wipers are detailed in use case “Turning on/off wipers (Use case ID: 23)” and “Increasing/decreasing wiper speed (Use case ID: 24)”. | |

1. **Audio system**

|  |  |
| --- | --- |
| **Name:** Turning on/off audio system | **ID:** 25 |
| **Stakeholders and Goals:** User – to simulate turning on/off the audio system | |
| **Description:** A user wants to simulate turning on/off the audio system | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to turn on/off the audio system. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Electrical System”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view the electrical components of the car. There will also be a few buttons available for the user which includes “Lights”, “Audio”, “Wipers”, “Cruise control” and “Defrost rear”. 3. The user will then click on “Audio” button in the simulation window. 4. The system will then display to the user the “Radio on” button, the lists of channel (list of channel is just a list of numbers representing different channels, 6 in total) and volume bar (the volume bar is similar to the bar we use to increase/decrease incline/speed/altitude). 5. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 6. The system will show that the engine has started, by showing different components being lit up on the dashboard. 7. The user will then click on the “Radio on” button. 8. The system will then show the flow of power from the battery to the radio and also to the speakers around the car (4 in total, 1 on each door). 9. The system will also update the user the current channel that it is currently playing and followed by the current volume in the **message centre**. 10. **IF** the user wishes to increase/decrease the volume of the radio or change the channels, it is listed in the next 2 use cases, use case “Increasing/decreasing volume (Use case ID: 26)” and use case “Changing channels (Use Case ID: 27)”. | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to turn off the radio instead, we are assuming the radio is already on)   1. The user will then click on the “Radio off” button. 2. The system will then stop the flow of power from the battery to the radio and also to the speakers around the car. | |

|  |  |
| --- | --- |
| **Name:** Increasing/decreasing volume | **ID:** 26 |
| **Stakeholders and Goals:** User – to increase/decrease the volume | |
| **Description:** A user wants to simulate increasing or decreasing the volume of the radio. | |
| **Actors:** User | |
| **Trigger:** User turns on the radio and wishes to increase/decrease the radio volume. | |
| **Normal Flow:**   1. This use case is continued from a use case detailed before, “Turning on/off audio system”, use case ID: 25. We continue from point number 9 of the normal flow of the use case. At this point, the radio is already ON. The reason why this use case is continued from before is because the user must turn on the radio first before being able to increase/decrease the volume of the radio. 2. The user will then increase the volume of the radio by clicking on the “+” symbol on the volume bar. 3. The system will then represent the increase in volume by increasing the number of arrows/flows from the battery to the speakers around the car. 4. The system will once again update the user the current channel that it is currently playing and followed by the increased volume level in the message centre. 5. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to reduce the volume)   1. The user will then decrease the volume of the radio by clicking on the “-” symbol on the volume bar. 2. The system will then represent the decrease in volume by decreasing the number of arrows/flows from the battery to the speakers around the car. 3. The system will once again update the user the current channel that it is currently playing and followed by the decreased volume level in the message centre.   (User attempts to increase/decrease the volume of the radio without having the radio on)   1. The user attempts to click on the “+”/“-” on the volume bar. 2. The system displays to the user that the radio should be turned on first before being able to increase/decrease the volume of the radio. 3. The process of increasing/decreasing the volume of the radio is detailed in use case “Turning on/off radio (Use case ID: 25)” and “Increasing/decreasing wiper speed (Use case ID: 26)”. | |

|  |  |
| --- | --- |
| **Name:** Changing channels | **ID:** 27 |
| **Stakeholders and Goals:** User – to change channels | |
| **Description:** A user wants to simulate changing the channels on the radio | |
| **Actors:** User | |
| **Trigger:** User turns on the radio and wishes to change the channel | |
| **Normal Flow:**   1. This use case is continued from a use case detailed before, “Turning on/off audio system”, use case ID: 25. We continue from point number 9 of the normal flow of the use case. At this point, the radio is already ON. The reason why this use case is continued from before is because the user must turn on the radio first before being able to change the channels on the radio. 2. The user will then change the channel by clicking on a different number among the lists of channel available for the user. 3. The system will once again update the user the current channel (should be updated to the one user has recently changed) that it is currently playing and followed by the current volume in the message centre. 4. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User attempts to change channels on the radio without having the radio on)   1. The user attempts to click on one of the numbers on the list of channels. 2. The system displays to the user that the radio should be turned on first before being able to change the channels on the radio. 3. The process of changing channels on the radio is detailed in use case “Turning on/off radio (Use case ID: 25)” and “Changing channels (Use case ID: 27)”. | |

1. **Cruise control**

|  |  |
| --- | --- |
| **Name:** Turning on/off cruise control | **ID:** 28 |
| **Stakeholders and Goals:** User – to simulate turning on/off cruise control | |
| **Description:** A user wants to simulate turning on/off cruise control in the car. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to turn on/off cruise control. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Electrical System”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view the electrical components of the car. There will also be a few buttons available for the user which includes “Lights”, “Audio”, “Wipers”, “Cruise control” and “Defrost rear”. 3. The user will then click on “Cruise control” button in the simulation window. 4. The system will then display to the user the “Cruise on” button and cruise speed bar (the cruise speed bar is similar to the bar we use to increase/decrease incline/speed/altitude). 5. In addition to the electrical components of the car (only for cruise control), the user will also be able to see additional components which includes the engine, the fuel tank and also the fuel pump. 6. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 7. The system will show that the engine has started, by showing different components being lit up on the dashboard. 8. The user will also be able to see fuel being pumped from the fuel tank to the engine by the fuel pump. 9. To be able to use turn on the cruise control the user needs to increase the speed of the car to **at least 45km/h**. To do so the user simply clicks the “+” symbol under the “Speed” section (This is on the persistent dashboard) until he/she has reached 45km/h. 10. The system will then display an increase in speed of the car on the dashboard. 11. The user will then click on the “Cruise on” button. 12. The system will then update the persistent dashboard to display the cruise control symbol. 13. The system will also show flow of power/arrows from the battery to the cruise control module. 14. **IF** the user wishes to increase/decrease the cruise speed, it will be detailed in the next use case “Increase/Decrease cruise speed (Use case ID: 29)”. | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to turn off the cruise control instead, we are assuming the cruise control is already on)   1. The user will then click on the “Cruise control off” button. 2. The system will then update the persistent dashboard to turn off the cruise control symbol. 3. The system will then stop the flow of power from the battery to the cruise control module.   (User increases the degree of incline while the cruise control is on)   1. The user will increase the incline by clicking on the “+” symbol on the incline section (This is located below the persistent dashboard). 2. The system will attempt to maintain the speed of the car (The speed where the user turned on the cruise control) by pumping more fuel into the engine. 3. The system will then display an increase in flow of fuel from the fuel tank, to the fuel pump and to the engine.   (User decreases the degree of incline while the cruise control is on)   1. The user will decrease the incline by clicking on the “-” symbol on the incline section (This is located below the persistent dashboard). 2. The system will attempt to maintain the speed of the car (The speed where the user turned on the cruise control) by pumping less fuel into the engine. 3. The system will then display a decrease in flow of fuel from the fuel tank, to the fuel pump and to the engine.   (User increases the altitude while the cruise control is on)   1. The user will increase the altitude by clicking on the “+” symbol on the altitude section (This is located below the persistent dashboard). 2. The system will attempt to maintain the speed of the car (The speed where the user turned on the cruise control) by pumping more fuel into the engine. More fuel is also required to be burnt at higher altitudes. 3. The system will then display an increase in flow of fuel from the fuel tank, to the fuel pump and to the engine.   (User decreases the altitude while the cruise control is off)   1. The user will decrease the altitude by clicking on the “-” symbol on the altitude section (This is located below the persistent dashboard). 2. The system will attempt to maintain the speed of the car (The speed where the user turned on the cruise control) by pumping less fuel into the engine. Less fuel is also required to be burnt at lower altitudes. 3. The system will then display a decrease in flow of fuel from the fuel tank, to the fuel pump and to the engine.   (User attempts to turn on cruise control when the speed of the car is not at 45 km/h)   1. The user will attempt to click on “Cruise control on”. 2. The system will then display to the user that the speed of the car must be at least 45 km/h before being able to turn on cruise control. 3. The process to turn on the cruise control is defined in this use case.   (User attempts to turn on cruise control without starting the engine for the car)   1. The user will attempt to click on “Cruise control on”. 2. The system will then display to the user that the engine of the car must be on and the speed of the car must be at least 45 km/h before being able to turn on cruise control. 3. The process to turn on the cruise control is defined in this use case. | |

|  |  |
| --- | --- |
| **Name:** Increasing/decreasing cruise speed | **ID:** 29 |
| **Stakeholders and Goals:** User – to increase/decrease the cruise speed | |
| **Description:** A user wants to simulate increasing or decreasing the current speed of the cruise. | |
| **Actors:** User | |
| **Trigger:** User turns on the cruise control and wishes to increase/decrease the current speed of the cruise. | |
| **Normal Flow:**   1. This use case is continued from a use case detailed before, “Turning on/off cruise control”, use case ID: 28. We continue from point number 13 of the normal flow of the use case. At this point, the cruise control is already ON. The reason why this use case is continued from before is because the user must turn on the cruise control first before being able to increase/decrease the current cruising speed. 2. The user will then increase the cruising speed by clicking on the “+” symbol on the cruise speed bar. 3. The system will then represent the increase in cruising speed by pumping more fuel from the fuel tank to the fuel pump and to the engine. More fuel is required for the engine to achieve the increased in speed. 4. The system will also update the persistent dashboard with the new speed. 5. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to decrease the cruising speed)   1. The user will then decrease the cruising speed by clicking on the “-” symbol on the cruise speed bar. 2. The system will then represent the decrease in cruising speed by pumping less fuel from the fuel tank to the fuel pump and to the engine. Less fuel is required when the speed is reduced. 3. The system will also update the persistent dashboard with the new speed.   (User attempts to increase/decrease the cruising sped without having the cruise control on)   1. The user attempts to click on the “+”/“-” on the cruise speed bar. 2. The system displays to the user that the cruise control should be turned on first before being able to increase/decrease the cruising speed. 3. The process of increasing/decreasing the cruising speed is detailed in use case “Turning on/off cruise control (Use case ID: 28)” and “Increasing/decreasing cruising speed (Use case ID: 29)”. | |

1. **Defrost rear**

|  |  |
| --- | --- |
| **Name:** Turning on/off rear defroster | **ID:** 30 |
| **Stakeholders and Goals:** User – to simulate turning on/off the rear defroster | |
| **Description:** A user wants to simulate turning on/off the rear defroster | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to turn on/off the rear defroster. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Electrical System”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view the electrical components of the car. There will also be a few buttons available for the user which includes “Lights”, “Audio”, “Wipers”, “Cruise control” and “Defrost rear”. 3. The user will then click on “Defrost rear” button in the simulation window. 4. The system will then display to the user the “Rear defroster on” button. 5. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 6. The system will show that the engine has started, by showing different components being lit up on the dashboard. 7. The user will then click on the “Rear defroster on” button. 8. The system will then show the flow of power from the battery to the rear defroster. 9. The system will then update the persistent dashboard to light up the rear defroster symbol. 10. End. | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to turn off the rear defroster instead, we are assuming the rear defroster is already on)   1. The user will then click on the “Rear defroster off” button. 2. The system will then stop the flow of power from the battery to the rear defroster. | |

1. **Steering system simulation**

|  |  |
| --- | --- |
| **Name:** Simulate the steering system | **ID:** 31 |
| **Stakeholders and Goals:** User – to simulate the steering system | |
| **Description:** A user wants to simulate turning the steering wheel in different directions, either left or right | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to simulate the steering system. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Steering System”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to see different components of the steering system of the car, which includes the steering wheel, the steering pinion, the steering rack and the wheels. The user will also be able to see two buttons, “< (To turn left)” and “> “To turn right” under the “Steer angle” section. 3. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 4. The system will show that the engine has started, by showing different components being lit up on the dashboard. 5. The user then clicks on the “<” button. 6. The system then updates the “Steer angle” column (It will constantly be updated by -15 degrees for every click, the negative symbol representing that it is going left). 7. The system will also update the different components, if the user is turning left, the steering wheel is turned to the left, the steering pinion will rotate towards the left, the steering rack will rotate in the opposite direction of the steering pinion, which is right, and the tire would then turn to the left by 15 degrees. 8. The image that is placed above the “Steer angle” section will also be replaced with another image that shows that the user is steering left. 9. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to turn to the right instead)   1. The user will then click on the “>” button. 2. The system then updates the current steer angle column (It will update by 15 degrees for every click, positive numbers mean that it is going right). 3. The system will also update the different components, if the user is turning right, the steering wheel is turned to the right, the steering pinion will rotate towards the right, the steering rack will rotate in the opposite direction of the steering pinion, which is left, and the tire would then turn to the right by 15 degrees. 4. The image that is placed above the “Steer angle” section will also be replaced with another image that shows that the user is steering right. | |

1. **Simulate braking system**

|  |  |
| --- | --- |
| **Name:** Engaging/Disengaging hand brakes | **ID:** 32 |
| **Stakeholders and Goals:** User – to simulate engaging/disengaging the hand brakes | |
| **Description:** A user wants to simulate engaging or disengaging the hand brakes. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to engage/disengage the hand brakes. | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Braking System”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view the components for the braking system, which includes the battery, the brake pump, the brake fluid, the hand brake and the brake pads. There will also be a few buttons available for the user which includes “Handbrake” and the “Foot brake”. 3. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 4. The system will show that the engine has started, by showing different components being lit up on the dashboard. 5. The user will then click on the “Handbrake on” button (Upon starting the car the handbrakes remain disengaged). 6. The system will then show A flow of power from the battery to the parking brakes (current only goes once to the parking brakes). 7. The system will then show that the parking brakes has been engaged by updating the persistent dashboard to light up the handbrake symbol. 8. End. | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (User wishes to disengage the handbrake)   1. The user will then click on the “Handbrake off” button. 2. The system will then show A flow of power from the battery to the parking brakes (current only goes once to the parking brakes). 3. The system will then show that the parking brakes has been disengaged by updating the persistent dashboard to turn off the handbrake symbol.   \*\*probably need to take into consideration the current speed of the car. The user should be able to engage the handbrake even when the car is moving. Emergency braking. | |

|  |  |
| --- | --- |
| **Name:** Simulating foot brakes | **ID:** 33 |
| **Stakeholders and Goals:** User – to simulate the foot brakes | |
| **Description:** A user wants to simulate the foot brakes | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to simulate the foot brakes | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Braking System”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to view the components for the braking system, which includes the battery, the brake pump, the brake fluid, the hand brake and the brake pads. There will also be a few buttons available for the user which includes “Handbrake” and the “Foot brake”. 3. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 4. The system will show that the engine has started, by showing different components being lit up on the dashboard. 5. The user increase the speed of the car by clicking on the “+” symbol under the “Speed” section (This is on the persistent dashboard) until he/she has reached 60km/h. 6. The system will then display an increase in speed of the car on the dashboard. 7. The user will then CLICK AND HOLD on the “Foot brake” button. 8. The system will then send a signal from the foot brake to the brake pump. 9. The system will also show that the brake pump will constantly pump brake fluid to the 4 wheels of the car. 10. The system will then constantly update the speed of the car on the persistent dashboard at a reducing rate of 15km/h. 11. End. | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:** None | |

1. **Simulate adding attachment**

|  |  |
| --- | --- |
| **Name:** Simulate adding/removing attachments | **ID:** 35 |
| **Stakeholders and Goals:** User – to simulate adding attachments | |
| **Description:** A user wants to simulate adding attachments to a car | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to simulate adding attachments | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Adding attachment”. The window (i.e the simulation window) below the persistent dashboard will then be updated. The user will then be able to see a picture which represents the back view of the car. There will also be a few buttons available for the user which includes “Add tow box”, “Add trailer” and the “Remove attachment” button (In “functionality pathway”, the word used was “unmount”, it doesn’t look like the word exists, so phrase “Remove attachment” used instead”). 3. The user will then click on the “Add tow box” button. 4. The system will then update the image (Image that shows the back of the car) to another image which shows the rear of the car with a tow box attached to it. 5. End. | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  (The user wishes to add a trailer instead, we are assuming that the user has not added any attachments to the car)   1. The user will then click on the “Add trailer” button 2. The system will then update the image (Image that shows the back of the car) to another image which shows the rear of the car with a trailer attached to it.   (The user wishes to remove any attachments that are on the car)   1. The user then click on the “Remove attachment” button. 2. The system will then update the image (Image that shows the back of the car WITH an attachment, could be either the tow box or the trailer) to the original image which only shows the rear of the car.   (The user wishes to add an attachment when there is already an attachment on the car)   1. The user attempts to click on “Add tow box”/”Add trailer” button (We are assuming that either the tow box or the trailer has already been attached to the back of the car) 2. The system then displays to the user that there is already an existing attachment on the car and should remove it first before being able to add another attachment to the car. 3. The process to removing an attachment to the car is explained in this use case (The section right above of this, “The user wishes to remove any attachments that are on the car”). | |

1. **Simulate alternative fuel powered drivetrains**

|  |  |
| --- | --- |
| **Name:** Simulate a hybrid car | **ID:** 36 |
| **Stakeholders and Goals:** User – to simulate a hybrid car | |
| **Description:** A user wants to simulate a hybrid car instead of a normal petrol car. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to simulate a hybrid car | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Alternative fuel powered drivetrains”. The window (i.e the simulation window) below the persistent dashboard will then be updated. There will be a few buttons available for the user which includes “Hybrid”, “Electric” and the “Hydrogen” button. 3. The user will then click on the “Hybrid” button to simulate a hybrid car. 4. The system will show different components on the screen that will represent a hybrid car, which includes the battery, the engine, fuel tank, fuel pump, battery powered motor, and also the wheels 5. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 6. The system will show that the engine has started, by showing different components being lit up on the dashboard. 7. The system will then show fuel flowing from the fuel tank through the fuel pump and to the engine. At the same time, power from the battery (which is placed at the back of the car) will also flow to the motor which is placed at the front part of the car. Both the engine and the motor will power the car and rotate the front wheels (This happens as the car is moving at an INCREASING speed). 8. End. | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  **Different things happen in a hybrid car in different situations compared to a petrol car.**  (When the car is at constant speed)   1. The system will then show the rotation of the back wheels sending power to the battery (which is placed at the back of the car). The rotation of the back wheels is recharging the battery.   (When the car remains static)   1. The system will then stop the flow of fuel from the fuel tank to the fuel pump and to the engine. The power from the battery will remain flowing to the motor. The engine has been shut down to save power.   (When the user slows down the car or applies the brake)   1. The system will then show the rotation of the back wheels sending power to the battery. The kinetic energy from the car which is slowing down is converted to heat energy and is recharging the battery of the car (Regenerative braking).   \*\*This use case needs work. | |

|  |  |
| --- | --- |
| **Name:** Simulate a hydrogen car | **ID:** 37 |
| **Stakeholders and Goals:** User – to simulate a hydrogen car | |
| **Description:** A user wants to simulate a hydrogen car instead of a normal petrol car. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to simulate a hydrogen car | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Alternative fuel powered drivetrains”. The window (i.e the simulation window) below the persistent dashboard will then be updated. There will be a few buttons available for the user which includes “Hybrid”, “Electric” and the “Hydrogen” button. 3. The user will then click on the “Hydrogen” button to simulate a hydrogen car. 4. The system will show different components on the screen that will represent a hydrogen car, which includes hydrogen tank, fuel cell stacks, motor and the wheels. 5. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 6. The system will show that the engine has started, by showing different components being lit up on the dashboard. 7. The system will then show hydrogen flowing from the hydrogen tank to the fuel stack cells which then produces power and is sent to the front motors to move the front wheels. 8. End. | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows: None** | |

|  |  |
| --- | --- |
| **Name:** Simulate an electric car | **ID:** 38 |
| **Stakeholders and Goals:** User – to simulate an electric car | |
| **Description:** A user wants to simulate an electric car instead of a normal petrol car. | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to simulate an electric car | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Simulation” button. The user will be brought to the simulation page where the user will be able to view the persistent dashboard and also a list of simulations. 2. On the list of simulations available to the user, the user will then click on “Alternative fuel powered drivetrains”. The window (i.e the simulation window) below the persistent dashboard will then be updated. There will be a few buttons available for the user which includes “Hybrid”, “Electric” and the “Hydrogen” button. 3. The user will then click on the “Electric” button to simulate an electric car. 4. The system will show different components on the screen that will represent an electric car, which includes the battery, motor and the wheels. 5. On the persistent dashboard the user will be able to see the “Start Engine” button (What the user may see may differ depending if the engine of the car is on/off). The user will then click on it to start the engine. 6. The system will show that the engine has started, by showing different components being lit up on the dashboard. 7. The system will then show power flowing from the battery towards the motor which is located at the front of the car and then rotating the front wheels. . 8. End. | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  **Different things happen in an electric car in different situations compared to a petrol car.**  (When the car is at constant speed)   1. The system will then show the rotation of the back wheels sending power to the battery (which is placed at the back of the car). The rotation of the back wheels is recharging the battery.   (When the user slows down the car or applies the brake)   1. The system will then show the rotation of the back wheels sending power to the battery. The kinetic energy from the car which is slowing down is converted to heat energy and is recharging the battery of the car (Regenerative braking).   \*\*This use case needs work. | |

**Quiz**

|  |  |
| --- | --- |
| **Name:** Take a quiz | **ID:** 39 |
| **Stakeholders and Goals:** User – to take a quiz | |
| **Description:** A user wants to take a quiz to test his/her understanding on car processes and components | |
| **Actors:** User | |
| **Trigger:** User runs the application and wishes to take a quiz | |
| **Normal Flow:**   1. User reaches the main page of the system and clicks on the “Quiz” button. Upon clicking the quiz button, the user will be brought into the quiz page. 2. The user will be prompted to choose the difficulty of the quiz he wishes to work on (At the moment there will be no difficulty, stretch goal). 3. Upon selecting the difficulty, a progress bar will appear on the screen to show that the system is preparing the questions to be asked for the quiz. The quiz will then start after the loading is complete. A timer will start. 4. The user will then answer the questions by clicking on the answer the user thinks that it is correct. The user will be able to use the keyboard as a form of input for the answers, numbers will represent each answer of the question (i.e 1 for A, 2 for B). 5. The user then clicks the next button 6. The system will show the next question. 7. The user will be able to have quick access to different questions by clicking on the quick access bar which will be placed above the question. 8. Steps 4 to 7 are repeated until the users have answered all the questions 9. Upon completion of all the questions available the user will be able to click on “Complete Quiz” button which will appear on the top right of the question box. 10. The user will then be brought to the “Review quiz” page, which will be the next use case. | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:**  **(After the quiz has been started from step 4 onwards)**   1. If the user wishes to quit halfway during the quiz, the user will be able to click on a “X” button which will be present on the top right of the screen. No statistics will be saved. 2. The systems prompts for reconfirmation to quit the quiz. 3. The user enters confirmation. 4. The system brings the user back to the main menu. | |

|  |  |
| --- | --- |
| **Name:** Review quiz | **ID:** 40 |
| **Stakeholders and Goals:** User – to review the quiz he/she has completed | |
| **Description:** A user wants to gain feedback (correct and wrongly answered questions) on the quiz that he/she has completed. | |
| **Actors:** User | |
| **Trigger:** User finishes the quiz and wishes to review his/her progress. | |
| **Normal Flow:**   1. Continuing from the previous use case, the user will now be on the “Review page” and user will be able to see his/her final score for the quiz. 2. The interface for reviewing the questions will be similar to the interface of the quiz. The user will then be able to click on the quick access bar to access different previously answered questions. Different colour on the quick access bar will represent if the user has answered the questions correctly or wrongly (Green for correctly answered questions, red for wrongly answered questions). 3. Upon entering a wrongly answered question, the answer that the user has chosen (which is the wrong one, if it is not clear enough) will be highlighted in red and the correct answer will be highlighted in green. If the user enters a question that he/she has already answered correctly, the user will see the correct answer (which is also the answer that he/she has chosen) in green. 4. The user will then be able to click the “X” button which will be on the top right corner to quit the quiz review and go back to the main menu. 5. End | |
| **Sub-Flows:** None | |
| **Alternative/Exceptional Flows:** None | |

# Change log of Use Cases

|  |  |
| --- | --- |
| **Date** | **Changes made** |
| 16/5/2015 | * Use case 1, updated to zoom in, use the up and down * Use case 2 updated * Use case 3 update to be linked from previous use case. Since video links can only been seen under the description of the video. Updated alternative * Use case 4 added * Use case 5 updated, updated from number 4 to 5, updated on what the user needs to do in order to view different wheel drive system * Use case “Take a quiz”, updated how one will take a quiz * Use case “Review quiz”, added today * Split use case into 3 different part and added change log |

State Diagrams

State diagrams also act as a good source of information in the development process, as it assists the user to identify the various states a certain object can be in. This would assist the developer as they would be able to best choose the type of variables used and the variables that are needed in order to realise the various states that have been defined in the state diagrams.

The following pages show the various state diagrams that are part of our application. There may be some changes in the state diagrams seen here, as we may have to add/remove/modify some states when we proceed in the development process.

State diagrams go here