Car Dashboard

-The code imports the required QtQuick and QtQuick Controls modules for building the user interface.

-Window defines the main application with a width of 1200 pixels, and a height of 800 pixels, and sets it to be initially visible. Sets the title of the window to "Vehicle".

-Properties define various aspects of the vehicle, including engine state, speed, acceleration and braking rates, acceleration and braking status, turning direction, volume level, current date, and fuel level.

-The **updateGear** function takes the current speed of the vehicle as input and updates the displayed gear accordingly by setting the gear text based on specific speed ranges.

-The **fuelConsumptionTimer** decrements the **fuelLevel** property every second, simulating fuel consumption, and stops when the fuel level reaches or falls below 0, resetting the fuel level, current speed will be 0, and gear to "N".

-The **accelerateTimer** increases the vehicle's current speed by the defined acceleration rate (**accelerationRate**) when the accelerating property is true and there is fuel available, ensuring the speed does not exceed the maximum speed.

-The **brakeTimer** decreases the vehicle's current speed by the defined braking rate (**brakingRate**) when the braking property is true, ensuring the speed does not fall below the minimum speed, and sets the gear to "N" to indicate the vehicle is in neutral after braking.

-Outer rectangle serves as the background of the dashboard and fills the entire parent element. Its color is set to #2a2b26, a dark grayish color.

-The inner rectangle with the id dashboard fills the parent element with a white background color, rounded corners, and an initialAlpha property set to 0 for controlling the opacity of the gradient animation.

-The SequentialAnimation named gradientAnimation, running when the engine is on, loops once, and contains actions to control the animation based on the engine state, reset the initialAlpha property, and animate its value from 0 to 1 over 1 second.

-The gradient property defines a gradient for the dashboard rectangle's background, with colors and opacities dependent on the engine state and initialAlpha property.

-The mainScreen Item contains a Text element displaying the title "Car Dashboard" in black, centered horizontally and anchored to the top, with a font size of 24 pixels.

-The Rectangle representing the gear display area is a black square with rounded corners, anchored to the top-left corner of the parent with specific margins for positioning.

-Inside the gear display rectangle, the gearText element displays the current gear, defaulted to "N" for Neutral, with a font size of 24 pixels and font family "Impact". The text color is yellow when the engine is on and light green when off, anchored to the bottom center of the parent rectangle with specific margins for positioning.

- An Image element displays the gear image sourced from "images/Gear.png" with PreserveAspectFit fill mode and a fixed width and height of 150 units, centered both horizontally and vertically within the parent rectangle.

-**Fuel Consumption Functions:**
Two functions are defined here:

consumeFuel(amount): This function is responsible for simulating fuel consumption. It decreases the fuelLevel by the specified amount if the engine is on, ensuring that the fuel level doesn't go below 0.
refillFuel(): This function refills the fuel tank to full (fuel level set to 100).

- A Rectangle element represents the fuel gauge, styled as a rounded square with dimensions 350x350 units, filled with a black color. Inside, an Image element displays the fuel gauge image sourced from "images/Fuel.png" with PreserveAspectFit fill mode and dimensions 270x270 units, centered within the rectangle.

- A small green Rectangle element inside the fuel gauge rectangle, representing the current fuel level indicator, with dimensions 40x20 units, centered within its parent.

- A Text element displaying the current fuel level percentage, with white text color, font size of 14 pixels, and centered horizontally and vertically within its parent, the fuel gauge rectangle.

- A Text element with id id_date displaying the current date, font size of 24 pixels, font family "Impact", horizontally centered within its parent, anchored to the bottom with a margin of 50 units, with text color yellow when engineOn is true and light green otherwise.

-A Text element with id id_clock displaying the current time, font size of 24 pixels, font family "Impact", horizontally centered within its parent, anchored to the bottom with a margin of 10 units, with text color yellow when engineOn is true and light green otherwise. The fuel gauge is anchored to the top-right corner of the parent window with specific margins for positioning.

- A Button anchored to the bottom-right corner of the parent window with specific margins, displaying "Refill Fuel" text, setting fuelLevel to 100 when clicked, enabled only when engineOn is true.

- An Image element (id: speedometer) displaying the vehicle's speedometer with a source of "images/speedo.png", fixed width and height of 400 units, centered within its parent window using anchors.centerIn.

- An Image element (id: needle) representing the speedometer needle with a source of "images/needlered.png", anchored to the center of the speedometer, vertically centered with an offset, with transformOrigin set to Item.Bottom, and rotation animated based on changes in the currentSpeed property.

- An Image element (id: turnIndicator) displaying the vehicle's turn indicator with a source dependent on engineTurningLeft and engineTurningRight states, width and height set to 50 units, horizontally centered relative to the speedometer, positioned just below it with a margin of 10 units, and visibility controlled based on the vehicle's turning direction.

-**leftTurnButton (Toggle Left Turn):** Toggles the left turn indicator when clicked, enabled only when the engine is on.

-**rightTurnButton (Toggle Right Turn):** Toggles the right turn indicator when clicked, enabled only when the engine is on.

-**engineButton(Turn Engine off/Turn Engine On):** Toggles the engine on or off when clicked. Resets speed and gear text when turning off the engine.

-**accelerateButton(Acceleration):** Controls acceleration. Starts or stops the accelerateTimer based on button press, enabled only when the engine is on.

-**brakeButton(Brake):** Controls braking. Starts or stops the brakeTimer based on button press, enabled only when the engine is on.

-**States and Transitions:** Handles the "EngineOff" state where the engine is turned off, resetting the rotation of the needle. Animates the needle rotation when transitioning to the "EngineOff" state for visual feedback.

-**Volume Controls Button:** Text set to "Volume Controls", behavior toggles visibility of dashboard and control screen based on engine state, enabled only when engine is on, horizontally centered within parent window with a top margin of 60 pixels.

-**Controls Screen (Rectangle):** ID set to "controlsScreen", initially invisible, occupies entire parent window, green color, title text "Volume Controls" centered at the top, volume slider centered horizontally and vertically, allows volume adjustment from 0 to 100, back button positioned at bottom center, toggles visibility of dashboard and control screen when clicked.

Code:

```
import QtQuick 2.15
import QtQuick.Controls 2.15

Window {
    width: 1200
    height: 800
    visible: true
    title: "Vehicle"

    property bool engineOn: false
    property int currentSpeed: 0
    property int maxSpeed: 100
    property int minSpeed: 0
    property int accelerationRate: 1
    property int brakingRate: -1
    property bool accelerating: false
    property bool braking: false
    property bool engineTurningLeft: false
    property bool engineTurningRight: false
    property int volume: 50
    property var currentDate: new Date()
    property int fuelLevel: 100
```

```
//Gear functionality
function updateGear(speed) {
        if (speed === 0) {
            gearText.text = "N"; // Park when speed is zero
        } else if (speed < 20) {
            gearText.text = "1"; //Gear 1
        } else if (speed < 40) {
            gearText.text = "2"; //Gear 2
        }else if (speed < 60) {
            gearText.text = "3"; //Gear 3
        }else if (speed < 80) {
            gearText.text = "4"; //Gear 4
        }else if (speed < 260) {
            gearText.text = "5"; //Gear 5
        } else {
            gearText.text = "R"; // Reverse for high speeds
        }
    }


Timer {
    id: fuelConsumptionTimer
    interval: 1000
    repeat: true
    running: engineOn // Only consume fuel when the engine is on
    onTriggered: {
        fuelLevel -= 1; // Simulate fuel consumption
        if (fuelLevel <= 0) {
                fuelLevel = 0;
                currentSpeed = 0; // Set speed to 0 when fuel level reaches 0
                gearText.text = "N"; // Set gear to "N" when fuel level reaches 0
                }
    }
}
```

```
    Timer {
        id: accelerateTimer
        interval: 100
        repeat: true
        running: accelerating
        onTriggered: {
        if (fuelLevel > 0) {
            var previousSpeed = currentSpeed;
            currentSpeed = Math.min(currentSpeed + accelerationRate,
maxSpeed);
            updateGear(currentSpeed, previousSpeed);
         }

        }
    }

    Timer {
        id: brakeTimer
        interval: 100
        repeat: true
        running: braking
        onTriggered: {
            currentSpeed = Math.max(currentSpeed + brakingRate, minSpeed);
            gearText.text = "N";
        }

    }
    Rectangle {
       anchors.fill: parent
        color: "#2a2b26"

        Rectangle {
            id: dashboard
            color: "#ffffff"
            anchors.fill: parent
```

```
        radius: width / 2

        property real initialAlpha: 0

        SequentialAnimation {
            id: gradientAnimation
            running: engineOn // Start animation when engine is turned on
            loops: 1
            PropertyAction { target: gradientAnimation; property: "running" } // Start/stop animation based on engine state
            PropertyAction { target: dashboard; property: "initialAlpha"; value: 0 } // Reset initialAlpha property
            NumberAnimation { target: dashboard; property: "initialAlpha"; to: 1; duration: 1000 } // Adjust duration as needed
          }
        // Bind the color of each gradient stop to the initial alpha value and engine state
        gradient: Gradient {
          GradientStop { position: 0.00; color: engineOn ? Qt.rgba(1, 1, 0, dashboard.initialAlpha) : "#008000" }
          GradientStop { position: 0.40; color: engineOn ? Qt.rgba(0.466, 0.694, 0.137, dashboard.initialAlpha) : "#23b278" }
          GradientStop { position: 0.60; color: engineOn ? Qt.rgba(0.569, 0.961, 0.012, dashboard.initialAlpha) : "#2f795b" }
          GradientStop { position: 0.80; color: engineOn ? Qt.rgba(0.278, 0.478, 0.192, dashboard.initialAlpha) : "#2a2727" }
          GradientStop { position: 0.90; color: engineOn ? Qt.rgba(0.192, 0.729, 0.392, dashboard.initialAlpha) : "#2a2727" }
        }

        Item {
            id: mainScreen
            anchors.fill: parent

            // Title
```

```
            Text {
                text: "Car Dashboard"
                font.pixelSize: 24
                color: "black"
                anchors.horizontalCenter: parent.horizontalCenter
                anchors.top: parent.top
            }
        }

    // Gare part
    Rectangle {
        width: 350
        height: 350
        radius: width / 2 // Make it a circle by setting the radius to half of the
width
        color: "black"
        anchors {
            top: parent.top
            topMargin: 203
            left: parent.left
            leftMargin: 90
          }

        Text {
            id: gearText
            text: "N" // Default to Neutral gear
            font.pixelSize: 24
            font.family: "Impact"
            color: engineOn ? "yellow" : "lightgreen"
            anchors {
                bottom: parent.bottom
                bottomMargin: 60
                horizontalCenter: parent.horizontalCenter
            }
          }
```

```
            Image {
                source: "images/Gear.png"
                fillMode: Image.PreserveAspectFit // Preserve the aspect ratio of the
image
                width: 150
                height: 150
                anchors {
                    horizontalCenter: parent.horizontalCenter
                    verticalCenter: parent.verticalCenter
                        }
             }

            }

        // Fuel part

        function consumeFuel(amount) {
                if (engineOn) {
                    fuelLevel -= amount;
                    if (fuelLevel < 0){
                        fuelLevel = 0; // Ensure fuel level doesn't go below 0
                    }
                }
            }

        function refillFuel() {
            fuelLevel = 100; // Refill tank to full
        }

        Rectangle {
            width: 350
            height: 350
            radius: width / 2 // Make it a circle by setting the radius to half of the
width
            color: "black"
```

```
        Image {
            source: "images/Fuel.png"
            fillMode: Image.PreserveAspectFit // Preserve the aspect ratio of the
image
            width: 270
            height: 270
            anchors.centerIn: parent

            Rectangle {
                width: 40
                height: 20
                color: "green"
                anchors.centerIn: parent
                    }

            Text {
                text: fuelLevel + "%"
                color: "white"
                font.pixelSize: 14
                anchors.centerIn: parent
            }

            Text {
                id: id_date
                anchors {
                    horizontalCenter: parent.horizontalCenter
                    bottom: parent.bottom
                    bottomMargin: 50
                }
                color: engineOn ? "yellow" : "lightgreen"
                font.pixelSize: 24
                font.family: "Impact"
                text: currentDate.getDate() + "/" + (currentDate.getMonth() + 1) +
"/" + currentDate.getFullYear()
            }
```

```
            Text {
                id: id_clock
                anchors {
                    horizontalCenter: parent.horizontalCenter
                    bottom: parent.bottom
                    bottomMargin: 10
                }
                color: engineOn ? "yellow" : "lightgreen"
                font.pixelSize: 24
                font.family: "Impact"
                text: currentDate.getHours() + ":" + currentDate.getMinutes() + ":" +
currentDate.getSeconds()
            }
        }

        anchors {
            top: parent.top
            right: parent.right
            topMargin: 200
            rightMargin: 90
        }
    }

    Button {
            text: "Refill Fuel"
            anchors {
                right: parent.right
                bottom: parent.bottom
                bottomMargin: 140
                rightMargin: 20
            }
            onClicked: {
                fuelLevel = 100; // Call the function to refill fuel
            }
```

```
            enabled: engineOn
        }

    // Speedometter
    Image {
        id: speedometer
        source: "images/speedo.png"
        width: 400
        height: 400
        anchors.centerIn: parent

        Image {
            id: needle
            source: "images/needlered.png"
            anchors.centerIn: parent
            height: 140
            anchors.verticalCenterOffset: -height / 2
            transformOrigin: Item.Bottom
            Behavior on rotation {
                NumberAnimation {
                    duration: 4000 // Duration of rotation animation in milliseconds
                }
            }
            rotation: -136 + currentSpeed * 2.72 // Adjust rotation based on speed
        }
    }

    // Turn Indicator
    Image {
        id: turnIndicator
        source: engineTurningLeft ? "images/Leftturn.png" : (engineTurn-
ingRight ? "images/Rightturn.png" : "")
        width: 50
        height: 50
        anchors.horizontalCenter: speedometer.horizontalCenter
```

```
            anchors.top: speedometer.bottom
            anchors.topMargin: 10
            visible: engineTurningLeft || engineTurningRight // Show only when turning
        }

        // Left turn button
        Button {
            id: leftTurnButton
            text: "Toggle Left Turn"
            anchors {
                left: parent.left
                bottom: parent.bottom
                bottomMargin: 80
                leftMargin: 20
            }
            onClicked: {
                if (engineOn) {
                    engineTurningLeft = !engineTurningLeft; // Toggle left turn indicator
                    engineTurningRight = false; // Ensure right turn indicator is off
                }
            }
            enabled: engineOn // Only enabled when engine is on
        }

        // Right turn button
        Button {
            id: rightTurnButton
            text: "Toggle Right Turn"
            anchors {
                right: parent.right
                bottom: parent.bottom
                bottomMargin: 80
                rightMargin: 20
```

```
            }
            onClicked: {
                if (engineOn) {
                    engineTurningRight = !engineTurningRight; // Toggle right turn indi-
cator
                    engineTurningLeft = false; // Ensure left turn indicator is off
                }
            }
            enabled: engineOn // Only enabled when engine is on
        }

        Button {
            id: engineButton
            text: engineOn ? "Turn Engine Off" : "Turn Engine On"
            anchors {
                horizontalCenter: parent.horizontalCenter
                bottom: parent.bottom
                bottomMargin: 40
            }
            onClicked: {
                engineOn = !engineOn;
                if (!engineOn) {
                    currentSpeed = 0;
                    gearText.text = "N";
                }
            }
        }

        Button {
            id: accelerateButton
            text: "Accelerate"
            anchors {
                left: parent.left
                bottom: parent.bottom
                bottomMargin: 20
```

```
            leftMargin: 20
        }
        onPressedChanged: {
            accelerating = pressed;
            if (pressed) accelerateTimer.start();
            else accelerateTimer.stop();
        }
        enabled: engineOn
    }

    Button {
        id: brakeButton
        text: "Brake"
        anchors {
            right: parent.right
            bottom: parent.bottom
            bottomMargin: 20
            rightMargin: 20
        }
        onPressedChanged: {
            braking = pressed;
            if (pressed) brakeTimer.start();
            else brakeTimer.stop();
        }
        enabled: engineOn
    }

    states: [
        State {
            name: "EngineOff"
            PropertyChanges {
                target: needle
                rotation: -136 // Rotate needle to minimum speed when engine is
off
            }
```

```
            }
        ]

        transitions: [
            Transition {
                from: "*"
                to: "EngineOff"
                NumberAnimation {
                    target: needle
                    property: "rotation"
                    duration: 1000 // Duration of needle rotation animation
                    easing.type: Easing.InOutQuad // Optional easing function
                }
            }
        ]

    }
}
    Button {
        text: " Volume Controls"
        onClicked: {
            dashboard.visible = false;
            controlsScreen.visible = engineOn;
        }
        enabled: engineOn // Enable the button only when the engine is on
        anchors {
                horizontalCenter: parent.horizontalCenter // Center the button hori-
zontally
                top: parent.top // Align the top of the button with the top of its parent
                topMargin: 60 // Add a margin of 20 pixels from the top
            }
    }

    // Controls screen
    Rectangle {
```

```
id: controlsScreen
visible: false
width: parent.width
height: parent.height
radius: width / 2
color: "#99b739"

// Title
Text {
    id:cartext
    text: "Volume Controls"
    font.pixelSize: 24
    color: "black"
    anchors.horizontalCenter: parent.horizontalCenter
    anchors.top: parent.top
    anchors.topMargin: 20
}

// Volume slider
Slider {
    width: 200
    height: 20
    anchors.horizontalCenter: parent.horizontalCenter
    anchors.verticalCenter: parent.verticalCenter
    orientation: Qt.Horizontal
    from: 0
    to: 100
    value: volume
    onValueChanged: volume = value; // Update volume value
}

// Back button
Button {
    text: "Back"
    anchors {
```

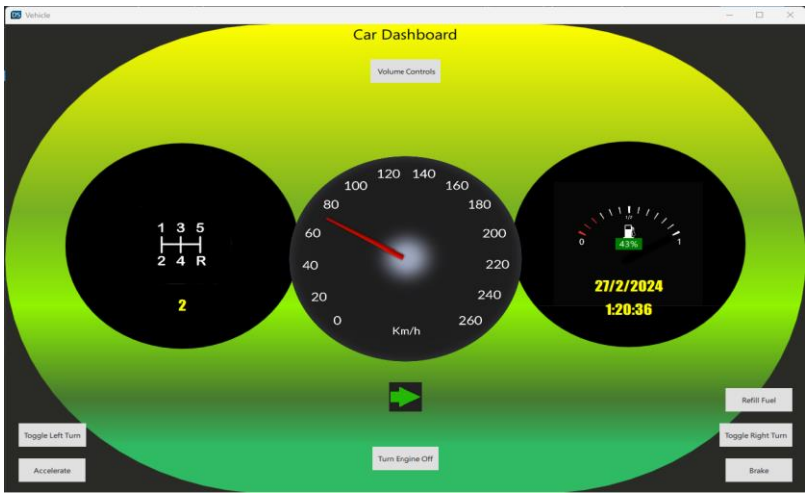
```
            horizontalCenter: parent.horizontalCenter
            bottom: parent.bottom
            bottomMargin: 20
        }
        onClicked: {
            dashboard.visible = true;
            controlsScreen.visible = false;
        }
    }
  }
}
```

Screenshots:

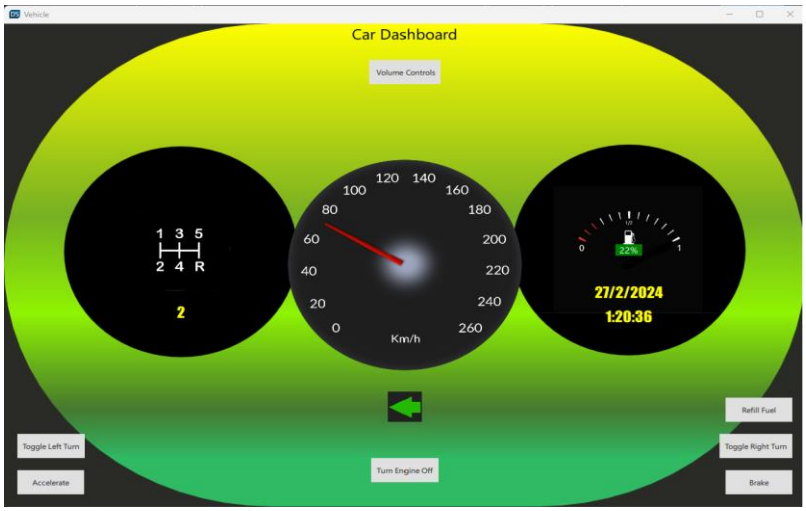
First screen. Because the engine is off no Button will work. Fuel doesn't reduce when the engine is off.

When the engine on all Button start to work. Gear, date and time text color change. The dashboard appears to light up as its background gradient smoothly transitions from transparent to fully opaque when the engine is turned on. Also, fuel reduction.

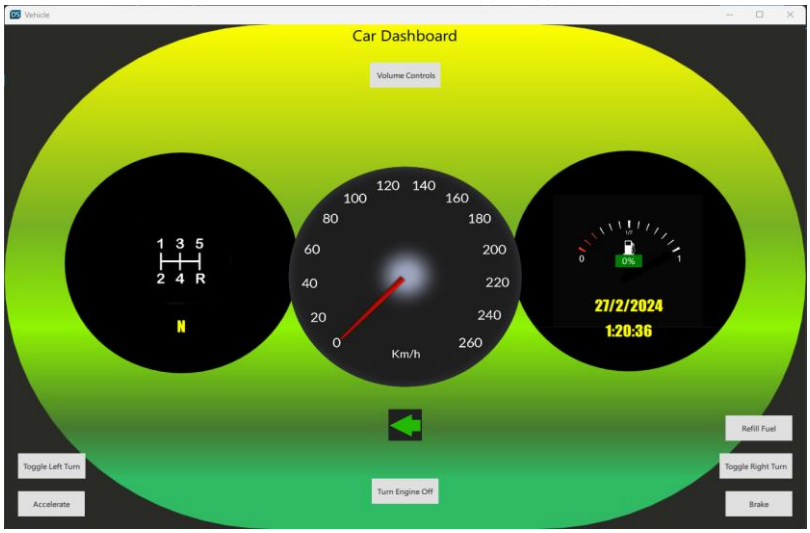
When press Accelerate button speed increases and gear change



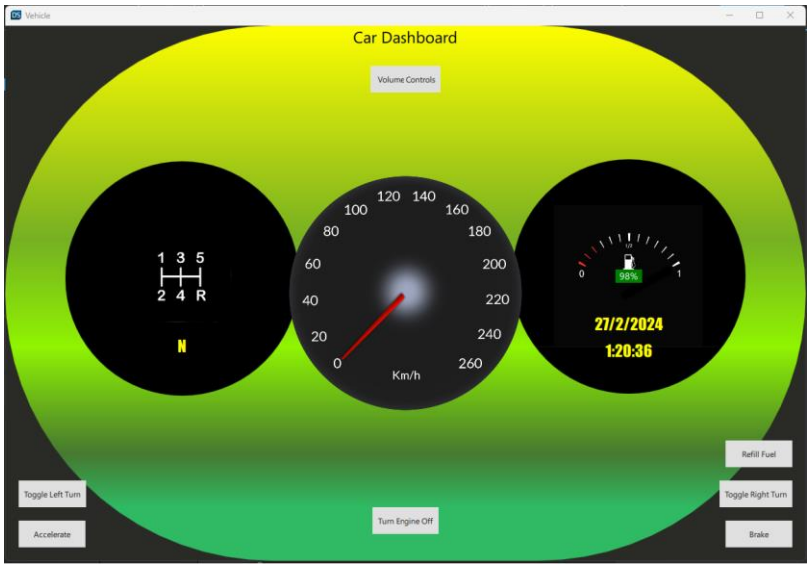
When click “Toggle Right Turn” button the right indicator shows. When clicked a second time right indicator will be removed.



When click “Toggle Left Turn” button the left indicator shows. When clicked the second time left indicator will be removed.



When fuel comes to 0, speed will be 0 and gear will be N (Neutral)



Refill the fuel.

Volume Control Button Navigate to controlScreen. Here can control the volume. Back button returns to the main Screen.