# Mobile App Hardware and Software Requirements

Hardware Requirements:

The application shall run on Android devices running 4.2 or later.

Software Requirements:

The application shall support a minimum SDK version of 17.

The application shall target SDK version 25.

# Mobile App Software Design

## Top Level Functions for the App

For this application there will be a single activity that the user will interact with by entering numbers into the EditText boxes, using increment and decrement buttons, selecting radio buttons, and computing the results using a Compute button.

There will be two EditText views that will be used to enter the distance of the trip and the speed limit.

An increment and decrement button will change the desired amount of speeding. The value of the desired amount of speeding will be displayed between the increment and decrement buttons in a TextView.

The environment will be selected using four buttons in a radio group. Each radio button will change the environment and will determine the algorithm for the total time of stops along the trip.

Lastly, once all of the values have been entered and selected the user will press the compute button, which will display the total time of the trip, the amount of time saved or lost by speeding, and an entertaining animation of wheels turning.

## Design of the User Interface

The figure below shows the design of the user interface for the application. All of the values and computations are done within the single activity. After defining the functions of the application, the various views were added to the design window and repositioned for the cleanest appearance. Instead of using RelativeLayout, it was found that AbsoluteLayout allows the UI to be customized more easily and with better results.

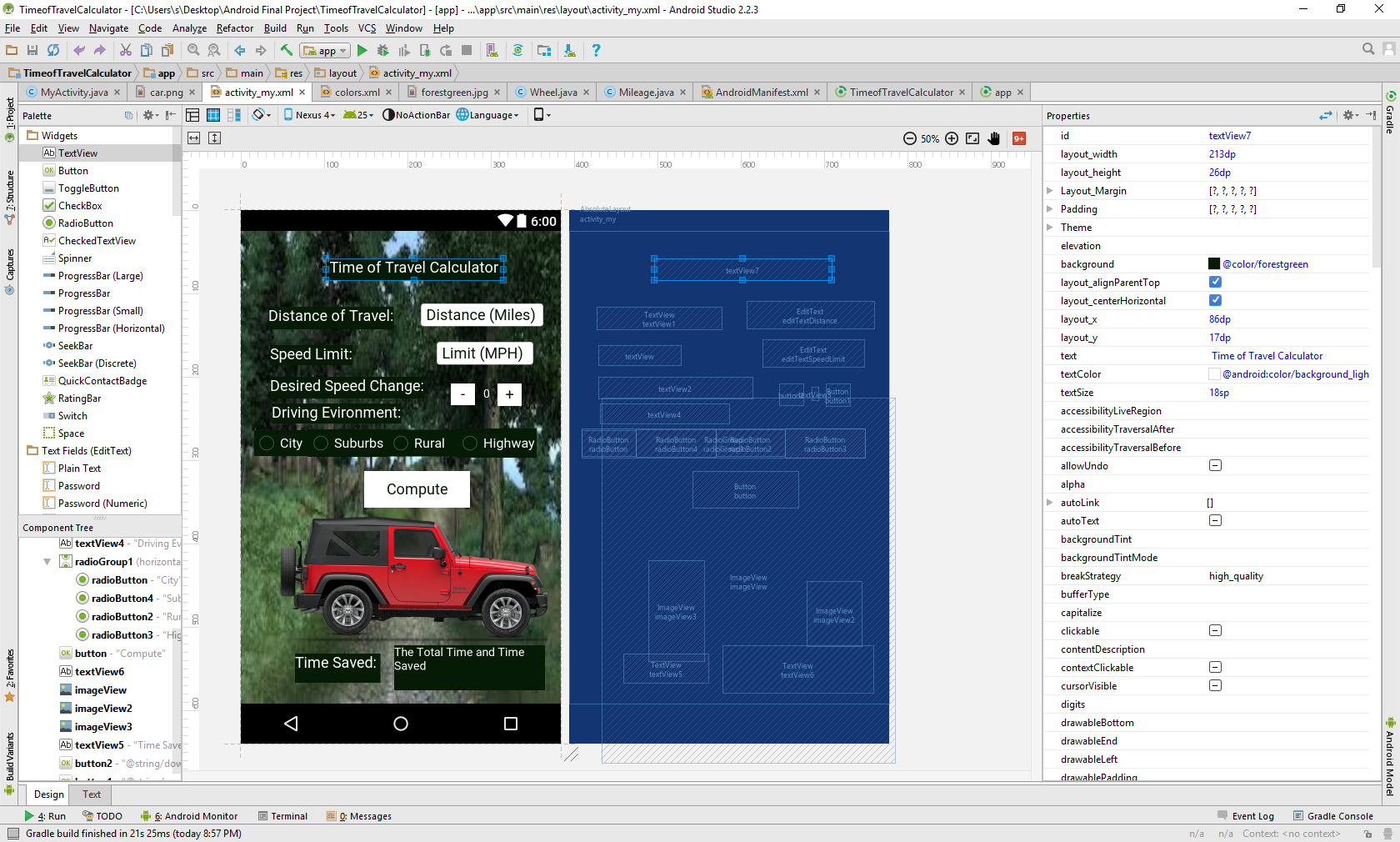


Figure : User Interface

## The State Diagram of the App

Below is the state diagram for the application showing the stages of its lifecycle.



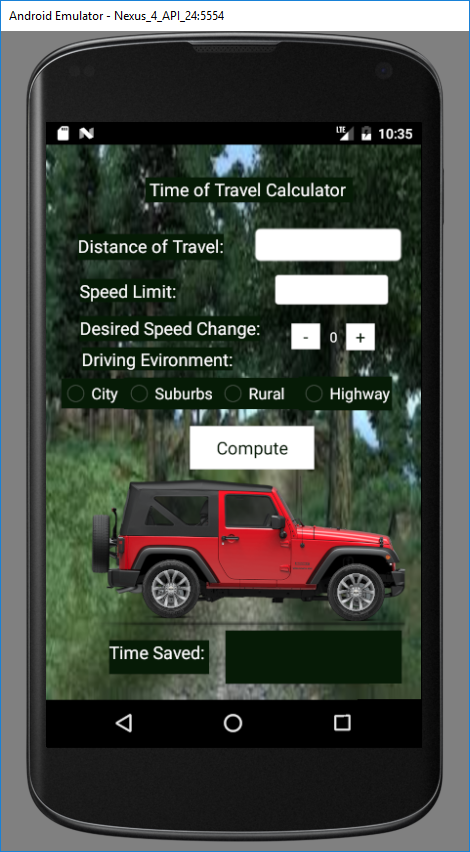
## Class Definitions (UML Class Diagram)

Below is the UML class diagram for the application showing the three java classes for the app. MyActivity is the main class for the application, and will contain and utilize a Wheel object and a Mileage object. The fields and methods are listed in the UML diagram including the input arguments for the methods and the return types.

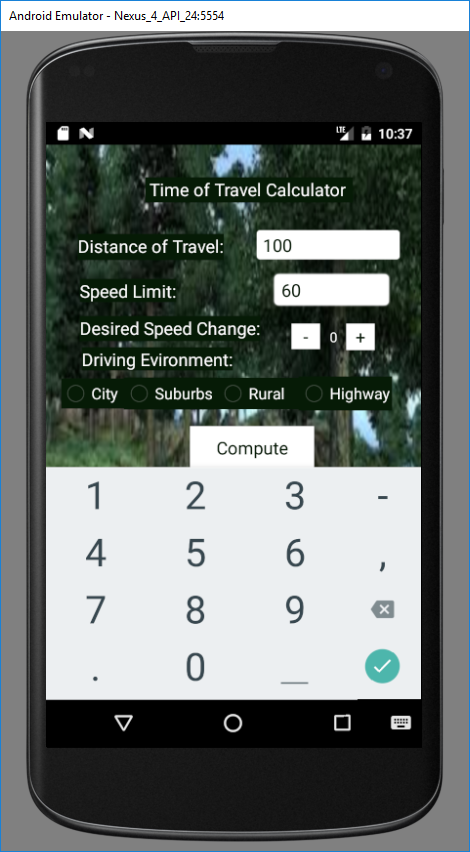


# Mobile App Software Testing and Demonstration

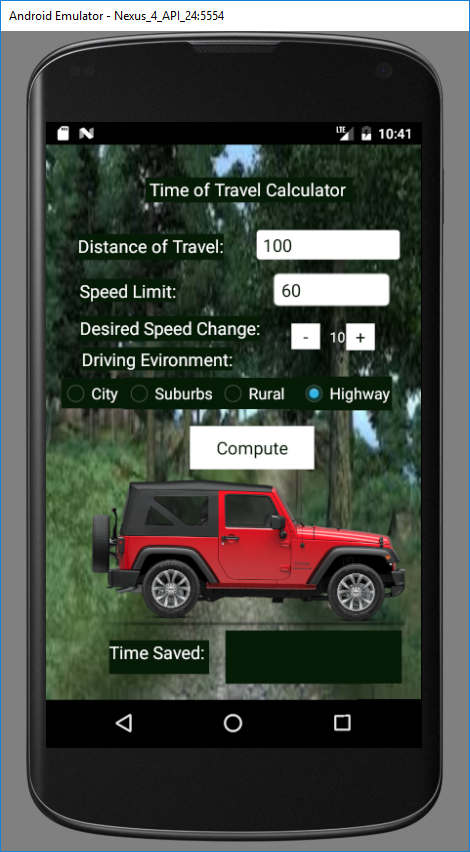
The figure below shows the android application being simulated and shows the android application in the Idle state waiting for values to be entered.



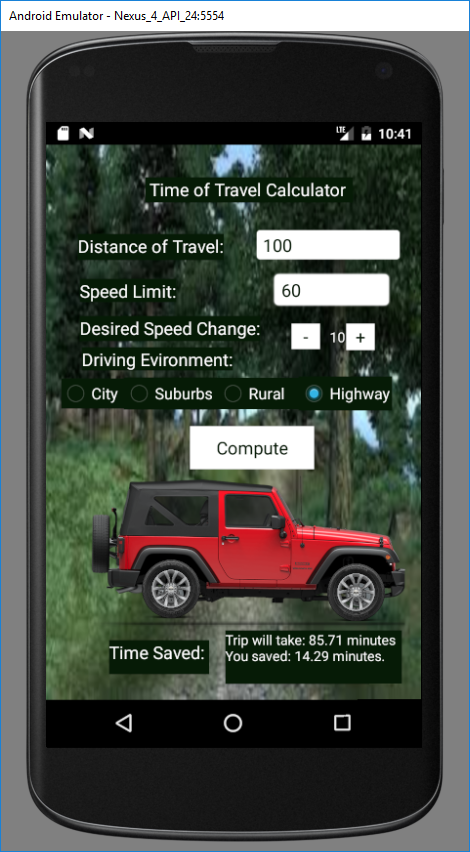
The figure below shows how the user enters data into the EditText boxes for the distance of travel and the speed limit. When the user presses the EditText view, the number keyboard will appear and allow the user to enter the values. For this instance, the user has entered that they will be traveling 100 miles at a rate of 60 miles per hour.



The figure below shows that the desired speed change can be changed by using the increment and decrement buttons and the driving environment can be selected using the radio buttons. For this instance, the user has indicated that they would like to travel at 10 miles per hour over the speed limit and intend to drive on the highway.



The figure below shows the results of entering values into all of the fields and then pressing the compute button. The results are shown at the bottom in a text field and indicates the total time that the trip will take, given the amount of speeding selected, and also the amount of time that was saved. Also, when the compute button is pressed the tires on the Jeep begin turning for 25 rotations. In this instance, the user will be traveling 100 miles at 70 miles per hour on the highway, which means that they will save 14 minutes and their total trip will take a little more than 85 minutes.



## Testing Results and Problems:

One of the most notable problems that we faced was with the radio buttons in the radio group. Our button listeners were not responding and the computed results would not change despite changing the environment. We found that we needed to create our own button listener that existed outside of the compute method in order to get the values stored for the driving environment to change when the selected radio button changed.