Requirements Verification Plan: Team CATVehicle

Requirement B.1: The Application should be able to establish a connection to a ROS publisher

Verification B.1: Run Test B.1.1 and Test B.1.2

Test B.1.1:

1. Connect the test phone and the ROS master machine to the same Wi-Fi network

2. Set up the ROS master using a separate machine and hardcode the IP address of the Wi-Fi network into the test code.

3. Execute the test program outlined below

Test B.1.1 pseudo-code:

```
//Send keystrokes for the hardcoded IP address

//Send an enter keystroke

//Begin a 1-minute countdown timer

//Check to see if ROSConnectActivity.isFinishing() returns true before the countdown
//timer elapses and if it does not the test fails
```

Test B.1.2:

1. Connect the test phone and the ROS master machine to the same Wi-Fi network

2. Set up the ROS master using a separate machine and hardcode an invalid IP address that differs from the Wi-Fi IP address into the test code.

3. Execute the test program outlined below

Test B.1.2 pseudo-code:

```
//Send keystrokes for the hardcoded IP address

//Send an enter keystroke

//Check to see if "ROS Connection Failed" appears on the screen.  If it does, the test
//passes

//If "ROS Connection Failed" does not appear on the screen, the test fails
```

Requirement B. 2: The Application should be able to receive the ROS message being published by the CATVehicle's ROS publisher

Verification B.2: Run Test B.2

Test B.2:

1. Connect the test phone and the ROS master machine to the same Wi-Fi network

2. Set up the ROS master using a separate machine and hardcode the IP address of the Wi-Fi network into the test code.

3. Play a pre-recorded rosbag file from our ROS master machine. This rosbag file contains a variety of ROS messages.

4. Execute the test program outlined below

Test B.2 pseudo-code:

Play the rosbag file. Have a counter in our call function that is called upon message receipt. After 5 seconds make sure this counter is a positive number.

```
//Send keystrokes for the hardcoded IP address

//Send an enter keystroke

//Begin a 5-second countdown timer

//Check to make sure that messageCount is greater than 0. If it is not, the test fails
```

Requirement B. 3: The Application should be able to correctly display a graphical representation of speedometer data being received from ROS master

Verification B.3: Run Test B.3

Test B.3:

1. Connect the test phone and the ROS master machine to the same Wi-Fi network

2. Set up the ROS master using a separate machine and hardcode the IP address of the Wi-Fi network into the test code.

3. Hardcode the speed values that are a part of the rosbag file to be played into the appropriate array in the test code

4. Play a pre-recorded rosbag file from our ROS master machine. This rosbag file contains a synchronization message followed by a series of speed data

5. Execute the test program outlined below

Test B.3 pseudo-code:

```
//Send keystrokes for the hardcoded IP address

//Send an enter keystroke

//Wait for the synchronization ROS message

//Compare speedometer.getSpeed() to the appropriate element in the hardcoded speed
//values vector. If they are not within 0.1 m/s of each other, the test fails
```

Requirement B.4: The Application will display an error message if the brake and the accelerator are actuated at the same time.

Verification B.4: Run Test B.4

Test B.4:

1. Connect the test phone and the ROS master machine to the same Wi-Fi network

2. Set up the ROS master using a separate machine and hardcode the IP address of the Wi-Fi network into the test code.

3. Play a pre-recorded rosbag file from our ROS master machine. This rosbag file contains an instance in which the brake and the accelerator are both greater than 0

4. Execute the test program outlined below

Test B.4 pseudo-code:

```
//Send keystrokes for the hardcoded IP address

//Send an enter keystroke

//Start a countdown timer for one minute

//errorFlag = 0
```

//Continuously check if the words "Error: Brake and Accelerator applied simultaneously" appear on the screen. If they do, errorFlag = 1

//When the countdown timer finishes, if errorFlag = 0 the test fails

Requirement B.5: The Application should be able to record and save the messages received during a session and should be able to email this data via the email client when prompted

Verification B.5: Run Test B.5.1 and Test B.5.2

Test B.5.1:

1. Connect the test phone and the ROS master machine to the same Wi-Fi network

2. Set up the ROS master using a separate machine and hardcode the IP address of the Wi-Fi network into the test code.

3. Play a pre-recorded rosbag file from our ROS master machine. This rosbag file contains various messages.

4. Execute the test program outlined below

Test B.5.1 pseudo-code:

//Create a service that can monitor the top activity

//Send keystrokes for the hardcoded IP address

//Send an enter keystroke

//Send a touch event to the record button

//Start a countdown timer for ten seconds

//Send a touch event to the stop recording button

//Send touch events to navigate to the overflow menu and then to the file menu

//Check to make sure that the fileListArray contains at least one entry. If not, the test //fails

Test B.5.2:

1. Connect the test phone and the ROS master machine to the same Wi-Fi network

2. Set up the ROS master using a separate machine and hardcode the IP address of the Wi-Fi network into the test code.

3. Play a pre-recorded rosbag file from our ROS master machine. This rosbag file contains various messages.

4. Execute the test program outlined below

Test B.5.2 pseudo-code:

```
//Create a service that can monitor the top activity

//Send keystrokes for the hardcoded IP address

//Send an enter keystroke

//Send a touch event to the record button

//Start a countdown timer for ten seconds

//Send a touch event to the stop recording button

//Send touch events to navigate to the overflow menu and then to the file menu

//Send a touch event to the share button for the file

//Search the log for topActivity and make sure it corresponds to the email client. If it
//does not, the test fails
```

Requirement A.1: The Application should be able to trigger a warning notification when it detects that the vehicle is exceeding the speed limit.

Verification A.1: Run Test A.1

Test A.1:

1. Connect the test phone and the ROS master machine to the same Wi-Fi network

2. Set up the ROS master using a separate machine and hardcode the IP address of the Wi-Fi network into the test code.

3. Play a pre-recorded rosbag file from our ROS master machine. This rosbag file contains speed data that exceeds the speed limit for a given street.

4. Execute the test program outlined below

Test A.1 pseudo-code:

```
//Send keystrokes for the hardcoded IP address

//Send an enter keystroke

//Start a one minute countdown timer

//If text similar to "Speed limit exceeded" are shown on the screen, the test passes and is
//exited

//If the timer finishes before the appearance of the text, the test fails
```

Requirement A.2: The Application should display an error message if the GPS data changes beyond the margin of error while the velocity is zero.

Verification A.2: Run Test A.2

Test A.2:

1. Connect the test phone and the ROS master machine to the same Wi-Fi network

2. Set up the ROS master using a separate machine and hardcode the IP address of the Wi-Fi network into the test code.

3. Play a pre-recorded rosbag file from our ROS master machine. This rosbag file contains GPS data that changes and speed data that has a value of 0

4. Execute the test program outlined below

Test A.2 pseudo-code:

```
//Send keystrokes for the hardcoded IP address

//Send an enter keystroke

//Start a one minute countdown timer

//If text similar to "GPS Error: GPS data is drifting while car is not moving" are shown
//on the screen, the test passes and is exited

//If the timer finishes before the appearance of the text, the test fails
```

Requirement A.3: The Application should be able to notify user if the LIDAR detects obstruction in front of the car

Verification A.3: Run Test A.3

Test A.3:

1. Connect the test phone and the ROS master machine to the same Wi-Fi network

2. Set up the ROS master using a separate machine and hardcode the IP address of the Wi-Fi network into the test code.

3. Play a pre-recorded rosbag file from our ROS master machine. This rosbag file contains LIDAR data that simulates an obstruction.

4. Execute the test program outlined below

Test A.3 pseudo-code:

```
//Send keystrokes for the hardcoded IP address

//Send an enter keystroke

//Start a one minute countdown timer

//If text similar to "Warning: Obstruction Detected" are shown on the screen, the test
//passes and is exited

//If the timer finishes before the appearance of the text, the test fails
```

Requirement A.4: The Application will compare phone GPS readings with ROS GPS readings to be within a specified preference of each other and will provide an alert if this is not the case

Verification A.4: Run Test A.3

Test A.4:

1. Connect the test phone and the ROS master machine to the same Wi-Fi network

2. Set up the ROS master using a separate machine and hardcode the IP address of the Wi-Fi network into the test code.

3. Determine the location that the test will be run and hardcode this GPS information into the rosbag file that will be played.

3. Play a pre-recorded rosbag file from our ROS master machine. This rosbag file contains GPS data that has been entered from the previous step.

4. Execute the test program outlined below

Test A.4 pseudo-code:

```
//Send keystrokes for the hardcoded IP address

//Send an enter keystroke

//Start a one minute countdown timer

//If text similar to " GPS Error: GPS of car and phone do not match" are shown on the
//screen, the test passes and is exited

//If the timer finishes before the appearance of the text, the test fails
```