**UI System Test Plan**

**For**

**EcoCAR**

Team members:

* Teja Tiriveedhi
* Nathan Rose
* Sabrina Yepez
* David Fadini
* Raymond (Sky) Kwasneski

|  |  |
| --- | --- |
| Version/Author | Date |
| 1/EcoCAR UI Team | 11/5/2020 |
| 2/EcoCAR UI Team | 11/23/2020 |
| 3/EcoCAR UI Team | 11/29/2020 |
|  |  |

Table of Contents

[1. Introduction 2](#_Toc530500519)

[1.1 Purpose 2](#_Toc530500520)

[1.2 Objectives 2](#_Toc530500521)

[2. Functional Scope 2](#_Toc530500522)

[3. Overall Strategy and Approach 2](#_Toc530500523)

[3.1 Testing Strategy 2](#_Toc530500524)

[3.2 UI System Testing Entrance Criteria 2](#_Toc530500525)

[3.3 Testing Types 2](#_Toc530500526)

[3.4 Suspension Criteria and Resumption Requirements 3](#_Toc530500527)

[4. Execution Plan 3](#_Toc530500528)

[4.1 Execution Plan 3](#_Toc530500529)

[5. Traceability Matrix & Defect Tracking 3](#_Toc530500530)

[5.1 Traceability Matrix 3](#_Toc530500531)

[5.2 Defect Severity Definitions 3](#_Toc530500532)

[6. Environment 4](#_Toc530500533)

[6.1 Environment 4](#_Toc530500534)

[7. Assumptions 4](#_Toc530500535)

[8. Risks and Contingencies 4](#_Toc530500536)

[9. Appendices 4](#_Toc530500537)

# Introduction

## Purpose

## This document is a test plan for the EcoCAR User Interface (UI) System Testing, produced by the EcoCAR UI Senior Design Team. It describes the testing strategy and approach the team will use to verify the application meets the requirements of the project prior to release.

## Objectives

* The driver shall have a video played for them the first time they try to use the Adaptive Cruise Control (ACC). The video content is a tutorial on how to use the ACC.
* The UI system shall gather Robot Operating System (ROS) data to show:
  + external obstacles near the EcoCAR. These obstacles include cars, trucks, and pedestrians.
  + the distance between the EcoCAR and an obstacle in front of the EcoCAR.
  + an alert to the driver of external obstacles that come within a desired proximity of the EcoCAR.

# Functional Scope

The functional scope for testing the EcoCAR UI system is solely focused on the UI display and information presented.

* The UI system shall update display information at an interval of no more than 100 milliseconds.
* The UI system shall read data from the ROS.

# Overall Strategy and Approach

## Testing Strategy

The EcoCAR UI system Testing will include testing of all functionalities that are in the scope (Refer Functional Scope Section). UI system testing activities include:

* Making sure the information on the display is readable to the driver.
* Making sure the information on the display updates within the desired interval of 100 milliseconds.
* Making sure the UI system alerts the driver when obstacles too close to the EcoCAR.

## UI System Testing Entrance Criteria

* Making sure there are no errors in the UI system when it is powered on.
* Making sure the UI system is stable for continuous use.
* Making sure the UI system can alert the driver.

## Testing Types

### Usability Testing

UI attributes, cosmetic presentation, and content will be tested for accuracy and general usability. The goal of usability testing is to ensure that the UI is easy to use and provides the driver with consistent access and navigation. Testing usability is done by simulating a driver using the UI system for its intended and unintended purposes.

### Functional Testing

The objective of the tests is to ensure that each element of the UI system meets the functional requirements of the project as outlined in the:

* Project/Functional Requirements.
* Project rules or conditions.
* Other functional documents produced during the project. (i.e., resolution to issues/change requests/feedback.)

## Suspension Criteria and Resumption Requirements

### Suspension Criteria

Testing will be suspended if the incidents found will not allow further testing of the UI system. If testing is halted and changes are made to the hardware, software, or database, it is up to the testing manager to decide whether the test plan will be re-executed.

### Resumption Requirements

Resumption of testing will be possible when a functionality that initially caused a suspension no longer causes that specific suspension.

# Execution Plan

## Execution Plan

The software for the UI system will be developed and tested in a Linux environment on the team’s local computers.

# Traceability Matrix & Defect Tracking

## Traceability Metrix

List of requirements (corresponding test cases)

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Test Case | Pass/Fail | Comments |
| The UI system shall become active when the software detects a signal for that specific task. | Have the UI system inactive before trying to make it active. |  | Inactive: The UI system is on, but not displaying anything.  Active: The UI system is on and displaying proper information. |
| The UI system shall become inactive when the software detects a signal for that specific task. | Have the UI system active before trying to make it inactive. |  | Inactive: The UI system is on, but not displaying anything.  Active: The UI system is on and displaying proper information. |
| The UI system shall update information to the display at an interval of no more than 100 milliseconds. | Make a call to update the UI screen at most once every 100 milliseconds. |  | The 100 milliseconds interval is a test figure for now. Later, we may try to reduce this interval. |
| The UI system shall display the relative position of external obstacles to the driver. | Give data to the UI that represents the relative position of external obstacles. |  | The obstacles include trucks, pedestrians and car. We will test the UI to check if we can display all the obstacles |
| The UI system shall display the lane type. | Give the UI a specified lane type. |  | They are different line types such as dashed, solid and double lane mark. We will test the UI to check if we can display all line types. |
| The UI system shall display lane location. | Give the UI a specified lane location. |  |  |
| The UI system shall display cruise control speed. | Give the UI the speed from the ROS. |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| The UI system shall provide a brief training video on how the ACC works to the driver. | Signal the UI to play the video. |  |  |
| The UI system shall get the driver’s attention by turning the distance text red when the EcoCAR detects an obstacle in front of it less than (V\* R) feet away.  Where:  V = speed of car (mi/hr.)  R = ratio (2hr) | Give data to the UI that represents an obstacle in front of it less than (V \* R) feet away. |  |  |
| The UI system shall display any issues that the on-board computer detects in the ACC system. | Display replicated defects such as the ACC is not functioning alert. |  |  |
| The UI system shall provide drivers a way to re-watch the training video. | Have the video play before trying to use a re-watch option. |  |  |
| The UI system will allow drivers to switch from a light mode to a dark mode. | The light/dark mode button on the UI display is pressed. |  |  |

## Defect Severity Definitions

|  |  |
| --- | --- |
| **Critical** | A critical defect is a defect that causes a catastrophic or severe error resulting in major problems that makes the UI system unusable to the driver. High effort is required to fix the defect. Examples of a critical defect are as follows:   * The UI system ends unexpectedly. * Data cannot flow through a project function/lifecycle. * Data is corrupted or cannot post to the database. |
| **Medium** | A medium defect is a defect that does not seriously harm UI system functions. Moderate effort is required to fix the defect. Examples of a medium defect are as follows:   * Navigation is incorrect. * Field labels are not consistent with global terminology. |
| **Low** | The low defect is a defect that has little to no impact on UI system functionality. Little effort is required to fix the defect. Examples of a low defect are as follows:   * Repositioning of fields on screens. * The font is incorrect. |

# Environment

## Environment

* A prototype of the UI system is currently being tested in a Linux environment. Later, it will be implemented using a Raspberry Pi in the EcoCAR. All physical environment requirements and conditions will be tested at a later stage once the software is integrated with the hardware.

# Assumptions

* The test plans will be tested independently on a platform by the designer, it will not be tested on the hardware. The EcoCAR has yet to be interfaced with the UI.
* The driver does not know how the UI nor the EcoCAR works in terms of functionally.
* Drivers do not own the EcoCAR.

# Risks and Contingencies

One risk is that the interface between the driver and the EcoCAR will change because there is a separate team working on generating data for the UI team (the team working on this document). This could lead to a conflict in data transmission. This has been accounted for by making the UI system modular and easy to change based on the data that the EcoCAR is giving us.

# Appendices

# 