SOFTWARE TEST DOCUMENT OF COMRADE PROJECT

Prepared by;
SIGHT EFFECT

Burak GOYNUK	M. Gorkem GENC	Mustafa AKILLI	Tayfun ATES
1819374	1819366	1818897	1818970

Table of Contents

1. Introduction

Comrade is the main android application that helps especially blind people to go some place from desired place. It contains several main functionalities that to be tested. As mentioned above, these functionalities are “navigation”, “OpenCV”, “Text-To-Speech” and “Speech-To-Text”. This application works on all advanced Android Platform and people can use this application with voice control. In this project, this STD document was prepared to test all of functionalities that were completed. Moreover, because of this “Software Test Document”, the quality of Comrade Project had been evaluated because all of functionalities work correctly for test cases.

1.1 Document Identifier

“Sight-Effect” team members prepared this document for testing all functionalities and properties of “Comrade Android Project”. This Software Test Document is the first document for explaining all of test cases which done to “Comrade Android Project” so the version of the document is “Version 1.0”. In addition to this, “IEEE Std 829-2008 Standard” was the reference of this Software Test Document. Notice that this document includes all of test cases and outcomes of functionalities such as “Navigation”, “OpenCV”, “Speech-To-Text” and “Text-To-Speech”.

1.2 Scope

This document was prepared to test all of functionalities and outcomes of “Comrade Android Project”. Moreover, the main purpose of the Software Test Document is evaluating the test cases that was done, and deciding whether outcomes are reliable (true) or not. In this system, there are two types of user, which are normal people, and visually impaired people so there are two types of interfaces. (The only interface that system is controlled voice and normal user interface). Because of that reasons, these interfaces and their functionalities are tested. Moreover, the software will be tested and evaluated using guidance. In addition to this information, inspection, analysis, demonstration and validation will be considered while test phase will be completed.

1.3 References

- “Comrade Android Project” SRS
- “Comrade Android Project” SDD
- IEEE 829-2008 Standard

1.4 Level in the overall sequence

There are possible level testing that can be used to test “Comrade Android Project” such as “integration testing”, “iteration testing” and “full-pass testing”. For this system, this STD document includes “full-pass testing” and detail information of the test of this test level. Moreover, the related document also include iteration that for the functionalities and new features.

1.5 Test classes and overall test conditions

Test classes are defined to functional requirements of “Comrade Android Project” according to Software Requirements Specification. In addition to this, also many test cases are explained detail in the next sections.

2. Details for system test plan

The following sections occurs some parts and the detail information about all of related parts are found below.

- Tests items and their identifiers
- Test traceability matrix
- Features to be tested
- Features not to be tested
- Approach
- Item pass/fail criteria
- Test deliverables

2.1 Test Items and Their Identifiers

The test items that are used in this document are identified in “Software Requirements Specification” document of “Comrade Android Project” so all of test items are found in this SRS document. “Comrade Android Project” is the project that uses the properties (APIs) of Google, OpencV APIs and Text-To-Speech and Speech-To-Text APIs. Moreover this application is based application that is developed in Android Platform. Because of this reason, these test cases were only done in Android Platform with visual test. This means that any test case does not try to black-box method. All of these methods are listed below.

- case 1 – Speech-to-Text Initialization
- case 2 – Speech-to-Text with Normal Speech Speed
- case 3 – Speech-to-Text with Speaking Fast
- case 4 – Speech-to-Text with Speaking Slow
- case 5 – Speech-to-Text with Words which are Hard to Understand
- case 6 – Speech-to-Text with Different Languages
- case 7 – Text-to-Speech with Normal Usage
- case 8 – Text-to-Speech with Words which are Hard to Pronounce
- case 9 – Text-to-Speech with Different Speech Speed
- case 10 – Text-to-Speech with Different Language
- case 11 – Text-to-Speech & Speech-to-Text Integration
- case 12 – String-To-LatLng
- case 13 – LatLng-To-String
- case 14 – Get-position
- case 15 – Get-path
- case 16 – Get-directions
- case 17 – Taxi-defraud

- case 18 – Training XML generation
- case 19 – Virtual Positive Sample Detection
- case 20 – Virtual Negative Sample Avoidance
- case 21 – Positive Sample Detection in Real Life
- case 22 – Negative Sample Avoidance in Real Life
- case 23 – Continuity of the Yellow Lines in Real Life
- case 24 – Obstacle, Gap Detection in Real Life

Notice that because of not to any other different platform is included, the only test item is the "Comrade Android Project" itself and all of the test cases that were listed below are explained detail in next sections.

2.2 Test traceability matrix

The test traceability matrix, which is shown below, includes two main guidelines that are use cases and test cases. As shown in figure, Use cases are indicated UC and their numbers; and test cases are indicated TC and their numbers. Notice that because of the size problem of traceability matrix, rows are described as test cases (TC) and columns are described as use cases (UC). Notice that there is only numbers of TC and UC in the table because of size problem.

	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC13
TC1	✓												
TC2	✓	✓							✓				
TC3	✓	✓							✓				
TC4	✓	✓							✓				
TC5	✓	✓							✓				
TC6	✓	✓											
TC7	✓	✓											
TC8	✓	✓											
TC9	✓	✓											
TC10	✓	✓											
TC11	✓	✓						✓		✓			
TC12		✓	✓										
TC13		✓	✓										
TC14			✓										
TC15			✓										
TC16			✓	✓									
TC17		✓				✓							
TC18												✓	✓
TC19												✓	✓
TC20												✓	✓
TC21												✓	✓

TC22												✓	✓
TC23												✓	✓
TC24												✓	✓

Notice that, because of such use cases that are explained section 2.4 did not developed; these use cases did not tested.

2.3 Features to be tested

In this document, all of the features to be tested were listed section 2.1. If more information about features that were tested is given, “text-to-speech”, “speech-to-text”, related properties of navigation part and related properties of “openCV” part were tested and evaluated in this document. Notice that all of the features, mentioned above, were tested with user interface. In addition to this, all of the information about features to be tested was found in “Software Requirements Specification” document of “Comrade Android Project”.

2.4 Features not to be tested

“Comrade Android Project” is not a project that literally finished, so some properties and features cannot be tested in this document. The use cases were tested in this system with user interface so not completed functionalities and nonfunctional requirements do not tested. Notice that all of features that are not finished are listed below.

- Start Application in the strict sense
- Bus Mode
- Get Status
- Integration of Object detection component with application

2.5 Approach

The approach that is applied in this document for testing features can be described as manual testing. This means that there is no any test method that can be associated with black-box method. For completing all of test case and evaluating whether their outcomes are reliable or not, manual tools are used. This means that if system gives continuously the same results, then test was true. Otherwise it was false. The main approach of this document is trying the same things in different “Android version in different telephone” to take same results.

2.6 Item pass/fail criteria

As we mentioned above, the main criteria of the project is whether the system gives same results or not in different “Android version” and in different telephone. If outcomes that are taken under these conditions are same, then item pass. Otherwise items fail. Notice that fail test case examined and fixed before test start over again.

2.7 Test Deliverables

In this document, the test deliverables is the “Software Test Document”. This means that the test deliverable is this document itself. Notice that all of the contexts are listed below.

- This document – “Software Test Document”
- Test Plans
- Test Cases
- Level Test Case
- Reporting Correct actions
- Reporting Fail actions

3. Test Management

3.1 Planned activities and tasks / test progression

In our test progression, all of test cases are tested in terms of systems and alone. This means that all of test cases are tested alone and if this test case is reliable for all trials then related test case referred to reliable test. After this process completed successfully, another test approach are done. This is the most important part of test progression to “Comrade Android Project”. Another test approach is that referred use case is tested in all of system. This is the true approach for “Comrade Android Project” because whether this test case breaks other features or not is the important issue. If related use case works properly in all system and does not break any other features then it is referred as reliable test case. Reliable test case means that related use case passes the entire test. This situation are tried to all of functionalities more than once because we are sure that related use case literally works properly in different “Android Version” and with different “telephone models”. Notice that if tested test case becomes fail when trying alone or in all system, then appropriate error is tried to find and fix; and all steps of test progression start anew. If all of tested steps are completed and all of steps give successful report then task is referred as reliable. Notice that all of different task is tested before integration the system. Thus, the test progression applies double testing. One the double testing is unit testing which is explained above and another is system testing which testing tasks after implementation of each module of the system. In addition to this, this test progression provides us to maximum information about tested tasks with minimum effort.

3.2 Environment/infrastructure

As mentioned above, we used an Android Platform. Moreover, we used different “Android version” and different “telephone models”. This means that we make ensure that “comrade” application works properly in different infrastructure.

4. Test Case Details

This section will explain the detailed information about the test case for each function requirement. Each test case includes identifier, objective, inputs, outcomes, environmental needs, special procedural requirements and intercase dependencies.

4.1. Speech Component Test Cases

In this part, the test scenarios related with speech API will be explained. The Speech Component is the composition of two minor components which are Text to Speech and Speech to Text. This component is one the most essential components of the Comrade System since it provides a communication between end-users, who are visually impaired people. This component is expected to behave as interface of the Comrade System. The details of the tests about that component can be found in below in detail.

4.1.1 Speech to Text Initialization

Test Case ID	Speech.TC.1
Objective	The aim of this test case is to verify that a signal or command should be sent to user in order to inform that speech to text component is working.
Inputs	Action to starting the application.
Outcomes	A signal or voice should be served to user.
Environmental Needs	The mobile platform should be connected to Internet.
Special Procedural Requirements	The user should speak loud enough and mobile phone must catch this voice.
Intercase Dependencies	-

4.1.2 Speech to Text with Normal Speech Speed

Test Case ID	Speech.TC.2
Objective	The aim of this test case is to test the most realistic scenario for Speech to Text component, which is speaking with normal speed.
Inputs	User voice.
Outcomes	The list of matched text should be returned to user.
Environmental Needs	The mobile platform should be connected to Internet.
Special Procedural Requirements	The user should speak loud enough and mobile phone must catch this voice.
Intercase Dependencies	The initialization of the Speech to Text conversion should be informed to user.

4.1.3 Speech to Text with Speaking Fast

Test Case ID	Speech.TC.3
Objective	Since each users' speaking speed can be different, the test case Speech.TC.3 and Speech.TC.4 cases is designed to simulate fast and slow speaking speed of users. This test case is designed for fast speaking users.
Inputs	User voice with fast speed.
Outcomes	The list of matched text should be returned to user.
Environmental Needs	The mobile platform should be connected to Internet.
Special Procedural Requirements	The user should speak loud enough and mobile phone must catch this voice.
Intercase Dependencies	The initialization of the Speech to Text conversion should be informed to user.

4.1.4 Speech to Text with Speaking Slow

Test Case ID	Speech.TC.4
Objective	To verify that slow-speaking users' speech is also can be converted to text.
Inputs	User voice with slow speed.
Outcomes	The list of matched text should be returned to user.
Environmental Needs	The mobile platform should be connected to Internet.
Special Procedural Requirements	The user should speak loud enough and mobile phone must catch this voice.
Intercase Dependencies	The initialization of the Speech to Text conversion should be informed to user.

4.1.5 Speech to Text with Words which are hard to Understand

Test Case ID	Speech.TC.5
Objective	This test case is designed to convert words which are hard to understand in Turkish.
Inputs	User voice with speaking different words in pronunciation.
Outcomes	The list of matched text should be returned to user.
Environmental Needs	The mobile platform should be connected to Internet.
Special Procedural Requirements	The user should speak loud enough and mobile phone must catch this voice.
Intercase Dependencies	The initialization of the Speech to Text conversion should be informed to user.

4.1.6 Speech to Text with Different Languages

Test Case ID	Speech.TC.6
Objective	Since for future work, the application will be served in different languages, the Speech to Text component should be tested for both English and Turkish. The aim of this test case is to verify that this component supports multiple languages.
Inputs	User voice with speaking in different languages. These inputs should be separate.
Outcomes	The list of matched text should be returned to user.
Environmental Needs	The mobile platform should be connected to Internet.
Special Procedural Requirements	The user should speak loud enough and mobile phone must catch this voice.
Intercase Dependencies	The initialization of the Speech to Text conversion should be informed to user.

4.1.7 Text to Speech with Normal Usage

Test Case ID	Speech.TC.7
Objective	The basic functionality of the Text to Speech component will be tested in that test scenario. In that case, the function should speak the given text as parameter.
Inputs	The text which is wanted to speak out by mobile phone.
Outcomes	The speech, which is the word entered to function, should be heard from the mobile phone's loudspeaker.
Environmental Needs	The mobile platform should support a valid loudspeaker.
Special Procedural Requirements	The loud level of the mobile platform should be high enough to be heard.
Intercase Dependencies	-

4.1.8 Text to Speech with Words which are hard to Pronounce

Test Case ID	Speech.TC.8
Objective	The purpose of this test case is to test the Text to Speech component with some hard words to pronounce. Thanks to that case, the developers will be able to see the pronunciation of that component.
Inputs	The text which is hard to pronounce.
Outcomes	The speech, which is the word entered to function, should be heard from the mobile phone's loudspeaker in a clear pronunciation.
Environmental Needs	The mobile platform should support a valid loudspeaker.
Special Procedural Requirements	The loud level of the mobile platform should be high enough to be heard.
Intercase Dependencies	The basic functionalities of the Text to Speech component should be tested.

4.1.9 Text to Speech with Different Speech Speed

Test Case ID	Speech.TC.9
Objective	The aim of this test case is to pronounce the entered text in different speech speed so that it can configurable from the settings menu of the application.
Inputs	The text to be pronounced and the speech speed which is in the enumeration format.
Outcomes	In case of slow speed is entered, the text should be pronounced in slow speed, on the other hand, if the fast speed is chosen, the text should be pronounced in fast speed.
Environmental Needs	The mobile platform should support a valid loudspeaker.
Special Procedural Requirements	The loud level of the mobile platform should be high enough to be heard.

Intercase Dependencies	The basic functionalities of the Text to Speech component should be tested.

4.1.10 Text to Speech with Different Language

Test Case ID	Speech.TC.10
Objective	In order to support multi language in the Comrade System, the Text to Speech component should also be tested in different language. The aim of this test scenario is to test Text to Speech conversion for both English and Turkish.
Inputs	The text to be pronounced and the language selection which is in the enumeration format.
Outcomes	The text, which is the argument of the function should be pronounced according to selected language.
Environmental Needs	The mobile platform should support a valid loudspeaker.
Special Procedural Requirements	The loud level of the mobile platform should be high enough to be heard.
Intercase Dependencies	The basic functionalities of the Text to Speech component should be tested.

4.1.11 Text to Speech & Speech to Text Integration

Test Case ID	Speech.TC.11
Objective	Since both Text to Speech and Speech to Text components will be composed in order to have a full Speech Component, they should be integrated and the integration of these components should be tested. The aim of that test scenario is to realize that they are working in systematic way.
Inputs	The speech which is user's voice.
Outcomes	Firstly, the user voice should be converted to text and it should be printed on the screen of Android device (for testing Speech to Text). Then, the converted text should be pronounced (for testing Text to

	Speech). Hence, the outcome is to provide user speech which is the repetition of what he said.
Environmental Needs	The mobile platform should support a valid loudspeaker and the device should be connected to Internet.
Special Procedural Requirements	The loud level of the mobile platform should be high enough to be heard.
Intercase Dependencies	Both Text to Speech and Speech to Text components should be implemented and their basic functionalities should be tested.

4.2. Navigation Component Test Cases

In this part, the test scenarios related with google maps API will be detailed. The most functions used in navigation part are all tested. These test cases are all successfully verified. The details about each test case will be explained in this part. Namely they are StringToLatlng, LatLngToString, getPositon, getPath, getDirections, taxiDefraudation and integration with speech component.

4.2.1 String to LatLng

Test Case ID	Navigation.TC.1
Objective	The purpose of this test case is to verify whether given address string is converted to a valid latlng variable correctly or not.
Inputs	Given address (destination) in string format.
Outcomes	Google's latlng variable.
Environmental Needs	User input voice should be correctly converted to string variable.
Special Procedural Requirements	User should say the address loud enough and mobile phone must be catch this voice.
Intercase Dependencies	All SpeechToText test cases should be done before this string to latlng conversion is made.

4.2.2 Latlng to String

Test Case ID	Navigation.TC.2
Objective	The purpose of this test case is to verify whether given latlng variable is converted to corresponding address string variable correctly or not.
Inputs	LatLng of the user.
Outcomes	The address of the user in string format.
Environmental Needs	The user must have an internet connection.
Special Procedural Requirements	The position of the user should be correctly got by using Google API.
Intercase Dependencies	Test of getting user input should be done before.

4.2.3 Get Position

Test Case ID	Navigation.TC.3
Objective	The aim of this test case is verifying the position of the user that is stored in latlng variable.
Inputs	GPS information of user.
Outcomes	The position of the user in latlng format.
Environmental Needs	The user must have an internet connection.
Special Procedural Requirements	Google Android map API functions should be correctly return latlng.
Intercase Dependencies	-

4.2.4 Get Path

Test Case ID	Navigation.TC.4
Objective	The aim of this test case is to verify that found path between position of the user and target address.
Inputs	User location and target location in latlng format.
Outcomes	Google API's polygon variable.
Environmental Needs	The user must have an internet connection.
Special Procedural Requirements	Google Android map API functions should be correctly return polygons.
Intercase Dependencies	SpeechToText test cases and StringToLatLng test case should be verified before this component is tested.

4.2.5 Get Directions

Test Case ID	Navigation.TC.5
Objective	The aim of this test case is to check whether the instructions of directions are correct or not.
Inputs	User location and target location in latlng format.
Outcomes	The information about path (Estimated remaining time and km as well as instructions) in json format.
Environmental Needs	The user must have an internet connection.
Special Procedural Requirements	Google Android map API functions should be correctly return expected JSON document.
Intercase Dependencies	SpeechToText test cases and StringToLatLng test case should be verified before this component is tested.

4.2.6 Taxi Defrauded

Test Case ID	Navigation.TC.6
Objective	The aim of this test case is to check that the application can capable of catching whether the taxi driver is defrauding the user or not.
Inputs	Succesive user locations in latlng format.
Outcomes	The information about defraudation.
Environmental Needs	The user must have an internet connection.
Special Procedural Requirements	-
Intercase Dependencies	-

4.2.7 Integration of Speech and Navigation Component

Test Case ID	Navigation.TC.7
Objective	The aim of this test case is to verify that the application is correctly run after integration of Speech and Navigation part.
Inputs	Speech input
Outcomes	Google map
Environmental Needs	The user must have an internet connection.
Special Procedural Requirements	-
Intercase Dependencies	All test cases of Speech and Navigation component should be correctly verified.

4.3. Vision Component Test Cases

Vision component is based on training of positive and negative samples via OpenCV. After each training, the following tests must be held and verified. In case of a failure in the training, the data set and the training parameters must be changed and tests must be verified again. Before explaining the test cases it is important to note that the positive samples are the objects (yellow line) that we are looking for and the negative samples are the possible backgrounds. These samples are extracted from video frames. Related test cases are placed below.

4.3.1 Training XML generation

Test Case ID	Vision.TC.1
Objective	The aim of this test case is to verify that our linux machine is able to produce an XML file of training with different sizes of positive and negative samples.
Inputs	Positive-negative samples, Feature Type, Number of total samples, Memory Usage, Number of stages.
Outcomes	An XML file shall be created when different combination of inputs provided.
Environmental Needs	At least 1024 MB of free memory in the Linux machine.
Special Procedural Requirements	The ratio of number of positive samples to negative samples should be nearly 0.5 for trainer to halt successfully.
Intercase Dependencies	-

4.3.2 Virtual Positive Sample Detection

Test Case ID	Vision.TC.2
Objective	The aim of this test case is to verify the application is able to find yellow lines when provided the trained positive samples on the computer screen.
Inputs	Positive samples that are used in the training process.
Outcomes	One or more green rectangles should be detected by the application.
Environmental Needs	An android device with OpenCV Manager installed.
Special Procedural Requirements	Positive sample images should be full screen on the computer.
Intercase Dependencies	Vision.TC.1 - XML file generation must be tested whether it is created successfully or not.

4.3.3 Virtual Negative Sample Avoidance

Test Case ID	Vision.TC.3
Objective	The aim of this test case is to verify the application is able to avoid backgrounds on the computer screen when provided the negative samples used in the training.
Inputs	Negative samples that are used in the training process.
Outcomes	Application should not find anything.
Environmental Needs	An android device with OpenCV Manager installed.
Special Procedural Requirements	Negative sample images should be full screen on the computer.
Intercase Dependencies	Vision.TC.1 - XML file generation must be tested whether it is created successfully or not.

4.3.4 Positive Sample Detection in Real Life

Test Case ID	Vision.TC.4
Objective	The aim of this test case is to verify the application is able to find yellow lines when provided positive frames in the nature that are NOT part of the training.
Inputs	Positive video frames that are NOT used in the training process.
Outcomes	One or more green rectangles should be detected by the application.
Environmental Needs	An android device with OpenCV Manager installed, a path containing yellow lines for visually impaired people.
Special Procedural Requirements	The tester walks from the beginning to the end of the path with its camera directed to the yellow lines on the path.
Intercase Dependencies	Vision.TC.1, Vision.TC.2, Vision.TC.3 - XML file generation and all virtual tests must be held before.

4.3.5 Negative Sample Avoidance in Real Life

Test Case ID	Vision.TC.5
Objective	The aim of this test case is to verify the application is able to avoid any types of background in real life that is not part of the training. If false alarms occur, these background shall be added to the successive training.
Inputs	Negative video frames that are NOT used in the training process
Outcomes	Application should not detect any green rectangles.
Environmental Needs	An android device with OpenCV Manager installed. a path containing different set of backgrounds.
Special Procedural Requirements	Tester should try to find false alarms by providing different set of backgrounds that are nearly look like positive samples.
Intercase Dependencies	Vision.TC.1, Vision.TC.2, Vision.TC.3 - XML file generation and all virtual tests must be held before.

4.3.6 Continuity of Yellow Lines in Real Life

Test Case ID	Vision.TC.6
Objective	The aim of this test case is to verify the application is able to understand the continuity of the path by finding two or more overlapping rectangles. The application should continue to find these rectangles when provided a series of frames containing yellow lines.
Inputs	Series of positive video frames that are NOT used in the training process.
Outcomes	Two or more overlapping green rectangles should be detected by the application.
Environmental Needs	An android device with OpenCV Manager installed, a path containing yellow lines for visually impaired people.
Special Procedural Requirements	The tester walks from the beginning to the end of the path with its camera directed to the yellow lines on the path.
Intercase Dependencies	Vision.TC.4, Vision.TC.5 - Positive Sample Detection and Negative Sample Avoidance tests in real life must be held.

4.3.7 Obstacle, Gap Detection in Real Life

Test Case ID	Vision.TC.7
Objective	The aim of this test case is to verify the application is able to detect any types of discontinuity like obstacles or gaps on the yellow line by finding two or more NON-overlapping rectangles
Inputs	Yellow line frames containing obstacles or gaps.
Outcomes	Two or more non-overlapping green rectangles should be detected by the application.
Environmental Needs	An android device with OpenCV Manager installed, a path containing yellow lines for visually impaired people. There should be gaps or obstacles on these yellow lines in order to verify this test case.
Special Procedural Requirements	The tester should stand at the one end of the yellow line and should direct the device to take both ends of gap or the obstacle. These both ends must contain the the positive object that the application is

	looking for.
Intercase Dependencies	Vision.TC.4, Vision.TC.5 - Positive Sample Detection and Negative Sample Avoidance tests in real life must be held.

5. System Test Report Details

This section gives us to overview of the test results, outcomes and detailed information about tested tasks. In addition to this, this section includes some properties, which are listed below.

- Overview of the test results
- Detailed test results
- Rationale for decisions
- Conclusions and recommendations

5.1 Overview of the test results

In test phase of all tasks that identified previous sections, it is observed that all of test cases are almost works properly. This means that all of tested tasks are applied and integrated system properly so the result of the test phase is as we expected. On the other hand, it is not possible to say that “Comrade Android Project” literally works properly because there are some features, which mentioned section 2.4. If features that not tested are developed and tested properly then it is easily said that “Comrade Android Project” fully developed. On the other hand, if some cases that the system behaves unexpectedly are observed then, firstly, use case that has an error will be fixed and described test progression will be applied.

5.2 Detailed test results

Speech to Text Initialization	Passed
Speech to Text with Normal Speech Speed	Passed
Speech to Text with Speaking Fast	Passed
Speech to Text with Speaking Slow	Passed
Speech to Text with Words which are Hard to Understand	Passed
Speech to Text with Different Languages	Passed
Text to Speech with Normal Usage	Passed
Text to Speech with Words which are Hard to Pronounce	Passed
Text to Speech with Different Speech	Passed

Speed	
Text to Speech with Different Language	Passed
Text to Speech & Speech to Text Integration	Passed
String to LatLng	Passed
Latlng to String	Passed
Get Position	Passed
Get Path	Passed
Get Directions	Passed
Taxi Defrauded	Passed
Integration of Speech and Navigation Component	Passed
Training XML generation	Passed
Virtual Positive Sample Detection	Passed
Virtual Negative Sample Avoidance	Passed
Positive Sample Detection in Real Life	Passed
Negative Sample Avoidance in Real Life	Passed
Continuity of Yellow Lines in Real Life	Passed
Obstacle, Gap Detection in Real Life	Passed

5.3 Rationale for decisions

Rationale for decisions is the maximizing the sequence of tests with the minimum effort.

5.4 Conclusions and Recommendations

As a result, all of tasks and use cases are tested properly and tested tasks passed test progression. This means that it can be easily said that the steps up until now were developed properly. However, there may be some problems in such test phase that applied features not tested. After this process we should develop the remaining features carefully and all of these features should be tested in detail. Notice that all of these features should be tested with module test and integration test because of maintaining the integrity of the system.