San Jose State University

Computer Engineering Department

CMPE 287 - Software Quality Assurance and Testing

Mobile Application Testing with AI Features (Envision AI)

Project Team - 9

Submitted To: Dr. Jerry Zeyu Gao

Submitted By:

Student ID	Name
012503270	Bhumika Tiwari
014544673	Mitra Gunakara Nayak
014502397	Atharva Chaitanya Munshi
014428141	Noopur Mehta

Table of Contents

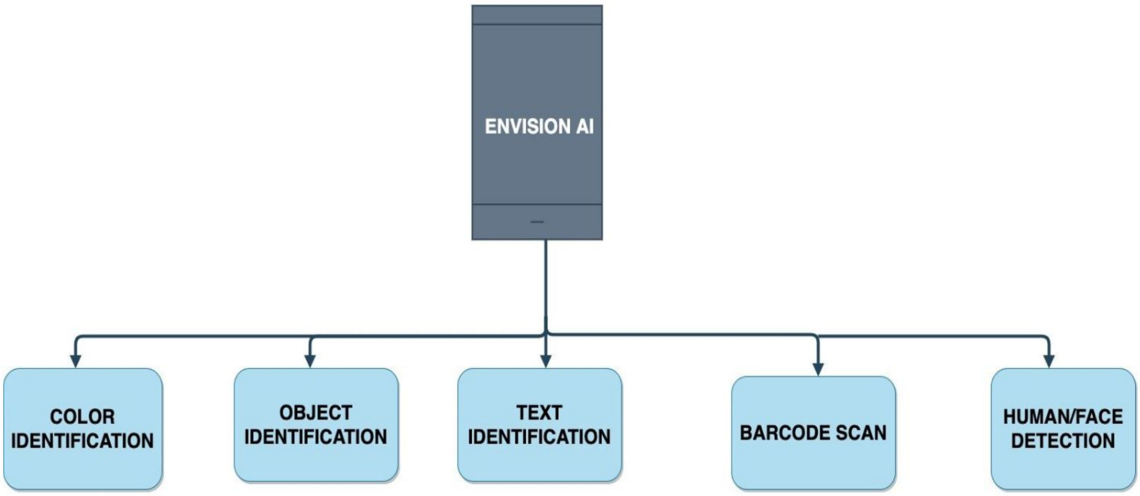
1. Introduction

1.1 Test Automation Focuses and Objectives

The purpose of this project is to perform AI testing for Envision mobile applications. We performed conventional testing for the app earlier and this document is intended to provide the details for automation testing for the AI functions of this app. Although conventional test execution did uncover some defects in the application under test, yet test automation was performed with the intent of uncovering more bugs.

Envision offers these AI functions:

- Color detection
- Object Identification
- Text Identification
- Barcode Scan
- Human/Face Detection



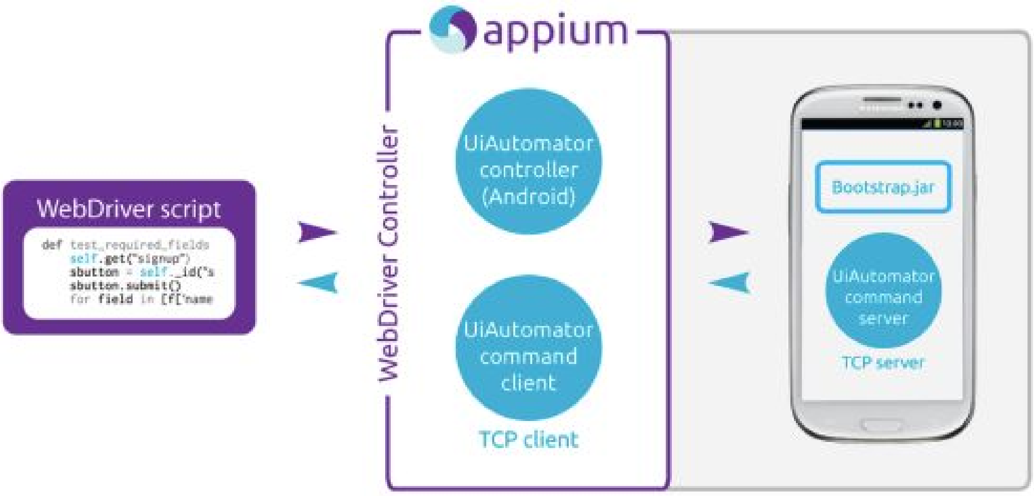
The main focus of Automated Testing is to provide test coverage of a cumulative nature, uncover defects, lower failure costs and facilitate repetitive tasks to run in an automated manner. This helps in keeping software cost to market under control and optimize usability of resources.

Automation testing focuses on below mentioned points:

1) Focus on engineers to save manual efforts for performing repetitive tasks.
2) Focus on accelerating the pace of work.
3) Focus on reducing cost.
4) Focus on achieving better quality software products.

By using automation in this project, we have got a good exposure to automation tools and technologies present in the software industry. It gave us an opportunity to evaluate the bunch of tools available and figure out the best suitable for testing Envision applications. Record and play feature of the Appium tool has helped us in saving time in performing manual runs. Not only did the Record and Play feature saved time, but also aided in generating reusable test scripts which can be utilized further to validate test scenarios and AI functionalities of the app.

1.2 Selected Tools



The tool that we have selected for test automation is **Appium**. Appium is an open-source tool. This is used to automate a given testing for mobile applications. Appium has the following features :

- Appium is **Free** !
- Appium can automate Native, Web and Hybrid mobile applications
- The generated tests from Appium can be executed on Real Devices, Simulators, and Emulators.
- Appium is cross-platform.
- Appium supports multiple programming languages.
- Appium gives the user an option to record a test and play the same
- Appium is compatible with majority of the test frameworks

Along with Appium, we have made use of the below softwares, jars and servers:

- Java
- ADB Driver
- Selenium Server
- TestNG
- IntelliJ IDE

2. AI Test Automation

AI test automation strategy

For testing Envision mobile application which offers multiple AI features, we have used classification based test strategy. As part of this strategy, we performed classification modeling for contexts, inputs and outputs. When input is fed to the app under test, events are triggered under a defined context environment to generate output. This test strategy helps in achieving maximum test coverage under diverse test conditions and inputs.

To leverage the advantages of classification based strategy, 3D Decision tables have been used in testing of this project. The three views of the decision cube -- *AI function context classification view, AI function input classification view, AI function output classification view* -- ensured better test coverage of Envision AI application.

We have used Spanning Trees extensively during test modeling for context, inputs, outputs and events. To test the project, we have used a real device testing approach wherein the mobile device with Envision app installed was connected to a computer on which Appium server was installed and running. We have written Java test scripts for testing the AI functions.

2.1 Human/object Detection

2.1.1 Test automation scenarios

Input Type	**Text**
Context	Light, Noise, input background, distance, Location, camera orientation, camera quality, action
Scenario 1	Human identification

S.No.	**Test Automation Test Case**	**Expected Result**	**Actual Result**	**Outcome**
2.1.1.1	Light - optimal Noise - no noise Input background- Plain distance - <1 feet Location- Indoor Camera Orientation- 45 degree Camera Quality- Clear Action- Male who is standing	The action mentioned and the gender (M) should be detected correctly	The App does not detect the gender correctly.	Fail

2.1.1.2	Light - No light Noise - Shaking/Moving Image Input background-Natural Scenario distance - 1 to 5 feet Location- Indoor Camera Orientation- Straight Camera Quality- Faulty Action- Male Who is sleeping	The action mentioned and the gender (M) should be detected correctly	The App detects the activity and gender correctly	Pass
2.1.1.3	Light - Artificial Light Noise - Distorted image Input background- Texture distance - 5 to 10 feet Location- Outdoor Camera Orientation- 90 degree Camera Quality- Sharp Action- Male who is walking	The action mentioned and the gender (M) should be detected correctly	The app is not able to identify the action and gender correctly	Fail
2.1.1.4	Light - Low Light Noise - Incomplete image of an object Input background- Natural Scenario distance - >10 feet Location- Indoor Camera Orientation- Straight Camera Quality-Sharp Action- Male who is playing	The action mentioned and the gender (M) should be detected correctly	The app is not able to identify the action and gender correctly	Fail
2.1.1.5	Light - Natural Sunlight Noise - Shaking/Moving Image Input background- Plain distance - 5 to 10 feet Location- Indoor Camera Orientation- 45 degree Camera Quality- Clear Action- Male who is reading	The app is not able to identify the action and gender correctly	The App is able to identify the action and the gender correctly	Fail
2.1.1.6	Light - No light Noise - Distorted image Input background- Texture distance - >10 feet Location- Outdoor	The app is not able to identify the action and gender correctly	The App does not recognise the action and the gender correctly	Fail

	Camera Orientation- 90 degree Camera Quality- clear Action- Male who is Walking			
2.1.1.7	Light - Low Light Noise - Shaking image Input background- Texture distance - 1 to 5 feet Location- Indoor Camera Orientation- Straight Camera Quality- sharp Action- Male who is standing	The app is not able to identify the action and gender correctly	The App is successfully able to detect the action and the gender	Pass
2.1.1.8	Light - Artificial Light Noise - Visibility Input background- Plain distance - >10 feet Location- Outdoor Camera Orientation- 90 degree Camera Quality- Faulty Action- Man who is sleeping	The app is not able to identify the action and gender correctly	The App is not able to detect the action and gender	Fail

Input Type	**Text**
Context	Objects, person, creature, activity, quality
Scenario 2	Object identification

S.No.	**Test Automation Test Case**	**Expected Result**	**Actual Result**	**Outcome**
2.1.1.1	Objects- Partial Object (Man made) Person- Male Creature- N/A Activity- Sitting in Chair Quality- Sharpness Action- Focus on the object	Envision AI should be able to identify the object correctly	The App does not detect the object correctly.	Fail
2.1.1.2	Objects- Complete object (Natural) Person- N/A Creature- Pet Activity- Standing Quality- Contrast	Envision AI should be able to identify the object correctly	The App does detect the object correctly.	Pass

2.1.1.3	Objects- Complete Object Person- Male Creature- N/A Activity- Using Laptop Quality- Clear	Envision AI should be able to identify the object correctly	Envision AI identifies the object correctly	Pass
2.1.1.4	Objects- Complete Object Person- N/A Creature- Wild Activity- Standing Quality- Sharpness	Envision AI should be able to identify the object correctly	Envision AI identifies the object correctly	Pass
2.1.1.5	Objects-Partial Object Person- Female Creature-N/A Activity-Looking in the mirror Quality-Contrast	Envision AI should be able to identify the object correctly	The App does not detect the object correctly.	Fail

2.2.2 Test scripts

Script 1

```
//Object detection in perfect lighting condition
import io.appium.java_client.MobileElement;
import io.appium.java_client.android.AndroidDriver;
import io.appium.java_client.android.AndroidElement;
import io.appium.java_client.remote.MobileCapabilityType;
import org.openqa.selenium.By;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;

import java.net.MalformedURLException;
import java.net.URL;

public class test1 {
   public static void main(String[] args) throws MalformedURLException {

      DesiredCapabilities dc = new DesiredCapabilities();

      dc.setCapability(MobileCapabilityType.DEVICE_NAME, "ZF6222B5JD");
      dc.setCapability("platformName", "android");
      dc.setCapability("appPackage", "com.letsenvision.envisionai");
      dc.setCapability("appActivity", ".login.SplashActivity");
      dc.setCapability("automationName", "UIAutomator1");
      dc.setCapability("noReset", true);

      AndroidDriver<AndroidElement> driver = new AndroidDriver<AndroidElement>(new
URL("http://127.0.0.1:4723/wd/hub"), dc);

        MobileElement el1 = (MobileElement)
driver.findElementByXPath("//androidx.appcompat.app.a.c[@content-desc=\"General\"]/an
droid.widget.TextView");
        el1.click();
        el1.click();
        MobileElement el2 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/iv_describe_scene_ic");
        el2.click();
        MobileElement el3 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/cl_image_feature_btns");
        el3.click();

        String targetResourceID = "com.letsenvision.envisionai:id/result_text_view";
        WebDriverWait wait = new WebDriverWait(driver, 20000);
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(targetResourceID)));
        String actualText = driver.findElementById(targetResourceID).getText();
        Assert.assertEquals(actualText,"Looks like a plastic container");
          }
}
```

<u>Script 2</u>

```
//Object detection with moving screen
import io.appium.java_client.MobileElement;
import io.appium.java_client.android.AndroidDriver;
import io.appium.java_client.android.AndroidElement;
import io.appium.java_client.remote.MobileCapabilityType;
import org.openqa.selenium.By;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;

import java.net.MalformedURLException;
import java.net.URL;

public class test2 {
    public static void main(String[] args) throws MalformedURLException {

        DesiredCapabilities dc = new DesiredCapabilities();
        dc.setCapability(MobileCapabilityType.DEVICE_NAME, "ZF6222B5JD");
        dc.setCapability("platformName", "android");
        dc.setCapability("appPackage", "com.letsenvision.envisionai");
        dc.setCapability("appActivity", ".login.SplashActivity");
        dc.setCapability("automationName", "UIAutomator1");
        dc.setCapability("noReset", true);
        AndroidDriver<AndroidElement> driver = new AndroidDriver<AndroidElement>(new
URL("http://127.0.0.1:4723/wd/hub"), dc);

        MobileElement el1 = (MobileElement)
driver.findElementByXPath("//androidx.appcompat.app.a.c[@content-desc=\"General\"]/android.widget.TextView");
        el1.click();
        el1.click();
        MobileElement el2 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/iv_describe_scene_ic");
        el2.click();
        MobileElement el3 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/cl_image_feature_btns");
        el3.click();

        String targetResourceID = "com.letsenvision.envisionai:id/result_text_view";
        WebDriverWait wait = new WebDriverWait(driver, 20000);
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(targetResourceID)));
        String actualText = driver.findElementById(targetResourceID).getText();
        Assert.assertEquals(actualText,"Looks like a laptop on desk");
    }
}
```

Script 3

```
Object detection with flash on
import io.appium.java_client.MobileElement;
import io.appium.java_client.android.AndroidDriver;
import io.appium.java_client.android.AndroidElement;
import io.appium.java_client.remote.MobileCapabilityType;
import org.openqa.selenium.By;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;
import java.net.MalformedURLException;
import java.net.URL;

public class test2 {
    public static void main(String[] args) throws MalformedURLException {

        DesiredCapabilities dc = new DesiredCapabilities();

        dc.setCapability(MobileCapabilityType.DEVICE_NAME, "ZF6222B5JD");
        dc.setCapability("platformName", "android");
        dc.setCapability("appPackage", "com.letsenvision.envisionai");
        dc.setCapability("appActivity", ".login.SplashActivity");
        dc.setCapability("automationName", "UIAutomator1");
        dc.setCapability("noReset", true);
        AndroidDriver<AndroidElement> driver = new AndroidDriver<AndroidElement>(new
URL("http://127.0.0.1:4723/wd/hub"), dc);

        MobileElement el1 = (MobileElement)
driver.findElementByXPath("//androidx.appcompat.app.a.c[@content-desc=\"General\"]/an
droid.widget.TextView");
        el1.click();
        el1.click();
        MobileElement el2 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/iv_describe_scene_ic");
        el2.click();
        MobileElement el3 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/cl_image_feature_btns");
        el3.click();

        String targetResourceID = "com.letsenvision.envisionai:id/result_text_view";
        WebDriverWait wait = new WebDriverWait(driver, 20000);
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(targetResourceID)));
        String actualText = driver.findElementById(targetResourceID).getText();
        Assert.assertEquals(actualText,"Looks like a bed");
    }
}
```

Script 4

```
//Face detection in perfect lighting condition
import io.appium.java_client.MobileElement;
import io.appium.java_client.android.AndroidDriver;
import io.appium.java_client.android.AndroidElement;
import io.appium.java_client.remote.MobileCapabilityType;
import org.openqa.selenium.By;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;

import java.net.MalformedURLException;
import java.net.URL;

public class test1 {
    public static void main(String[] args) throws MalformedURLException {

        DesiredCapabilities dc = new DesiredCapabilities();

        dc.setCapability(MobileCapabilityType.DEVICE_NAME, "ZF6222B5JD");
        dc.setCapability("platformName", "android");
        dc.setCapability("appPackage", "com.letsenvision.envisionai");
        dc.setCapability("appActivity", ".login.SplashActivity");
        dc.setCapability("automationName", "UIAutomator1");
        dc.setCapability("noReset", true);

        AndroidDriver<AndroidElement> driver = new AndroidDriver<AndroidElement>(new URL("http://127.0.0.1:4723/wd/hub"), dc);

        MobileElement el1 = (MobileElement) driver.findElementByXPath("//androidx.appcompat.app.a.c[@content-desc=\"General\"]/android.widget.TextView");
        el1.click();
        el1.click();
        MobileElement el2 = (MobileElement) driver.findElementById("com.letsenvision.envisionai:id/iv_describe_scene_ic");
        el2.click();
        MobileElement el3 = (MobileElement) driver.findElementById("com.letsenvision.envisionai:id/cl_image_feature_btns");
        el3.click();

        String targetResourceID = "com.letsenvision.envisionai:id/result_text_view";
        WebDriverWait wait = new WebDriverWait(driver, 20000);
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(targetResourceID)));
        String actualText = driver.findElementById(targetResourceID).getText();
        Assert.assertEquals(actualText,"Looks like a woman");
          }
}
```

Script 5

```
//Face detection with moving screen
import io.appium.java_client.MobileElement;
import io.appium.java_client.android.AndroidDriver;
import io.appium.java_client.android.AndroidElement;
import io.appium.java_client.remote.MobileCapabilityType;
import org.openqa.selenium.By;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;
import java.net.MalformedURLException;
import java.net.URL;

public class test2 {
    public static void main(String[] args) throws MalformedURLException {

        DesiredCapabilities dc = new DesiredCapabilities();
        dc.setCapability(MobileCapabilityType.DEVICE_NAME, "ZF6222B5JD");
        dc.setCapability("platformName", "android");
        dc.setCapability("appPackage", "com.letsenvision.envisionai");
        dc.setCapability("appActivity", ".login.SplashActivity");
        dc.setCapability("automationName", "UIAutomator1");
        dc.setCapability("noReset", true);

        AndroidDriver<AndroidElement> driver = new AndroidDriver<AndroidElement>(new
URL("http://127.0.0.1:4723/wd/hub"), dc);

        MobileElement el1 = (MobileElement)
driver.findElementByXPath("//androidx.appcompat.app.a.c[@content-desc=\"General\"]/an
droid.widget.TextView");
        el1.click();
        el1.click();
        MobileElement el2 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/iv_describe_scene_ic");
        el2.click();
        MobileElement el3 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/cl_image_feature_btns");
        el3.click();

        String targetResourceID = "com.letsenvision.envisionai:id/result_text_view";
        WebDriverWait wait = new WebDriverWait(driver, 20000);
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(targetResourceID)));
        String actualText = driver.findElementById(targetResourceID).getText();
        Assert.assertEquals(actualText,"Looks like a man in red shirt");
    }
}
```

Script 6

```
//Face detection with flash on
import io.appium.java_client.MobileElement;
import io.appium.java_client.android.AndroidDriver;
import io.appium.java_client.android.AndroidElement;
import io.appium.java_client.remote.MobileCapabilityType;
import org.openqa.selenium.By;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;
import java.net.MalformedURLException;
import java.net.URL;

public class test2 {
    public static void main(String[] args) throws MalformedURLException {

        DesiredCapabilities dc = new DesiredCapabilities();

        dc.setCapability(MobileCapabilityType.DEVICE_NAME, "ZF6222B5JD");
        dc.setCapability("platformName", "android");
        dc.setCapability("appPackage", "com.letsenvision.envisionai");
        dc.setCapability("appActivity", ".login.SplashActivity");
        dc.setCapability("automationName", "UIAutomator1");
        dc.setCapability("noReset", true);

        AndroidDriver<AndroidElement> driver = new AndroidDriver<AndroidElement>(new
URL("http://127.0.0.1:4723/wd/hub"), dc);

        MobileElement el1 = (MobileElement)
driver.findElementByXPath("//androidx.appcompat.app.a.c[@content-desc=\"General\"]/an
droid.widget.TextView");
        el1.click();
        el1.click();
        MobileElement el2 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/iv_describe_scene_ic");
        el2.click();
        MobileElement el3 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/cl_image_feature_btns");
        el3.click();

        String targetResourceID = "com.letsenvision.envisionai:id/result_text_view";
        WebDriverWait wait = new WebDriverWait(driver, 20000);
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(targetResourceID)));
        String actualText = driver.findElementById(targetResourceID).getText();
        Assert.assertEquals(actualText,"Looks like a group of people");
    }
}
```

2.2 Text Detection

2.2.1 Test automation scenarios

Input Type	**Text**
Context	Light, Camera angle and distance, Color of Text, Size of Text, Language of text
Scenario 1	Text present on the computer screen

S.No.	**Test Automation Test Case**	**Expected Result**	**Actual Result**	**Outcome**
2.2.1.1	Light - optimal Camera angle - optimal Distance - optimal Color of Text - black Size of Text - optimal	The App reads “I love Testing”	The App reads “I love testing “.	Fail

	Language of text- English			
2.2.1.2	Light - **Dark** Camera angle - optimal Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The App reads “I love Testing”	The App reads “No Text Found”	Fail
2.2.1.3	Light - **Too Bright** Camera angle - optimal Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The App reads “I love Testing”	The App reads “No Text Found”	Fail
2.2.1.4	Light - optimal Camera angle - **Slant** Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The App reads “I love Testing”	The App reads “I love testing “	Fail
2.2.1.5	Light - optimal Camera angle - optimal Distance - **Too close** Color of Text - black Size of Text - optimal Language of text- English	The App reads “I love Testing”	The App reads “I love ”	Fail
2.2.1.6	Light - optimal Camera angle - optimal Distance - **Too Far** Color of Text - black Size of Text - optimal Language of text- English	The App reads “I love Testing”	The App does not recognise	Fail
2.2.1.7	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Green** Size of Text - optimal Language of text- English	The App reads “I love Testing”	The App reads “I love testing “	Fail
2.2.1.8	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Red** Size of Text - optimal Language of text- **Hindi**	The App reads “मुझे परीक्षण पसंद है”	The App reads “मुझे परीक्षण पसंद है ”	Fail

2.2.1.9	Light - optimal Camera angle - optimal Distance - optimal Color of Text - Black Size of Text - **Too big** Language of text- **Hindi**	The App reads “मुझे परीक्षण पसंद है”	The App reads “मुझे ”	Fail
2.2.1.10	Light - optimal Camera angle - optimal Distance - optimal Color of Text - black Size of Text - **Too Small** Language of text- **Hindi**	The App reads “मुझे परीक्षण पसंद है”	The App reads “ՀԵՏ InՁՆԵ ԵԼՈՒ “	Fail
2.2.1.11	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Blue** Size of Text - optimal Language of text- **Numerical**	The App reads “12,345,678”	The App reads “2 , 345 , 678”	Fail

Input Type	**Text**
Context	Light, Camera angle and distance, Color of Text, Size of Text, Language of text
Scenario 2	Handwritten Text present on a sheet of paper

S.No.	**Test Automation Test Case**	**Expected Result**	**Actual Result**	**Outcome**
2.2.2.1	Light - optimal Camera angle - optimal Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The app should read “Software”.	The app reads “Software”.	Pass
2.2.2.2	Light - **Dark** Camera angle - optimal Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The app should read “Software”.	The app reads “No Text Found”.	Fail
2.2.2.3	Light - **Too Bright** Camera angle - optimal Distance - optimal Color of Text - black	The app should read “Software”.	The app reads “No Text Found”.	Fail

	Size of Text - optimal Language of text- English			
2.2.2.4	Light - optimal Camera angle - **Slant** Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The app should read “Software”.	The App reads “Software”.	Pass
2.2.2.5	Light - optimal Camera angle - optimal Distance - **Too close** Color of Text - black Size of Text - optimal Language of text- English	The app should read “Software”.	The App reads “Software”.	Pass
2.2.2.6	Light - optimal Camera angle - optimal Distance - **Too Far** Color of Text - black Size of Text - optimal Language of text- English	The app should read “Software”.	The app doesn’t detect the text correctly and shows some random text as output.	Fail
2.2.2.7	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Green** Size of Text - optimal Language of text- English	The app should read “Software”.	The app reads “Software”.	Pass
2.2.2.8	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Red** Size of Text - optimal Language of text- **Hindi**	The app should read “कार्यक्रम”.	The app reads “कार्यक्रम”.	Pass
2.2.2.9	Light - optimal Camera angle - optimal Distance - optimal Color of Text - Black Size of Text - **Too big** Language of text- English	The app should read “Software”.	The app reads “Software”.	Pass
2.2.2.10	Light - optimal Camera angle - optimal Distance - optimal Color of Text - black Size of Text - **Too Small**	The app should read “Software”.	The app reads “Software”.	Pass

	Language of text- English			
2.2.2.11	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Blue** Size of Text - optimal Language of text- **Numerical**	The app should read “2198765421”.	The app reads “2198765421”.	Pass

2.2.2 Test scripts

Script 1

```
public void textDetection1(AndroidDriver<AndroidElement> driver) {
  MobileElement el3 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/iv_document_scanner_ic");
   el3.click();
   try{
      Thread.sleep(10000);
   }catch (InterruptedException e){

   }
   MobileElement el4 = (MobileElement) driver.findElementByAccessibilityId("Take Photo");
   el4.click();
   String targetResourceID =
"/hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.Fram
eLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.view.ViewGrou
p/android.widget.FrameLayout[2]/android.view.ViewGroup/android.view.ViewGroup/androi
dx.recyclerview.widget.RecyclerView/android.widget.ScrollView/android.widget.LinearLay
out/android.widget.TextView";
  WebDriverWait wait = new WebDriverWait(driver, 20000);
   wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(targetResourceID)));
   String actualText = driver.findElementByXPath(targetResourceID).getText();
   Assert.assertEquals(actualText,"I Love Testing");
}

public static void main(String args[]) throws MalformedURLException {
   DesiredCapabilities dc = new DesiredCapabilities();
   dc.setCapability(MobileCapabilityType.DEVICE_NAME, "RF8MA01QL8Y");
   dc.setCapability("platformName", "android");
   dc.setCapability("appPackage", "com.letsenvision.envisionai");
   dc.setCapability("appActivity", ".MainActivity");
   dc.setCapability("automationName", "UIAutomator1");
   dc.setCapability("noReset", true);
   AndroidDriver<AndroidElement> driver = new AndroidDriver<AndroidElement>(new
URL("http://127.0.0.1:4723/wd/hub"), dc);
   TestAutomation ta = new TestAutomation();
java.awt.Desktop.getDesktop().browse(java.net.URI.create("https://docs.google.com/prese
ntation/d/1W1QMiHtMc5D7zN4J_OyJUZsYmrH5dAq_HK9tD0wBEAk/edit#slide=id.p"));
driver.launchApp();
ta.textDetection1(driver);
driver.closeApp();

}
```

Script 2

```
public void textDetection2(AndroidDriver<AndroidElement> driver) {
   MobileElement el3 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/iv_document_scanner_ic");
   el3.click();
   try{
       Thread.sleep(10000);
   }catch (InterruptedException e){

   }
  MobileElement el4 = (MobileElement) driver.findElementByAccessibilityId("Take Photo");
   el4.click();
   String targetResourceID =
"/hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.Fram
eLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.view.ViewGrou
p/android.widget.FrameLayout[2]/android.view.ViewGroup/android.view.ViewGroup/androi
dx.recyclerview.widget.RecyclerView/android.widget.ScrollView/android.widget.LinearLay
out/android.widget.TextView";
   WebDriverWait wait = new WebDriverWait(driver, 20000);
   wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(targetResourceID)));
   String actualText = driver.findElementByXPath(targetResourceID).getText();
   Assert.assertEquals(actualText,"मुझे परीक्षण पसंद है");
}
public static void main(String args[]) throws MalformedURLException {
   DesiredCapabilities dc = new DesiredCapabilities();
   dc.setCapability(MobileCapabilityType.DEVICE_NAME, "RF8MA01QL8Y");
   dc.setCapability("platformName", "android");
   dc.setCapability("appPackage", "com.letsenvision.envisionai");
   dc.setCapability("appActivity", ".MainActivity");
   dc.setCapability("automationName", "UIAutomator1");
   dc.setCapability("noReset", true);
   AndroidDriver<AndroidElement> driver = new AndroidDriver<AndroidElement>(new
URL("http://127.0.0.1:4723/wd/hub"), dc);
   TestAutomation ta = new TestAutomation();

java.awt.Desktop.getDesktop().browse(java.net.URI.create("https://docs.google.com/prese
ntation/d/1W1QMiHtMc5D7zN4J_OyJUZsYmrH5dAq_HK9tD0wBEAk/edit#slide=id.g8418
c0e8f2_0_16"));
driver.launchApp();
ta.textDetection2(driver);
driver.closeApp();

}
```

Script 3

```
public void textDetection3(AndroidDriver<AndroidElement> driver) {
   MobileElement el3 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/iv_document_scanner_ic");
   el3.click();
   try{
       Thread.sleep(10000);
   }catch (InterruptedException e){

  }
   MobileElement el4 = (MobileElement) driver.findElementByAccessibilityId("Take Photo");
   el4.click();
   String targetResourceID =
"/hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.view.ViewGroup/android.widget.FrameLayout[2]/android.view.ViewGroup/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.ScrollView/android.widget.LinearLayout/android.widget.TextView";
   WebDriverWait wait = new WebDriverWait(driver, 20000);
  wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(targetResourceID)));
   String actualText = driver.findElementByXPath(targetResourceID).getText();
   Assert.assertEquals(actualText,"12,345,678");
}
public static void main(String args[]) throws MalformedURLException {
   DesiredCapabilities dc = new DesiredCapabilities();
   dc.setCapability(MobileCapabilityType.DEVICE_NAME, "RF8MA01QL8Y");
   dc.setCapability("platformName", "android");
   dc.setCapability("appPackage", "com.letsenvision.envisionai");
   dc.setCapability("appActivity", ".MainActivity");
   dc.setCapability("automationName", "UIAutomator1");
   dc.setCapability("noReset", true);
   AndroidDriver<AndroidElement> driver = new AndroidDriver<AndroidElement>(new
URL("http://127.0.0.1:4723/wd/hub"), dc);
   TestAutomation ta = new TestAutomation();

java.awt.Desktop.getDesktop().browse(java.net.URI.create("https://docs.google.com/presentation/d/1W1QMiHtMc5D7zN4J_OyJUZsYmrH5dAq_HK9tD0wBEAk/edit#slide=id.g8418c0e8f2_0_21"));
driver.launchApp();
ta.textDetection3(driver);
driver.closeApp();

}
```

Script 4

```
public class textIdentification {

        public static void main(String[] args) throws MalformedURLException {

                DesiredCapabilities dc = new DesiredCapabilities();
                dc.setCapability(MobileCapabilityType.DEVICE_NAME, "97QAY11NMT");
                dc.setCapability("platformName", "android");
                dc.setCapability("appPackage", "com.letsenvision.envisionai");
                dc.setCapability("appActivity", ".MainActivity");
                dc.setCapability("noReset", "true");
                AndroidDriver<AndroidElement> ad = new
AndroidDriver<AndroidElement>(new URL("http://127.0.0.1:4723/wd/hub"), dc);
                textIdentification ti = new textIdentification();
                ti.textIdentificationtHandwrittenTestOne(ad);
}

        public void textIdentificationtHandwrittenTestOne(AndroidDriver<AndroidElement>
ad) {

                MobileElement el2 = (MobileElement)
ad.findElementById("com.letsenvision.envisionai:id/iv_document_scanner_ic");
                el2.click();
                try {
                        Thread.sleep(10000);
        } catch (InterruptedException e) {
                        e.printStackTrace();
                }
                MobileElement el7 = (MobileElement)
ad.findElementByAccessibilityId("Take Photo");
                el7.click();
                String targetResourceID =
"/hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.Fram
eLayout/"+"android.widget.LinearLayout/android.widget.FrameLayout/android.view.ViewG
roup/android.widget.FrameLayout[2]/"
+"android.view.ViewGroup/android.view.ViewGroup/androidx.recyclerview.widget.Recycle
rView/android.widget.ScrollView/" +
"android.widget.LinearLayout/android.widget.TextView";
                WebDriverWait wait = new WebDriverWait(ad, 20000);
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(targetResourceID)));
        String actualText = ad.findElementByXPath(targetResourceID).getText();
        Assert.assertEquals(actualText, "Software");
    }
}
```

<u>Script 5</u>

```
public class textIdentification {

public static void main(String[] args) throws MalformedURLException {

        DesiredCapabilities dc = new DesiredCapabilities();
        dc.setCapability(MobileCapabilityType.DEVICE_NAME, "97QAY11NMT");
        dc.setCapability("platformName", "android");
        dc.setCapability("appPackage", "com.letsenvision.envisionai");
        dc.setCapability("appActivity", ".MainActivity");
        dc.setCapability("noReset", "true");
        AndroidDriver<AndroidElement> ad = new
AndroidDriver<AndroidElement>(new URL("http://127.0.0.1:4723/wd/hub"), dc);
        textIdentification ti = new textIdentification();
        ti.textIdentificationtHandwrittenTestOne(ad);

    }

    public void textIdentificationtHandwrittenTestOne(AndroidDriver<AndroidElement>
ad) {

        MobileElement el2 =
(MobileElement)ad.findElementById("com.letsenvision.envisionai:id/iv_document_scanner
_ic");
        el2.click();
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        MobileElement el7 = (MobileElement)
ad.findElementByAccessibilityId("Take Photo");
        el7.click();
        String targetResourceID =
"/hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.Fram
eLayout/"+"android.widget.LinearLayout/android.widget.FrameLayout/android.view.ViewG
roup/android.widget.FrameLayout[2]/"
+"android.view.ViewGroup/android.view.ViewGroup/androidx.recyclerview.widget.Recycle
rView/android.widget.ScrollView/" +
"android.widget.LinearLayout/android.widget.TextView";
        WebDriverWait wait = new WebDriverWait(ad, 20000);

wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(targetResourceID)));
        String actualText = ad.findElementByXPath(targetResourceID).getText();
        Assert.assertEquals(actualText, "कार्यक्रम");
    }
}
```

Script 6

```
public class textIdentification {

    public static void main(String[] args) throws MalformedURLException {

        DesiredCapabilities dc = new DesiredCapabilities();
        dc.setCapability(MobileCapabilityType.DEVICE_NAME, "97QAY11NMT");
        dc.setCapability("platformName", "android");
    dc.setCapability("appPackage", "com.letsenvision.envisionai");
        dc.setCapability("appActivity", ".MainActivity");
        dc.setCapability("noReset", "true");
        AndroidDriver<AndroidElement> ad = new
AndroidDriver<AndroidElement>(new URL("http://127.0.0.1:4723/wd/hub"), dc);
    textIdentification ti = new textIdentification();
        ti.textIdentificationtHandwrittenTestOne(ad);

    }

public void textIdentificationtHandwrittenTestOne(AndroidDriver<AndroidElement> ad) {

        MobileElement el2 = (MobileElement)
ad.findElementById("com.letsenvision.envisionai:id/iv_document_scanner_ic");
        el2.click();
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        MobileElement el7 = (MobileElement)
ad.findElementByAccessibilityId("Take Photo");
        el7.click();
        String targetResourceID =
"/hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.Fram
eLayout/"+"android.widget.LinearLayout/android.widget.FrameLayout/android.view.ViewG
roup/android.widget.FrameLayout[2]/"
+"android.view.ViewGroup/android.view.ViewGroup/androidx.recyclerview.widget.Recycle
rView/android.widget.ScrollView/" +
"android.widget.LinearLayout/android.widget.TextView";
                    WebDriverWait wait = new WebDriverWait(ad, 20000);

wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(targetResourceID)));
                    String actualText = ad.findElementByXPath(targetResourceID).getText();
                    Assert.assertEquals(actualText, "2198765421");
          }
}
```

2.3 Color Detection

2.3.1 Test automation scenarios

Input Type	**Color**
Context	Light, Camera angle and distance
Scenario 1	Color present on the computer screen

S.No.	**Test Automation Test Case**	**Expected Result**	**Actual Result**	**Outcome**
2.3.1.1	Light - optimal Camera angle - optimal Distance - optimal Color - **Red**	Detects the color Red	The app detects the color Red.	Pass
2.3.1.2	Light - optimal Camera angle - optimal Distance - optimal Color - **Blue**	Detects the color Blue	The app detects the color Blue.	Pass
2.3.1.3	Light - optimal Camera angle - optimal Distance - optimal Color - **Yellow**	Detects the color Yellow	The app doesn’t detect the color correctly.	Fail
2.3.1.4	Light - optimal Camera angle - optimal Distance - optimal Color - **Purple**	Detects the color Purple	The app doesn’t detect the color correctly.	Fail

2.3.1.5	Light - optimal Camera angle - optimal Distance - optimal Color - **Orange**	Detects the color Orange	The app doesn't detect the color correctly.	Fail
2.3.1.6	Light - optimal Camera angle - optimal Distance - optimal Color - **Green**	Detects the color Green	The app detects the color Green.	Pass
2.3.1.7	Light - **Too Bright** Camera angle - optimal Distance - optimal Color - Red	Detects the color Red	The app doesn't detect the color correctly.	Fail
2.3.1.8	Light - **Dark** Camera angle - optimal Distance - optimal Color - Red	Detects the color Red	The app doesn't detect the color correctly.	Fail
2.3.1.9	Light - optimal Camera angle - **Slant** Distance - optimal Color - Red	Detects the color Red	The app doesn't detect the color correctly.	Fail
2.3.1.10	Light - Optimal Camera angle - optimal Distance - **Too Far** Color - Red	Detects the color Red	The app doesn't detect the color correctly.	Fail
2.3.1.11	Light - optimal Camera angle - **Slant** Distance - **Too close** Color - Blue	Detects the color Red	The app doesn't detect the color correctly.	Fail

Input Type	**Color**
Context	Light, Camera angle and distance
Scenario 2	Color present on an object

S.No.	**Test Automation Test Case**	**Expected Result**	**Actual Result**	**Outcome**
2.3.1.1	Light - optimal Camera angle - optimal Distance - optimal Color - **Red**	Detects the color Red	The app detects the color Red.	Pass
2.3.1.2	Light - optimal	Detects the color	The app detects	Pass

	Camera angle - optimal Distance - optimal Color - **Blue**	Blue	the color Blue.	
2.3.1.3	Light - optimal Camera angle - optimal Distance - optimal Color - **Yellow**	Detects the color Yellow	The app doesn’t detect the color correctly.	Fail
2.3.1.4	Light - optimal Camera angle - optimal Distance - optimal Color - **Purple**	Detects the color Purple	The app doesn’t detect the color correctly.	Fail
2.3.1.5	Light - optimal Camera angle - optimal Distance - optimal Color - **Orange**	Detects the color Orange	The app doesn’t detect the color correctly.	Fail
2.3.1.6	Light - optimal Camera angle - optimal Distance - optimal Color - **Green**	Detects the color Green	The app detects the color Green.	Pass
2.3.1.7	Light - **Too Bright** Camera angle - optimal Distance - optimal Color - Red	Detects the color Red	The app doesn’t detect the color correctly.	Fail
2.3.1.8	Light - **Dark** Camera angle - optimal Distance - optimal Color - Red	Detects the color Red	The app doesn’t detect the color correctly.	Fail
2.3.1.9	Light - optimal Camera angle - **Slant** Distance - optimal Color - Red	Detects the color Red	The app doesn’t detect the color correctly.	Fail
2.3.1.10	Light - Optimal Camera angle - optimal Distance - **Too Far** Color - Red	Detects the color Red	The app doesn’t detect the color correctly.	Fail
2.3.1.11	Light - optimal Camera angle - **Slant** Distance - **Too close** Color - Blue	Detects the color Red	The app doesn’t detect the color correctly.	Fail

2.3.2 Test scripts

Script 1

```
public void colorDetection1(AndroidDriver<AndroidElement> driver) {
   MobileElement el1 = (MobileElement)
driver.findElementByXPath("//androidx.appcompat.app.a.c[@content-desc=\"General\"]/android.widget.TextView");
   el1.click();
   try{
       Thread.sleep(1000);
   }catch (InterruptedException e){

   }
  MobileElement el2 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/iv_detect_color_ic");
   el2.click();
}
public static void main(String args[]) throws MalformedURLException {
   DesiredCapabilities dc = new DesiredCapabilities();
   dc.setCapability(MobileCapabilityType.DEVICE_NAME, "RF8MA01QL8Y");
   dc.setCapability("platformName", "android");
 dc.setCapability("appPackage", "com.letsenvision.envisionai");
   dc.setCapability("appActivity", ".MainActivity");
   dc.setCapability("automationName", "UIAutomator1");
   dc.setCapability("noReset", true);
   AndroidDriver<AndroidElement> driver = new AndroidDriver<AndroidElement>(new
URL("http://127.0.0.1:4723/wd/hub"), dc);
   TestAutomation ta = new TestAutomation();

java.awt.Desktop.getDesktop().browse(java.net.URI.create("https://docs.google.com/presentation/d/1W1QMiHtMc5D7zN4J_OyJUZsYmrH5dAq_HK9tD0wBEAk/edit#slide=id.g8418c0e8f2_0_29"));
driver.launchApp();
ta.colorDetection1(driver);



}
```

2.4 Barcode Detection

2.4.1 Test automation scenarios

Input Type	**Barcode**
Context	Light, Camera angle and distance
Scenario 1	Barcode present on a real physical object

S.No.	Test Automation TestCase	Expected Result	Actual Result	Outcome
2.4.1.1	Light - optimal Camera angle - optimal Distance - optimal	The app should read “Volini”.	The app reads “Volini”.	Pass
2.4.1.2	Light - **Too Bright** Camera angle - optimal Distance - optimal	The app should read “Volini”.	The app does not detect the barcode.	Fail
2.4.1.3	Light - **Dark** Camera angle - optimal Distance - optimal	The app should read “Volini”.	The app does not detect the barcode.	Fail
2.4.1.4	Light - Optimal Camera angle - Slant Distance - optimal	The app should read “Volini”.	The app reads “Volini”.	Pass
2.4.1.5	Light - Optimal Camera angle - optimal Distance - **Too Close**	The app should read “Volini”.	The app does not detect the barcode.	Fail
2.4.1.6	Light - Optimal Camera angle - optimal Distance - **Too Far**	The app should read “Volini”.	The app does not detect the barcode.	Fail

Input Type	Barcode
Context	Light, Camera angle and distance
Scenario 2	On-Screen Barcode

S.No.	Test Automation Test Case	Expected Result	Actual Result	Outcome
2.4.1.1	Light - optimal Camera angle - optimal Distance - optimal	The App reads “Tissue”	The App reads “Tissue”	Pass
2.4.1.2	Light - **Too Bright** Camera angle - optimal Distance - optimal	The App reads “Tissue”	The App does not detect	Fail
2.4.1.3	Light - **Dark** Camera angle - optimal Distance - optimal	The App reads “Tissue”	The App does not detect	Fail
2.4.1.4	Light - Optimal	The App reads	The App reads	Pass

	Camera angle - **Slant** Distance - optimal	“Tissue”	“Tissue”	
2.4.1.5	Light - Optimal Camera angle - optimal Distance - **Too Close**	The App reads “Tissue”	The App does not detect	Fail
2.4.1.6	Light - Optimal Camera angle - optimal Distance - **Too Far**	The App reads “Tissue”	The App does not detect	Fail

2.4.2 Test scripts

Script 1

```
public void barCodeDetection1(AndroidDriver<AndroidElement> driver) {
  MobileElement el1 = (MobileElement)
driver.findElementByXPath("//androidx.appcompat.app.a.c[@content-desc=\"General\"]/android.widget.TextView");
  el1.click();
  try{
     Thread.sleep(1000);
  }catch (InterruptedException e){

  }
 MobileElement el2 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/iv_scan_barcode_ic");
  el2.click();
  String targetResourceID = "com.letsenvision.envisionai:id/result_text_view";
  WebDriverWait wait = new WebDriverWait(driver, 20000);
  wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(targetResourceID)));
  String actualText = driver.findElementById(targetResourceID).getText();
  Assert.assertEquals(actualText,"Volini");
}
public static void main(String args[]) throws MalformedURLException {
  DesiredCapabilities dc = new DesiredCapabilities();
  dc.setCapability(MobileCapabilityType.DEVICE_NAME, "RF8MA01QL8Y");
  dc.setCapability("platformName", "android");
  dc.setCapability("appPackage", "com.letsenvision.envisionai");
  dc.setCapability("appActivity", ".MainActivity");
  dc.setCapability("automationName", "UIAutomator1");
  dc.setCapability("noReset", true);
  AndroidDriver<AndroidElement> driver = new AndroidDriver<AndroidElement>(new
URL("http://127.0.0.1:4723/wd/hub"), dc);
  TestAutomation ta = new TestAutomation();

java.awt.Desktop.getDesktop().browse(java.net.URI.create("https://docs.google.com/presentation/d/1W1QMiHtMc5D7zN4J_OyJUZsYmrH5dAq_HK9tD0wBEAk/edit#slide=id.g8418c0e8f2_0_11"));
driver.launchApp();
ta.barCodeDetection1(driver);
driver.closeApp();

}
```

Script 2

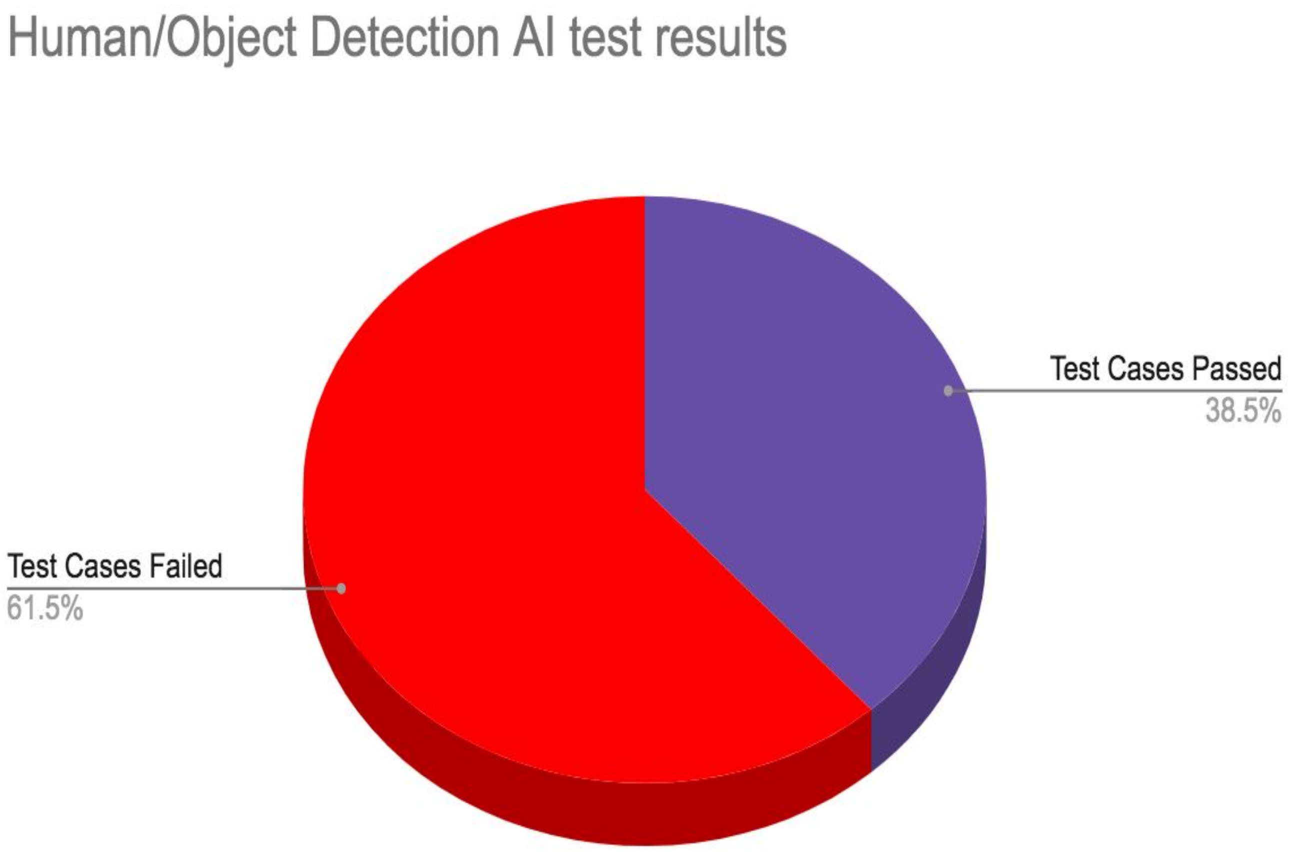
```
public void barCodeDetection1(AndroidDriver<AndroidElement> driver) {
 MobileElement el1 = (MobileElement)
driver.findElementByXPath("//androidx.appcompat.app.a.c[@content-desc=\"General\"]/an
droid.widget.TextView");
   el1.click();
   try{
       Thread.sleep(1000);
   }catch (InterruptedException e){

   }
   MobileElement el2 = (MobileElement)
driver.findElementById("com.letsenvision.envisionai:id/iv_scan_barcode_ic");
   el2.click();
   String targetResourceID = "com.letsenvision.envisionai:id/result_text_view";
   WebDriverWait wait = new WebDriverWait(driver, 20000);
   wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(targetResourceID)));
   String actualText = driver.findElementById(targetResourceID).getText();
   Assert.assertEquals(actualText,"Tissue");
}
public static void main(String args[]) throws MalformedURLException {
   DesiredCapabilities dc = new DesiredCapabilities();
   dc.setCapability(MobileCapabilityType.DEVICE_NAME, "RF8MA01QL8Y");
   dc.setCapability("platformName", "android");
   dc.setCapability("appPackage", "com.letsenvision.envisionai");
  dc.setCapability("appActivity", ".MainActivity");
   dc.setCapability("automationName", "UIAutomator1");
   dc.setCapability("noReset", true);
   AndroidDriver<AndroidElement> driver = new AndroidDriver<AndroidElement>(new
URL("http://127.0.0.1:4723/wd/hub"), dc);
   TestAutomation ta = new TestAutomation();
   ta.barCodeDetection1(driver);

}
```

3. AI Test Automation Comparative Results

3.1 AI Test Results in Statistical and graphical Format

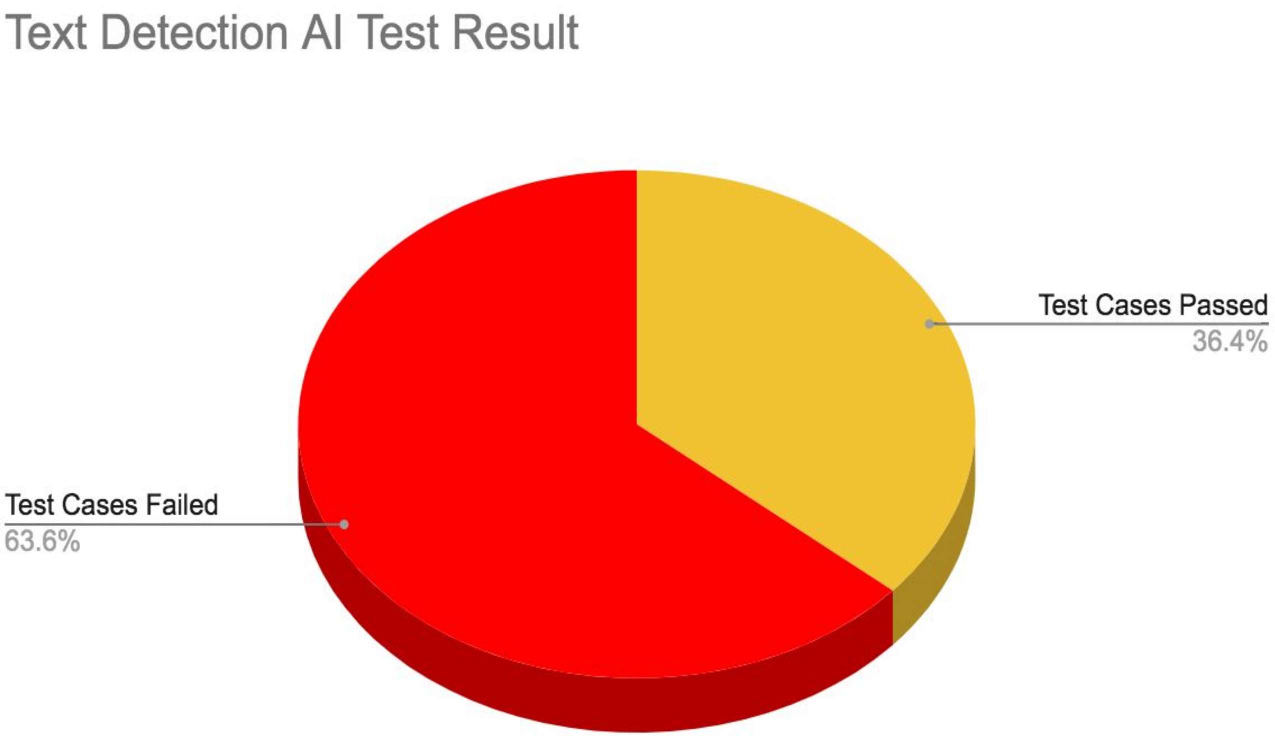
3.1.1 Graph for Human/Object Detection AI test results

Test Case Matrix	Total number of Test Cases	Test Cases Passed	Test Cases Failed
Count	13	5	8
Percentage	100%	38.5%	61.5%



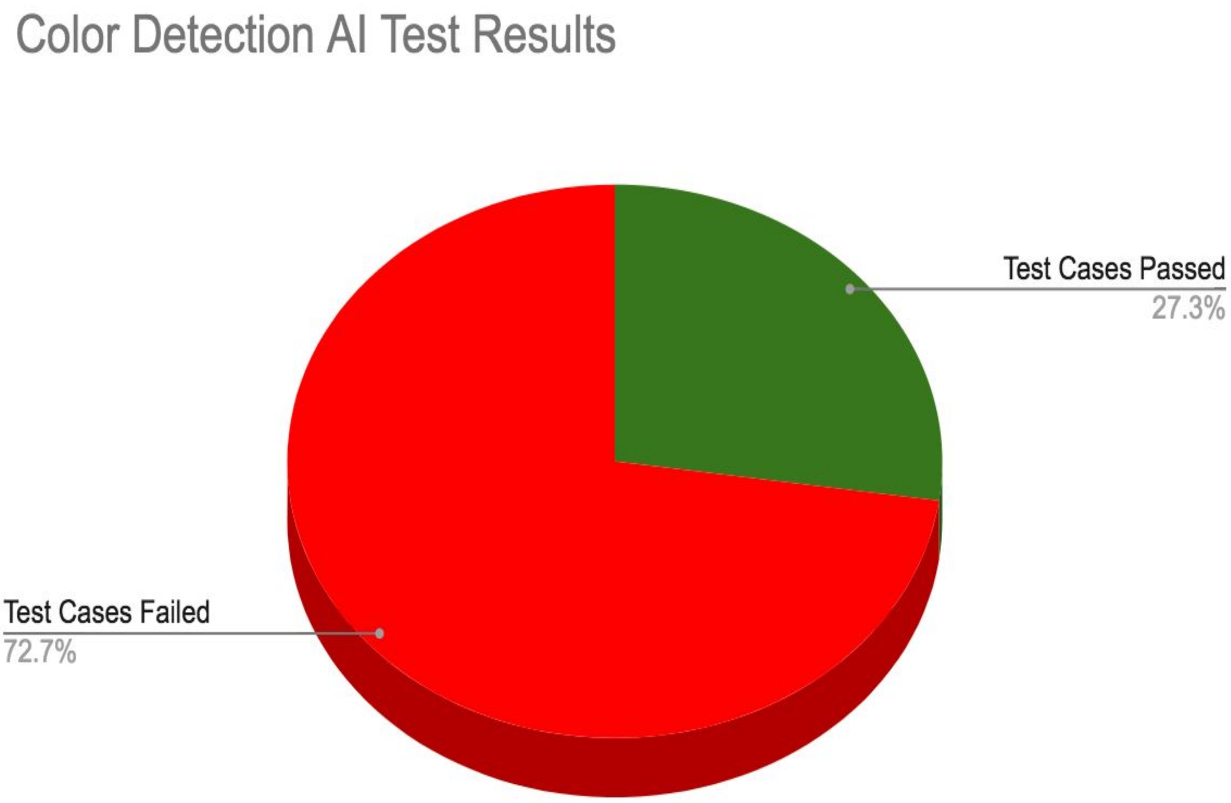
3.1.2 Graph for Text Detection AI test results

Test Case Matrix	Total number of Test Cases	Test Cases Passed	Test Cases Failed
Count	22	8	14
Percentage	100%	36.4%	63.6%



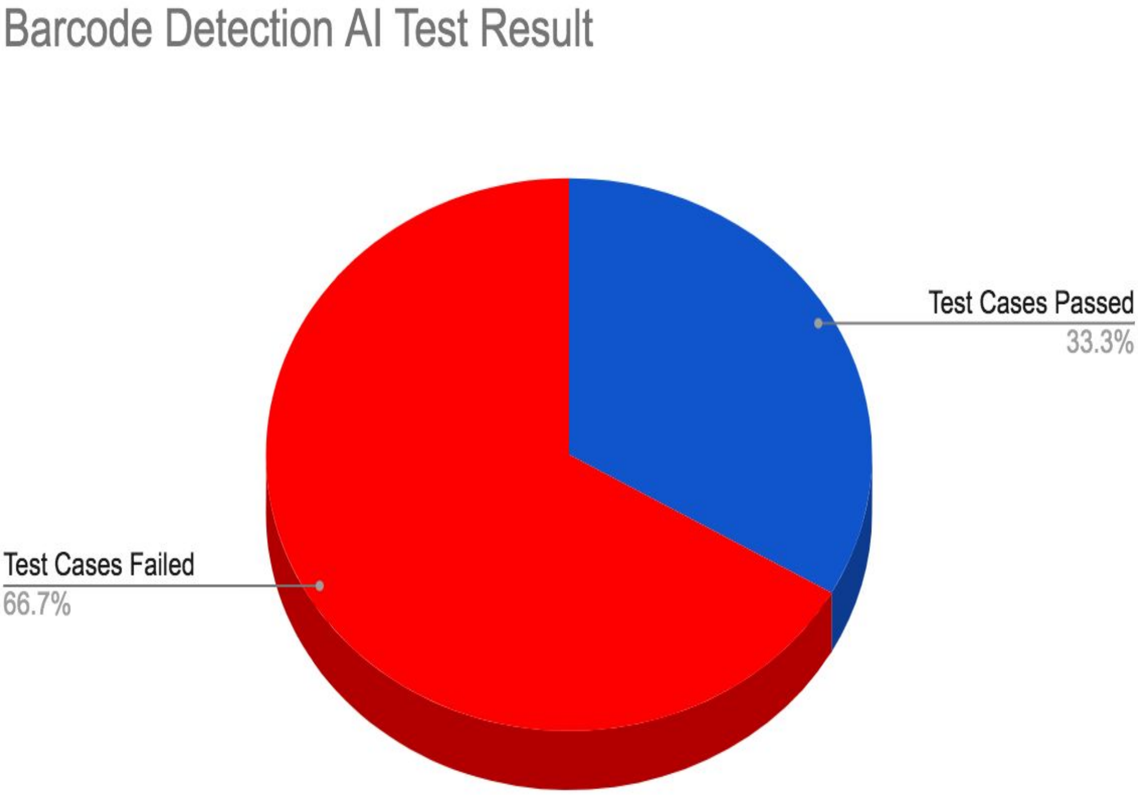
3.1.3 Graph for Color Detection AI Test Results

Test Case Matrix	Total number of Test Cases	Test Cases Passed	Test Cases Failed
Count	22	6	16
Percentage	100%	27.3%	72.7%



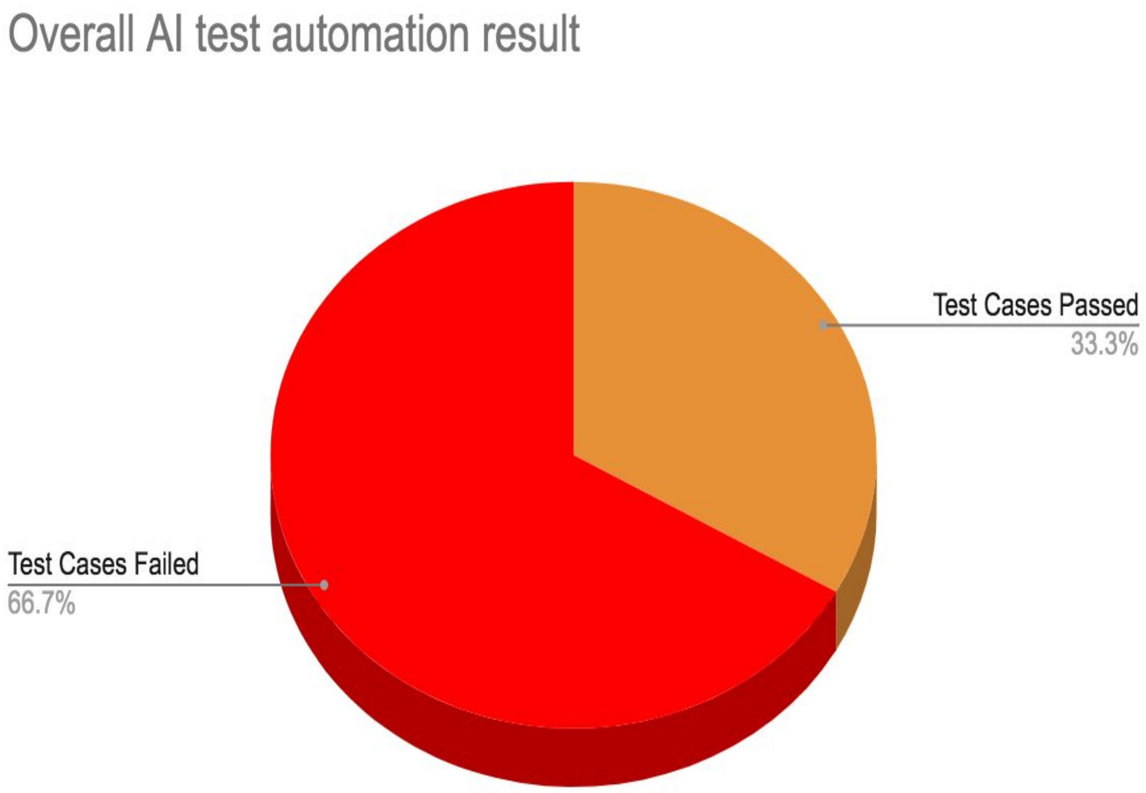
3.1.4 Graph for Barcode Detection AI Test Results

Test Case Matrix	**Total number of Test Cases**	**Test Cases Passed**	**Test Cases Failed**
Count	12	4	8
Percentage	100%	33.3%	66.7%



3.1.5 Graph for overall AI test automation result

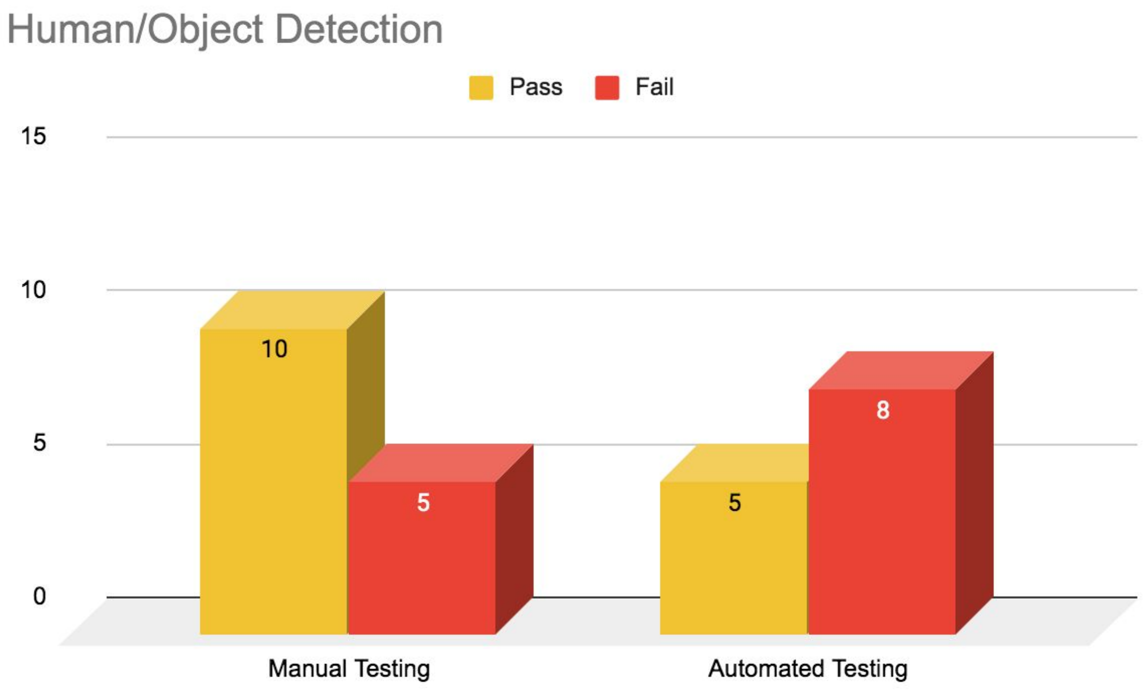
Test Case Matrix	Total number of Test Cases	Test Cases Passed	Test Cases Failed
Count	69	23	46
Percentage	100%	33.33%	66.66%



3.2 Comparative results between manual testing and automated testing

3.2.1 Human/Object Detection

Manual vs Automated	Total number of Test Cases	Pass	Fail
Manual Testing	13	10	5
Automated Testing	13	5	8



Input Type	Text
Context	Light, Noise, input background, distance, Location, camera orientation, camera quality, action, Objects, person, creature, activity, Quality
Scenario 1 and 2	Test case for human and Objects

S.No.	Test Automation Test Case	Expected Result	Manual Test execution result	Automated Test execution result

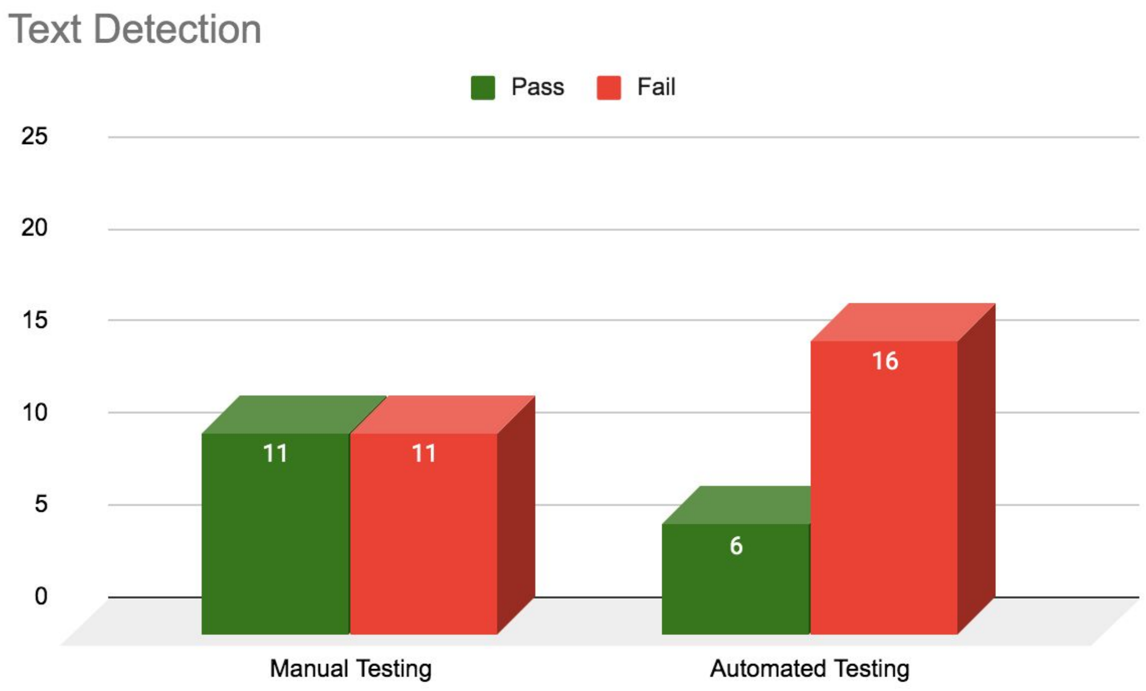
2.1.1.1	Light - optimal Noise - no noise Input background- Plain distance - <1 feet Location- Indoor Camera Orientation- 45 degree Camera Quality- Clear Action- Male who is standing	The action mentioned and the gender (M) should be detected correctly	Pass	Fail
2.1.1.2	Light - No light Noise - Shaking/Moving Image Input background-Natural Scenario distance - 1 to 5 feet Location- Indoor Camera Orientation- Straight Camera Quality- Faulty Action- Male Who is sleeping	The action mentioned and the gender (M) should be detected correctly	Fail	Fail
2.1.1.3	Light - Artificial Light Noise - Distorted image Input background- Texture distance - 5 to 10 feet Location- Outdoor Camera Orientation- 90 degree Camera Quality- Sharp Action- Male who is walking	The action mentioned and the gender (M) should be detected correctly	Pass	Fail
2.1.1.4	Light - Low Light Noise - Incomplete image of an object Input background- Natural Scenario distance - >10 feet Location- Indoor Camera Orientation- Straight Camera Quality-Sharp Action- Male who is playing	The action mentioned and the gender (M) should be detected correctly	Pass	Pass
2.1.1.5	Light - Natural Sunlight Noise - Shaking/Moving Image Input background- Plain distance - 5 to 10 feet Location- Indoor	The app is not able to identify the action and gender correctly	Pass	Fail

	Camera Orientation- 45 degree Camera Quality- Clear Action- Male who is reading			
2.1.1.6	Light - No light Noise - Distorted image Input background- Texture distance - >10 feet Location- Outdoor Camera Orientation- 90 degree Camera Quality- clear Action- Male who is Walking	The app is not able to identify the action and gender correctly	Fail	Fail
2.1.1.7	Light - Low Light Noise - Shaking image Input background- Texture distance - 1 to 5 feet Location- Indoor Camera Orientation- Straight Camera Quality- sharp Action- Male who is standing	The app is not able to identify the action and gender correctly	Pass	Fail
2.1.1.8	Light - Artificial Light Noise - Visibility Input background- Plain distance - >10 feet Location- Outdoor Camera Orientation- 90 degree Camera Quality- Faulty Action- Man who is sleeping	The app is not able to identify the action and gender correctly	Fail	Fail
2.1.1.9	Objects- Partial Object (Man made) Person- Male Creature- N/A Activity- Sitting in Chair Quality- Sharpness Action- Focus on the object	Envision AI should be able to identify the object correctly	Pass	Pass
2.1.1.10	Objects- Complete object (Natural) Person- N/A Creature- Pet Activity- Standing Quality- Contrast	Envision AI should be able to identify the object correctly	Pass	Pass
2.1.1.11	Objects- Complete Object	Envision AI should	Fail	Fail

	Person- Male Creature- N/A Activity- Using Laptop Quality- Clear	be able to identify the object correctly		
2.1.1.12	Objects- Complete Object Person- N/A Creature- Wild Activity- Standing Quality- Sharpness	Envision AI should be able to identify the object correctly	Fail	Pass
2.1.1.13	Objects-Partial Object Person- Female Creature-N/A Activity-Looking in the mirror Quality-Contrast	Envision AI should be able to identify the object correctly	Pass	Pass

3.2.2 Text Detection

Manual vs Automated	**Total number of Test Cases**	**Pass**	**Fail**
Manual Testing	22	11	11
Automated Testing	22	6	16



Input Type	**Text**
Context	Light, Camera angle and distance, Color of Text, Size of Text, Language of text
Scenario 1	Text present on the computer screen

S.No.	**Test Automation Test Case**	**Expected Result**	**Manual Test Execution Result**	**Automated Test Execution Result**
3.2.2.1.1	Light - optimal Camera angle - optimal Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The App reads “I Love Testing”	Pass	Fail
3.2.2.1.2	Light - **Dark** Camera angle - optimal Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The App reads “I Love Testing”	Fail	Fail
3.2.2.1.3	Light - **Too Bright** Camera angle - optimal Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The App reads “I Love Testing”	Fail	Fail
3.2.2.1.4	Light - optimal Camera angle - **Slant** Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The App reads “I Love Testing”	Pass	Fail
3.2.2.1.5	Light - optimal Camera angle - optimal Distance - **Too close** Color of Text - black Size of Text - optimal Language of text- English	The App reads “I Love Testing”	Fail	Fail
3.2.2.1.6	Light - optimal Camera angle - optimal Distance - **Too Far**	The App reads “I Love Testing”	Pass	Fail

	Color of Text - black Size of Text - optimal Language of text- English			
3.2.2.1.7	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Green** Size of Text - optimal Language of text- English	The App reads “I Love Testing”	Pass	Fail
3.2.2.1.8	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Red** Size of Text - optimal Language of text- **Hindi**	The app reads “मुझे कागज़ पर लिखना पसंद है”	Pass	Fail
3.2.2.1.9	Light - optimal Camera angle - optimal Distance - optimal Color of Text - Black Size of Text - **Too big** Language of text- Hindi	The app reads “मुझे कागज़ पर लिखना पसंद है”	Fail	Fail
3.2.2.1.10	Light - optimal Camera angle - optimal Distance - optimal Color of Text - black Size of Text - **Too Small** Language of text- Hindi	The app reads “मुझे कागज़ पर लिखना पसंद है”	Fail	Fail
3.2.2.1.11	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Blue** Size of Text - optimal Language of text- **Numerical**	The app reads “12,345,678”	Pass	Fail

Input Type	**Text**
Context	Light, Camera angle and distance, Color of Text, Size of Text, Language of text
Scenario 2	Handwritten Text present on a sheet of paper

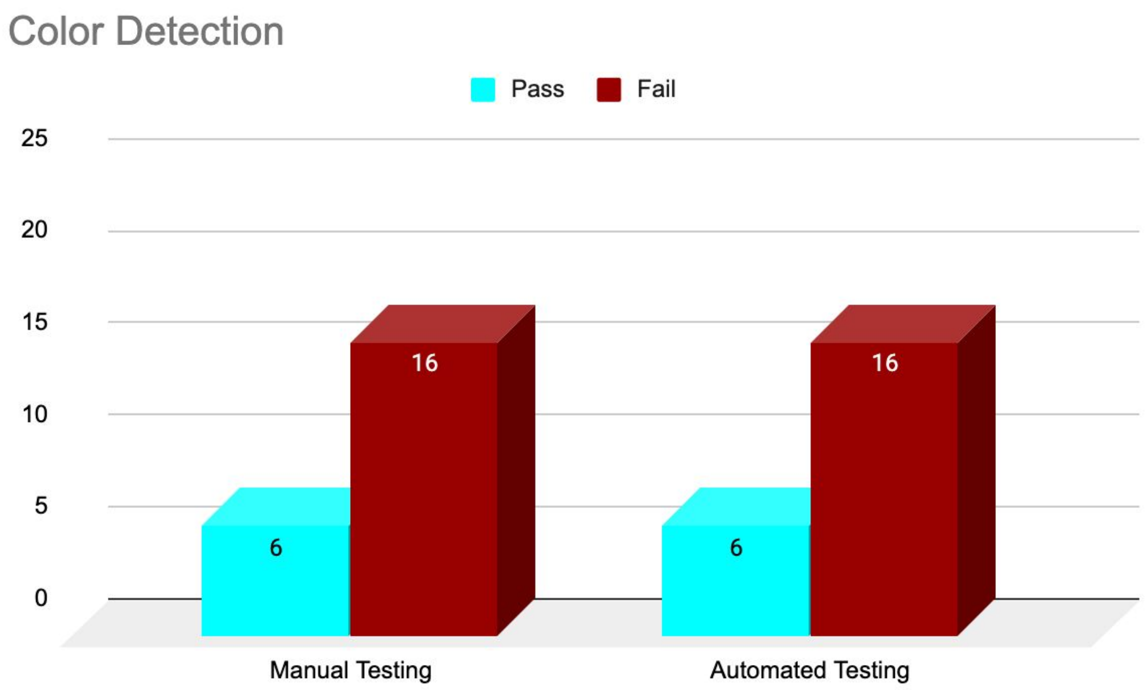
S.No.	**Test Automation Test Case**	**Expected Result**	**Manual Test Execution**	**Automated Test**

			Result	**Execution Result**
3.2.2.2.1	Light - optimal Camera angle - optimal Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The app should read “Software”.	Pass	Pass
3.2.2.2.2	Light - **Dark** Camera angle - optimal Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The app should read “Software”.	Fail	Fail
3.2.2.2.3	Light - **Too Bright** Camera angle - optimal Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The app should read “Software”.	Fail	Fail
3.2.2.2.4	Light - optimal Camera angle - **Slant** Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The app should read “Software”.	Pass	Pass
3.2.2.2.5	Light - optimal Camera angle - optimal Distance - **Too close** Color of Text - black Size of Text - optimal Language of text- English	The app should read “Software”.	Pass	Pass
3.2.2.2.6	Light - optimal Camera angle - optimal Distance - **Too Far** Color of Text - black Size of Text - optimal Language of text- English	The app should read “Software”.	Fail	Fail
3.2.2.2.7	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Green** Size of Text - optimal Language of text- English	The app should read “Software”.	Pass	Pass

3.2.2.2.8	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Red** Size of Text - optimal Language of text- **Hindi**	The app should read “कार्यक्रम”.	Pass	Pass
3.2.2.2.9	Light - optimal Camera angle - optimal Distance - optimal Color of Text - Black Size of Text - **Too big** Language of text- English	The app should read “Software”.	Pass	Pass
3.2.2.2.10	Light - optimal Camera angle - optimal Distance - optimal Color of Text - black Size of Text - **Too Small** Language of text- English	The app should read “Software”.	Pass	Pass
3.2.2.2.11	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Blue** Size of Text - optimal Language of text- **Numerical**	The app should read “2198765421”.	Pass	Pass

3.2.3 Color Detection

Manual vs Automated	Total number of Test Cases	Pass	Fail
Manual Testing	22	6	16
Automated Testing	22	6	16



Input Type	Color
Context	Light, Camera angle and distance
Scenario 1	Color present on the computer screen

S.No.	Test Automation TestCase	Expected Result	Manual Test Execution Result	Automated Test Execution Result
3.2.3.1.1	Light - optimal Camera angle - optimal Distance - optimal Color - **Red**	Detects the color Red	Pass	Pass
3.2.3.1.2	Light - optimal Camera angle - optimal Distance - optimal Color - **Blue**	Detects the color Blue	Pass	Pass
3.2.3.1.3	Light - optimal Camera angle - optimal Distance - optimal Color - **Yellow**	Detects the color Yellow	Fail	Fail
3.2.3.1.4	Light - optimal Camera angle - optimal	Detects the color Purple	Fail	Fail

	Distance - optimal Color - **Purple**			
3.2.3.1.5	Light - optimal Camera angle - optimal Distance - optimal Color - **Orange**	Detects the color Orange	Fail	Fail
3.2.3.1.6	Light - optimal Camera angle - optimal Distance - optimal Color - **Green**	Detects the color Green	Pass	Pass
3.2.3.1.7	Light - **Too Bright** Camera angle - optimal Distance - optimal Color - Red	Detects the color Red	Fail	Fail
3.2.3.1.8	Light - **Dark** Camera angle - optimal Distance - optimal Color - Red	Detects the color Red	Fail	Fail
3.2.3.1.9	Light - optimal Camera angle - **Slant** Distance - optimal Color - Red	Detects the color Red	Fail	Fail
3.2.3.1.10	Light - Optimal Camera angle - optimal Distance - **Too Far** Color - Red	Detects the color Red	Fail	Fail
3.2.3.1.11	Light - optimal Camera angle - **Slant** Distance - **Too close** Color - Red	Detects the color Red	Fail	Fail

Input Type	**Color**
Context	Light, Camera angle and distance
Scenario 2	Color present on an object

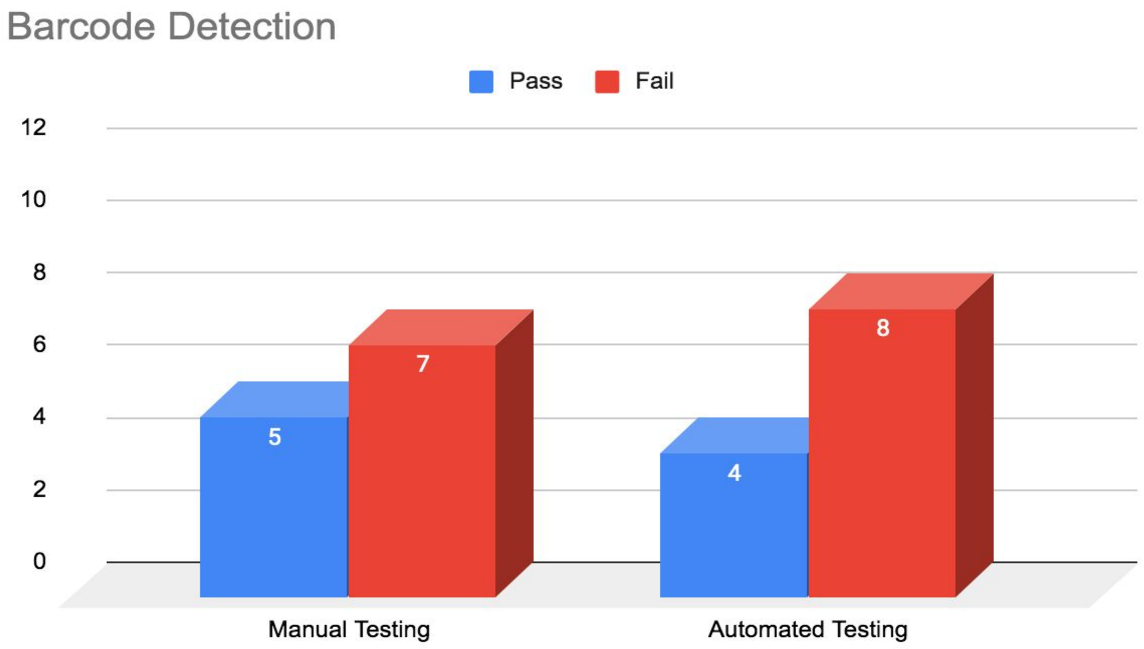
S.No.	**Test Automation Test Case**	**Expected Result**	**Manual Test Execution Result**	**Automated Test Execution Result**
3.2.3.2.1	Light - optimal	Detects the color	Pass	Pass

	Camera angle - optimal Distance - optimal Color - **Red**	Red		
3.2.3.2.2	Light - optimal Camera angle - optimal Distance - optimal Color - **Blue**	Detects the color Blue	Pass	Pass
3.2.3.2.3	Light - optimal Camera angle - optimal Distance - optimal Color - **Yellow**	Detects the color Yellow	Fail	Fail
3.2.3.2.4	Light - optimal Camera angle - optimal Distance - optimal Color - **Purple**	Detects the color Purple	Fail	Fail
3.2.3.2.5	Light - optimal Camera angle - optimal Distance - optimal Color - **Orange**	Detects the color Orange	Fail	Fail
3.2.3.2.6	Light - optimal Camera angle - optimal Distance - optimal Color - **Green**	Detects the color Green	Pass	Pass
3.2.3.2.7	Light - **Too Bright** Camera angle - optimal Distance - optimal Color - Red	Detects the color Red	Fail	Fail
3.2.3.2.8	Light - **Dark** Camera angle - optimal Distance - optimal Color - Red	Detects the color Red	Fail	Fail
3.2.3.2.9	Light - optimal Camera angle - **Slant** Distance - optimal Color - Red	Detects the color Red	Fail	Fail
3.2.3.2.10	Light - Optimal Camera angle - optimal Distance - **Too Far** Color - Red	Detects the color Red	Fail	Fail
3.2.3.2.11	Light - optimal Camera angle - **Slant** Distance - **Too close**	Detects the color Red	Fail	Fail

	Color - Blue			

3.2.4 Barcode Detection

Manual vs Automated	Total number of Test Cases	Pass	Fail
Manual Testing	12	5	7
Automated Testing	12	4	8



Input Type	Barcode
Context	Light, Camera angle and distance
Scenario 1	Barcode present on a real physical object

S.No.	Test Automation Test Case	Expected Result	Manual Test Execution Result	Automated Test Execution Result
3.2.4.1.1	Light - optimal Camera angle - optimal Distance - optimal	The App reads "Tissue"	Pass	Pass
3.2.4.1.2	Light - **Too Bright** Camera angle - optimal	The App reads "Tissue"	Fail	Fail

	Distance - optimal			
3.2.4.1.3	Light - **Dark** Camera angle - optimal Distance - optimal	The App reads “Tissue”	Fail	Fail
3.2.4.1.4	Light - Optimal Camera angle - Slant Distance - optimal	The App reads “Tissue”	Pass	Pass
3.2.4.1.5	Light - Optimal Camera angle - optimal Distance - **Too Close**	The App reads “Tissue”	Fail	Fail
3.2.4.1.6	Light - Optimal Camera angle - optimal Distance - **Too Far**	The App reads “Tissue”	Pass	Fail

Input Type	**Barcode**
Context	Light, Camera angle and distance
Scenario 2	On-Screen Barcode

S.No.	**Test Automation Test Case**	**Expected Result**	**Manual Test Execution Result**	**Automated Test Execution Result**
3.2.4.2.1	Light - optimal Camera angle - optimal Distance - optimal	The App reads “Tissue”	Pass	Pass
3.2.4.2.2	Light - **Too Bright** Camera angle - optimal Distance - optimal	The App reads “Tissue”	Fail	Fail
3.2.4.2.3	Light - **Dark** Camera angle - optimal Distance - optimal	The App reads “Tissue”	Fail	Fail
3.2.4.2.4	Light - Optimal Camera angle - Slant Distance - optimal	The App reads “Tissue”	Pass	Pass
3.2.4.2.5	Light - Optimal Camera angle - optimal Distance - **Too Close**	The App reads “Tissue”	Fail	Fail
3.2.4.2.6	Light - Optimal	The App reads	Fail	Fail

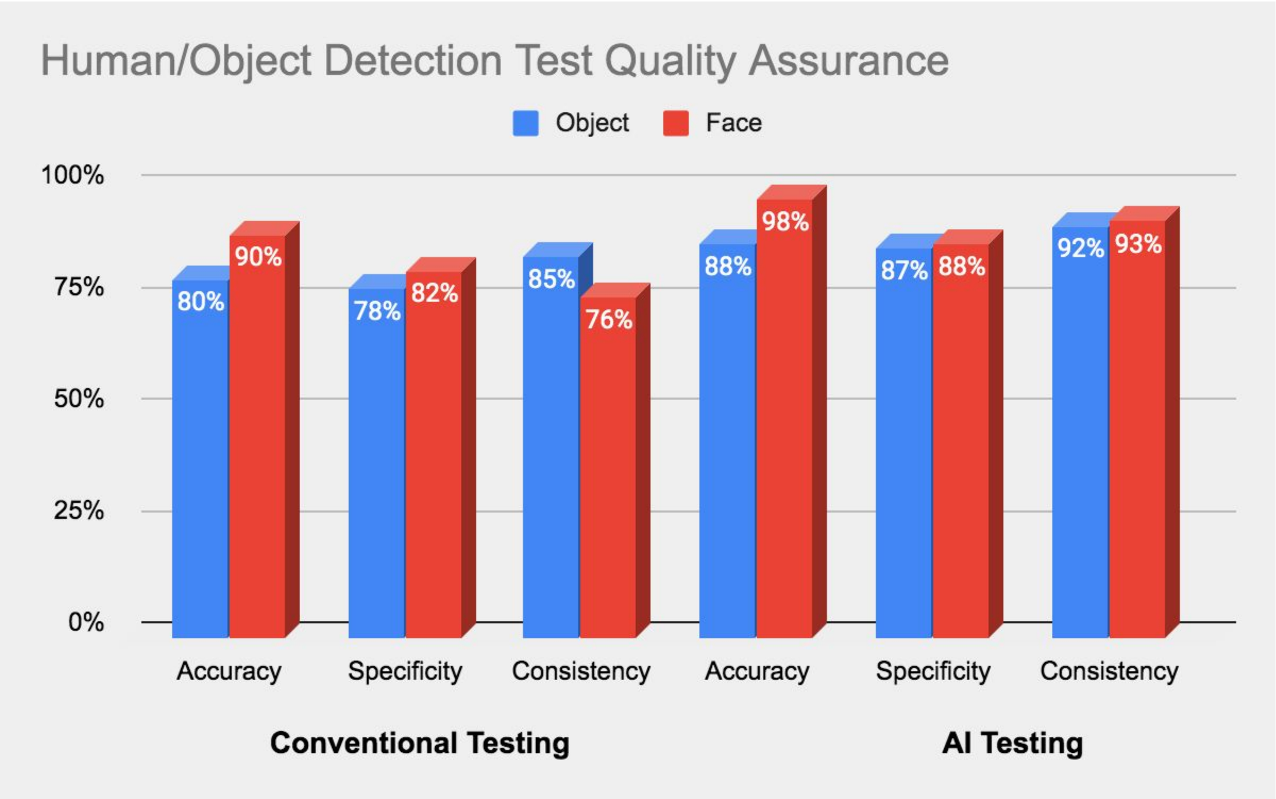
	Camera angle - optimal Distance - **Too Far**	“Tissue”		

4. Comparative AI Testing Quality Assurance and Bug Report

4.1 AI Testing Quality Assurance

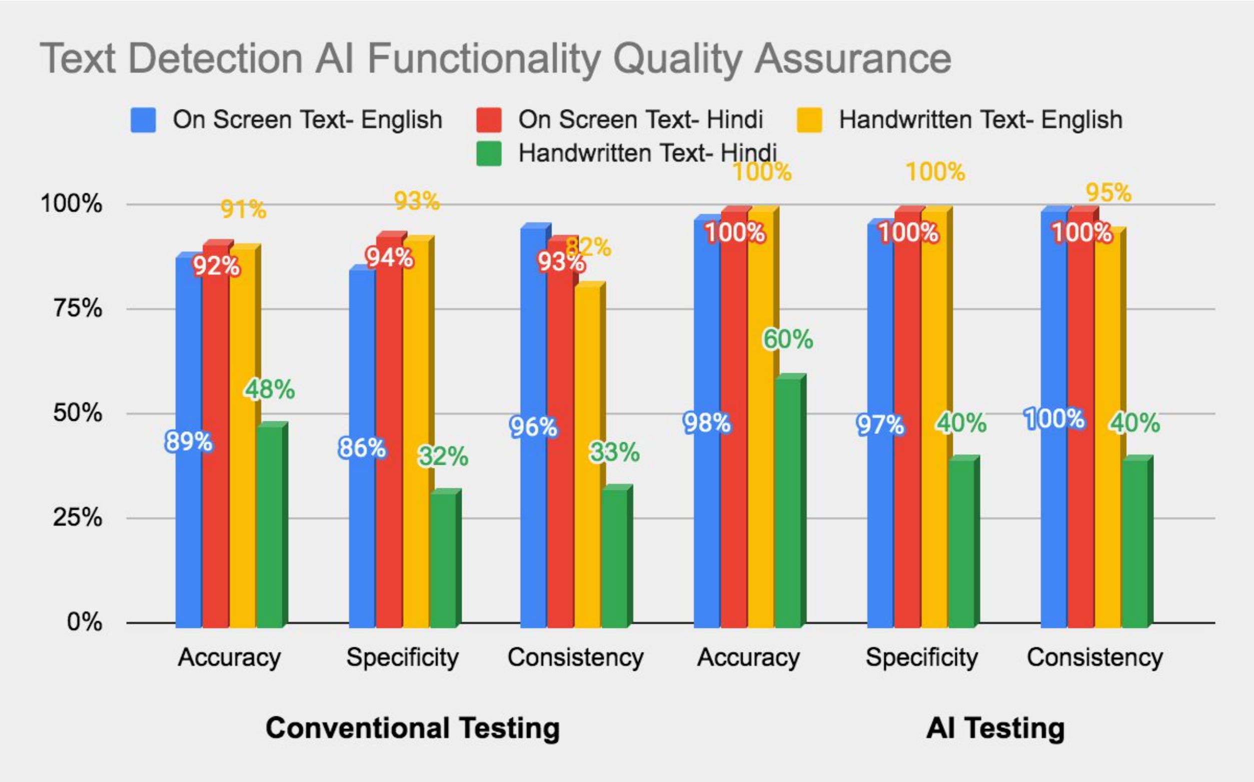
4.1.1 Human/Object Detection AI Functionality Quality Assurance

	AI Testing			Conventional Testing		
	Accuracy	**Specificity**	**Consistency**	**Accuracy**	**Specificity**	**Consistency**
Object	80%	78%	85%	88%	87%	92%
Face	90%	82%	76%	98%	88%	93%



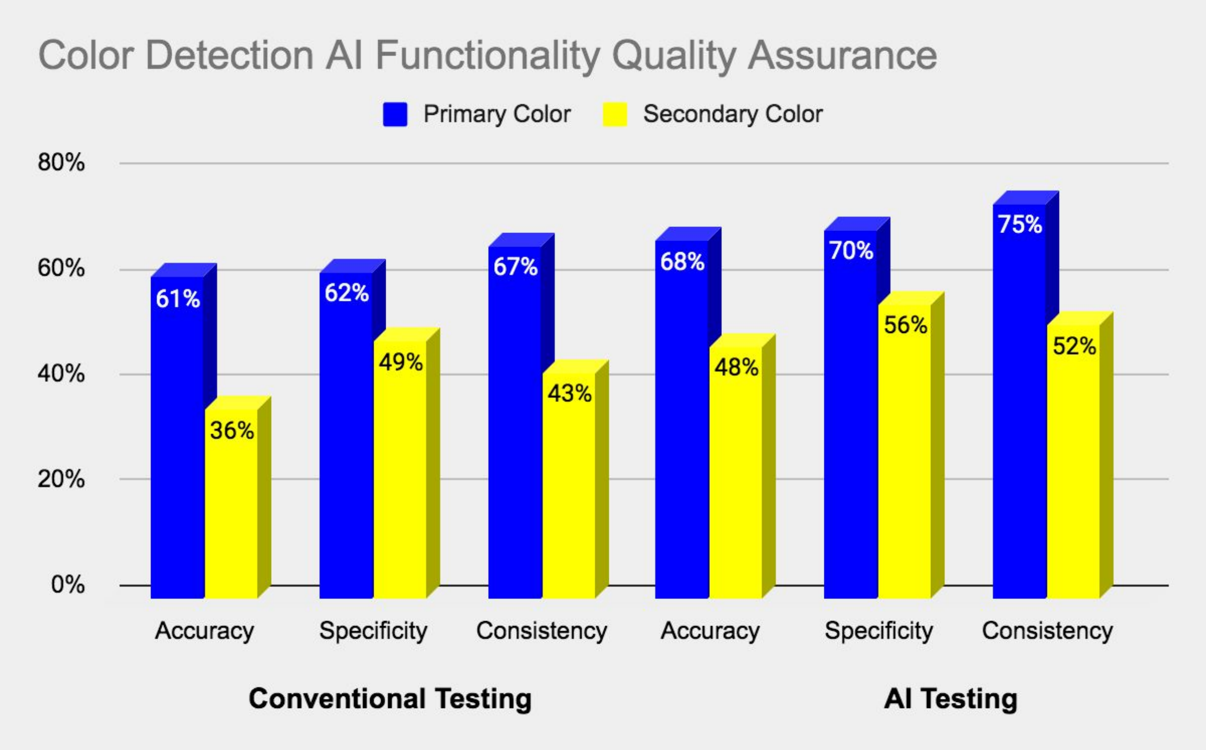
4.1.2 Text Detection AI Functionality Quality Assurance

	AI Testing			Conventional Testing		
	Accuracy	Specificity	Consistency	Accuracy	Specificity	Consistency
On Screen Text- English	89%	86%	96%	98%	97%	100%
On Screen Text- Hindi	92%	94%	93%	100%	100%	100%
Handwritten Text- English	91%	93%	82%	100%	100%	95%
Handwritten Text- Hindi	48%	32%	33%	60%	40%	40%



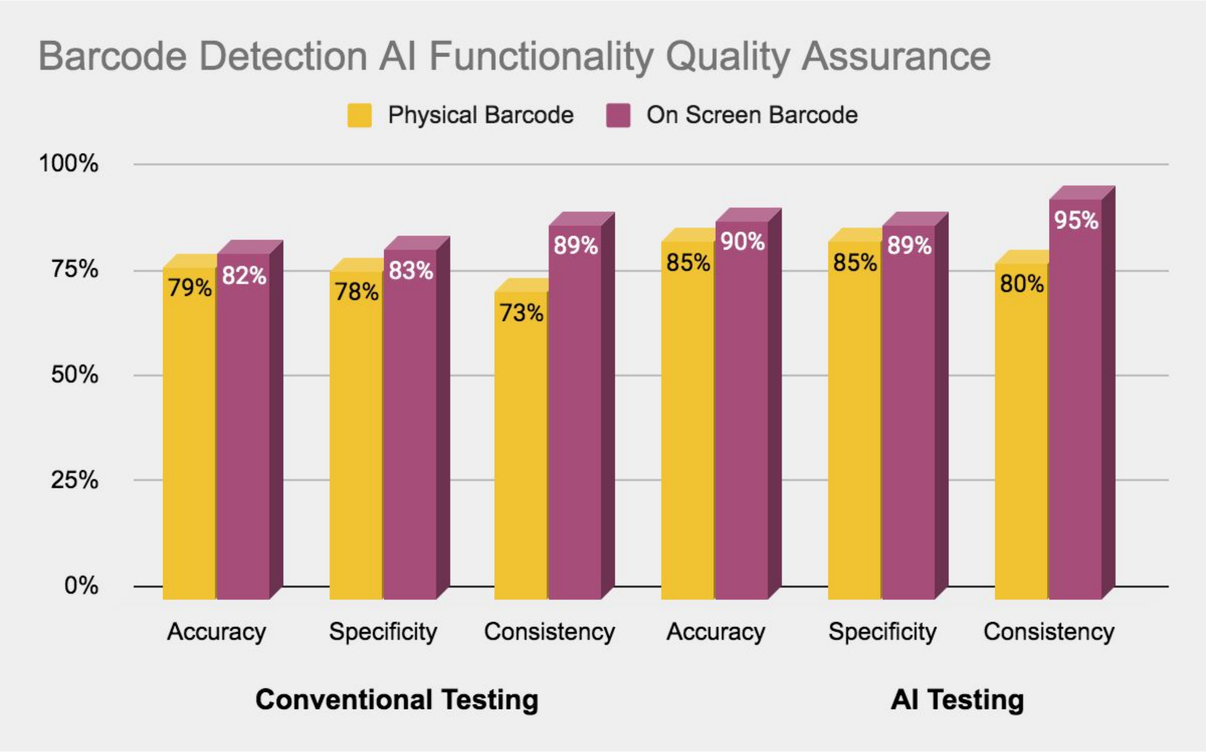
4.1.3 Color Detection AI Functionality Quality Assurance

	AI Testing			Conventional Testing		
	Accuracy	**Specificity**	**Consistency**	**Accuracy**	**Specificity**	**Consistency**
Primary Color	61%	62%	67%	68%	70%	75%
Secondary Color	36%	49%	43%	48%	56%	52%



4.1.4 Barcode Detection AI Functionality Quality Assurance

	AI Testing			Conventional Testing		
	Accuracy	Specificity	Consistency	Accuracy	Specificity	Consistency
Physical Barcode	79%	78%	73%	85%	85%	80%
On Screen Barcode	82%	83%	89%	90%	89%	95%



4.2 Comparative Bug Analysis (AI Testing vs Conventional Testing)

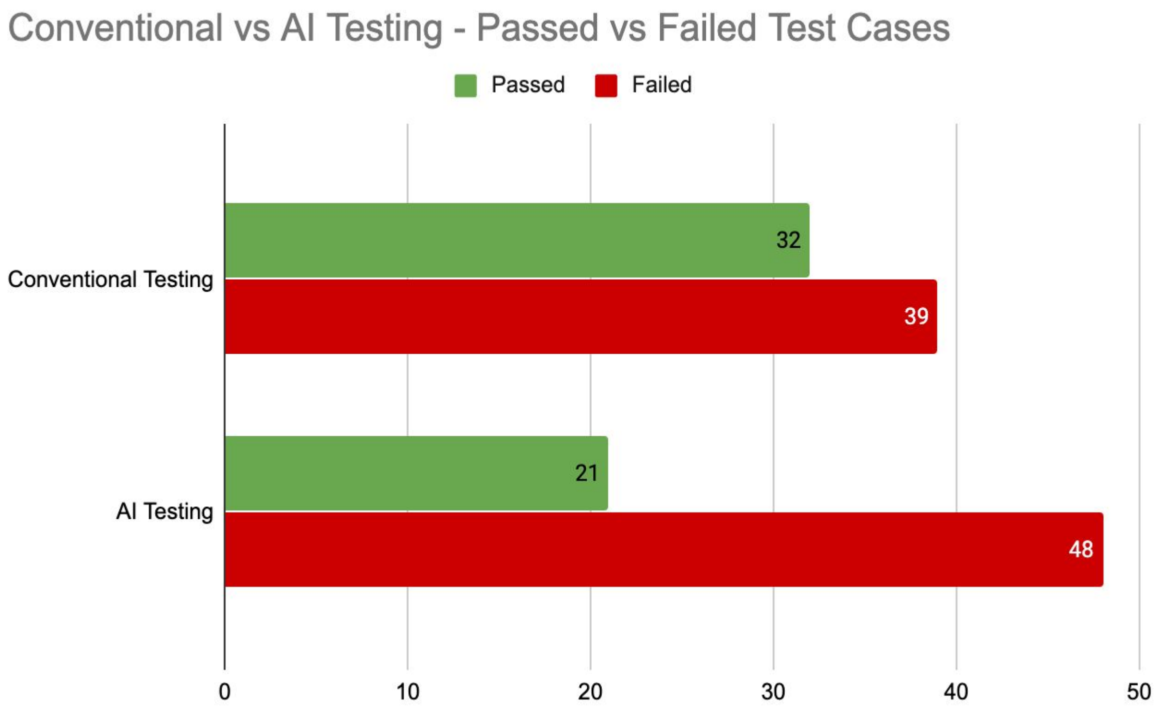
S.No.	Test Automation Test Case	Expected Result	Conventional Test execution result	Automated Test execution result
2.1.1.1	Light - optimal Noise - no noise Input background- Plain distance - <1 feet Location- Indoor Camera Orientation- 45 degree Camera Quality- Clear Action- Male who is standing	The action mentioned and the gender (M) should be detected correctly	Pass	Fail
2.1.1.3	Light - Artificial Light Noise - Distorted image Input background- Texture distance - 5 to 10 feet Location- Outdoor Camera Orientation- 90 degree Camera Quality- Sharp Action- Male who is walking	The action mentioned and the gender (M) should be detected correctly	Pass	Fail
2.1.1.5	Light - Natural Sunlight Noise - Shaking/Moving Image Input background- Plain distance - 5 to 10 feet Location- Indoor Camera Orientation- 45 degree Camera Quality- Clear Action- Male who is reading	The app is not able to identify the action and gender correctly	Pass	Fail
2.1.1.7	Light - Low Light Noise - Shaking image Input background- Texture distance - 1 to 5 feet Location- Indoor Camera Orientation- Straight Camera Quality- sharp	The app is not able to identify the action and gender correctly	Pass	Fail

	Action- Male who is standing			
2.1.1.12	Objects- Complete Object Person- N/A Creature- Wild Activity- Standing Quality- Sharpness	Envision AI should be able to identify the object correctly	Fail	Pass
3.2.2.1.1	Light - optimal Camera angle - optimal Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The App reads “I Love Testing”	Pass	Fail
3.2.2.1.4	Light - optimal Camera angle - **Slant** Distance - optimal Color of Text - black Size of Text - optimal Language of text- English	The App reads “I Love Testing”	Pass	Fail
3.2.2.1.6	Light - optimal Camera angle - optimal Distance - **Too Far** Color of Text - black Size of Text - optimal Language of text- English	The App reads “I Love Testing”	Pass	Fail
3.2.2.1.7	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Green** Size of Text - optimal Language of text- English	The App reads “I Love Testing”	Pass	Fail
3.2.2.1.8	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Red** Size of Text - optimal Language of text- **Hindi**	The app reads “मुझे कागज़ पर लिखना पसंद है”	Pass	Fail
3.2.2.1.11	Light - optimal Camera angle - optimal Distance - optimal Color of Text - **Blue** Size of Text - optimal Language of text- **Numerical**	The app reads “12,345,678”	Pass	Fail

3.2.4.1.6	Light - Optimal Camera angle - optimal Distance - **Too Far**	The App reads "Tissue"	Pass	Fail

Total Bug Statistics

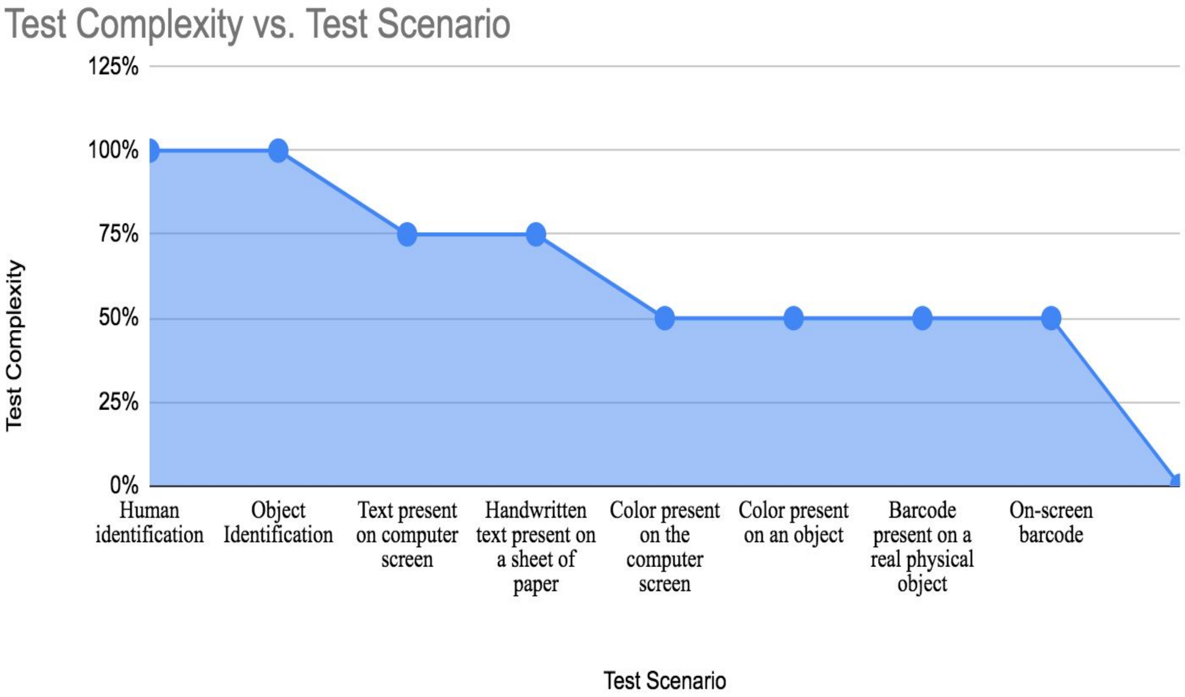
Conventional vs AI Testing	**Total number of test cases**	**Passed**	**Failed**
Conventional Testing	69	32	39
AI Testing	69	21	48



4.3 Comparative Test Complexity (AI Testing vs Conventional Testing)

Test Complexity of AI Functions (Scenario-Based)

AI Functionality	Test Scenario	Test Complexity
Human/Object Detection	Human identification	High Complexity
	Object Identification	High Complexity
Text Detection	Text present on computer screen	Medium Complexity
	Handwritten text present on a sheet of paper	Medium Complexity
Color Detection	Color present on the computer screen	Low Complexity
	Color present on an object	Low Complexity
Barcode Detection	Barcode present on a real physical object	Low Complexity
	On-screen barcode	Low Complexity


Test Complexity vs. Test Scenario
Test Complexity
Human identification
Object Identification
Text present on computer screen
Handwritten text present on a sheet of paper
Color present on the computer screen
Color present on an object
Barcode present on a real physical object
On-screen barcode
Test Scenario