Mobile Voting Application

Detailed Implementation Review

Version 1.0.0

Group 2

Zachary Bugay

Jamie Min

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 29 Mar 2018 | 1.0.0 | Final documentation write-up | Prepared by Jamie Min |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Implementation 3](#_Toc510047955)

[1.1 Functions 3](#_Toc510047956)

[1.2 Performance 3](#_Toc510047957)

[2. Software Manual 4](#_Toc510047958)

[2.1 Features 4](#_Toc510047959)

[3. Appendix 9](#_Toc510047960)

[3.1 Appendix A 9](#_Toc510047961)

[3.2 Appendix B 12](#_Toc510047962)

# Implementation

**Mobile Voting Application** is a collaborative-based social media platform that offers a unique opportunity for individuals to vote for their favorite poster. This document will review what will be implemented for the final milestone.

The application will primarily target University of Pittsburgh students.

## Functions

This section of the *Software Development Plan* is to gather all information necessary to outline the capabilities of the project. It describes the needs and approaches to the development of the software and is the top-level plan generated and used by managers to direct the development of the functions in this project.

The component will house two major functions. First, the voting component will combine the remote component into one useable API with each component being distinct but conjoined. Secondly, the component will have a TDD component to aid in debugging and test cases for the application.

## Performance

This section of the *Software Development Plan* describes the overall anticipated performance of **Mobile Voting Application**, including deployment of the product.

* System still performs as expected
* Interacting components still perform as expected

# Software Manual

This section will cover the newly implemented features of the application.

## Features

As seen in Figure 1, the user is presented with the main screen of the voting component. After the user has enabled the server, the user must connect and register the client to the server. The user then has the option to set posters and enable or close voting. The user also has the option to view results. And finally, the user now has the option to run test cases.

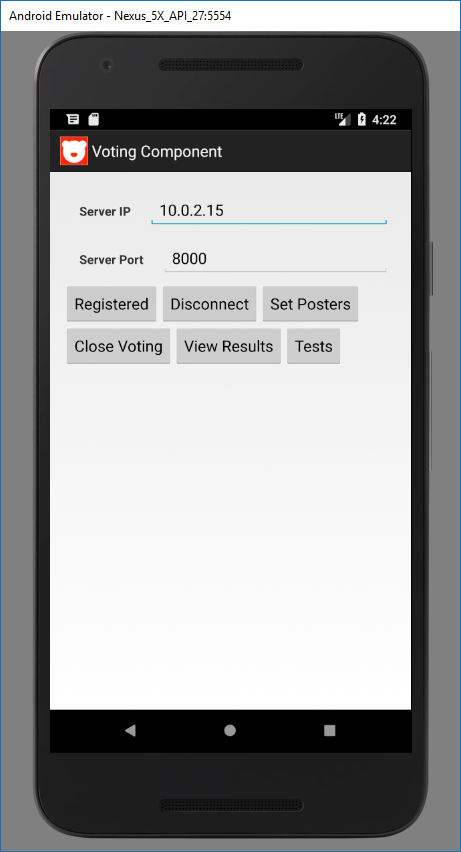


Figure - Main Screen

As seen in Figure 2, when the user presses the “Run Tests” button, the user is presented with administrative test tools to determine if the component is ready to be deployed. See Appendix B for the XML/KVLs being loaded.

* Button 701 sends a standard 701 message to the client and a 711 is returned.
* Button 702 sends a standard 702 message to the client and a 712 is returned.
* Button 703 sends a standard 703 message to the client, and a test pair is created.
* Run Tests runs the automated test script that allows the user to determine if all the features are working as intended. See Appendix A for source code.



Figure - Test Console

As seen in Figure 3, when the “Run Tests” button has been pressed. An automated script will detail with of the test cases have passed and which has failed. The following figures represents different scenarios of test cases.

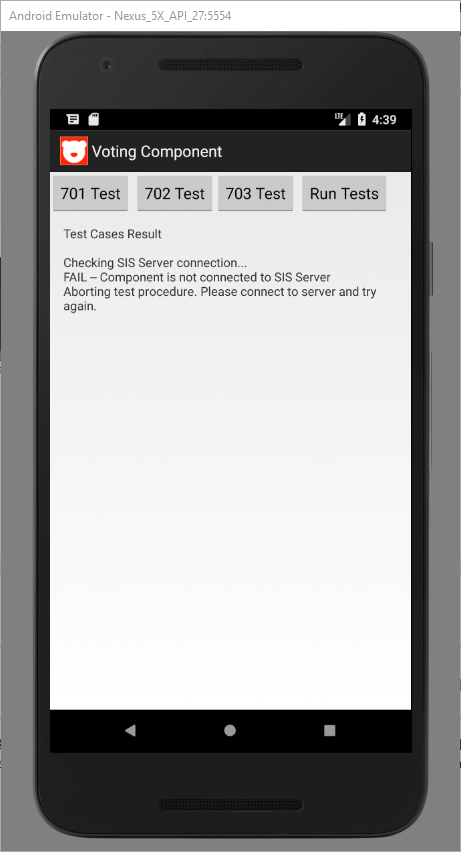
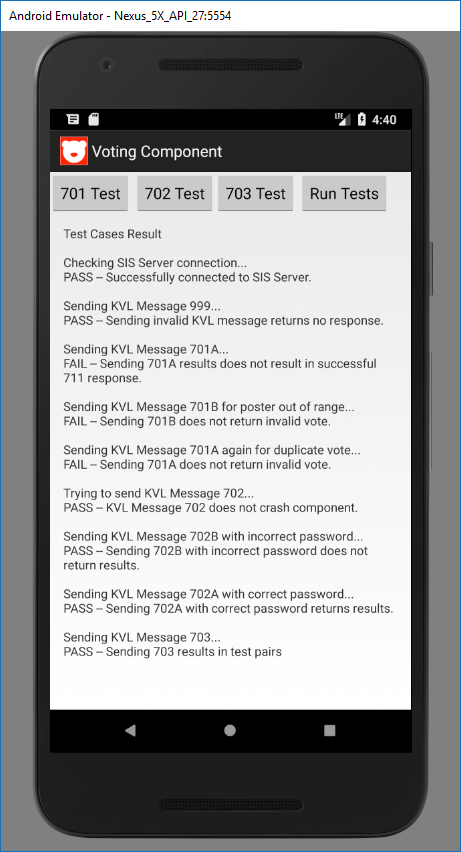


Figure - Server Not Connected

  
Figure - Test Case Voting Not Enabled

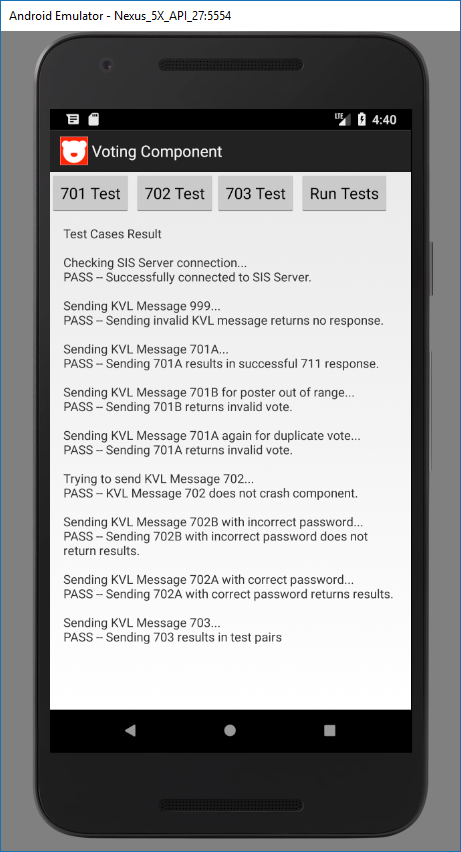


Figure - Test Case All Scenarios Passed

# Appendix

This section will cover the newly implemented features of the application.

## Appendix A

Source code for TestingActivity.java used for the TDD component.

package input;

import android.app.Activity;

import android.os.Bundle;

import android.os.StrictMode;

import android.text.method.ScrollingMovementMethod;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

import org.tdr.R;

import java.util.HashSet;

import java.util.Set;

public class TestingActivity extends Activity {

Button b701, b702, b703, bTests;

TextView tc;

boolean canContinue = true;

static Set<String> msgs = new HashSet<>();

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_testing);

StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();

StrictMode.setThreadPolicy(policy);

b701 = (Button) findViewById(R.id.button701);

b702 = (Button) findViewById(R.id.button702);

b703 = (Button) findViewById(R.id.button703);

bTests = (Button) findViewById(R.id.buttonRunTests);

tc = (TextView) findViewById(R.id.testConsole);

tc.setMovementMethod(new ScrollingMovementMethod());

//set button handlers

b701.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

MainActivity.sendMessage(KVLLoader.get701A());

}

});

b702.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

MainActivity.sendMessage(KVLLoader.get702A());

}

});

b703.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

MainActivity.sendMessage(KVLLoader.get703());

}

});

bTests.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Toast.makeText(TestingActivity.this,"Running Tests...",Toast.LENGTH\_SHORT).show();

tc.append("Test Cases Result\n\n");

// CHECKING CONNECTION

tc.append("Checking SIS Server connection...\n");

if (MainActivity.client != null && MainActivity.client.isSocketAlive())

tc.append("PASS -- Successfully connected to SIS Server.\n");

else {

tc.append("FAIL -- Component is not connected to SIS Server\n");

canContinue = false;

}

if (canContinue) {

// CHECKING 999 KVL Message

tc.append("\nSending KVL Message 999...\n");

MainActivity.sendMessage(KVLLoader.get999());

if (msgs.isEmpty())

tc.append("PASS -- Sending invalid KVL message returns no response.\n");

else

tc.append("FAIL -- Sending invalid KVL message returns response.\n");

// Sending 701 results in 711

tc.append("\nSending KVL Message 701A...\n");

MainActivity.sendMessage(KVLLoader.get701A());

if (msgs.contains("711"))

tc.append("PASS -- Sending 701A results in successful 711 response.\n");

else

tc.append("FAIL -- Sending 701A results does not result in successful 711 response.\n");

// Sending 701 for poster out of range results invalid

tc.append("\nSending KVL Message 701B for poster out of range...\n");

MainActivity.sendMessage(KVLLoader.get701B());

if (msgs.contains("Invalid711"))

tc.append("PASS -- Sending 701B returns invalid vote.\n");

else

tc.append("FAIL -- Sending 701B does not return invalid vote.\n");

// Sending 701 for duplicate vote results invalid

tc.append("\nSending KVL Message 701A again for duplicate vote...\n");

MainActivity.sendMessage(KVLLoader.get701A());

if (msgs.contains("Invalid711"))

tc.append("PASS -- Sending 701A returns invalid vote.\n");

else

tc.append("FAIL -- Sending 701A does not return invalid vote.\n");

// Sending 702 does not crash component

tc.append("\nTrying to send KVL Message 702...");

try {

MainActivity.sendMessage(KVLLoader.get702A());

tc.append("\nPASS -- KVL Message 702 does not crash component.\n");

// Sending 702 with incorrect password

tc.append("\nSending KVL Message 702B with incorrect password...\n");

MainActivity.sendMessage(KVLLoader.get702B());

if (msgs.contains("Invalid712"))

tc.append("PASS -- Sending 702B with incorrect password does not return results.\n");

else

tc.append("FAIL -- Sending 702B with correct password returns results.\n");

} catch (Exception e) {

tc.append("\nFAIL -- KVL Message 702 does not work as intended.\n");

}

// Sending 702 with correct password results in pass

tc.append("\nSending KVL Message 702A with correct password...\n");

MainActivity.sendMessage(KVLLoader.get702A());

for (String s : msgs)

System.out.println(s);

if (msgs.contains("712"))

tc.append("PASS -- Sending 702A with correct password returns results.\n");

else

tc.append("FAIL -- Sending 702A with correct password does not return results.\n");

// Sending 703 returns test pair

tc.append("\nSending KVL Message 703...\n");

MainActivity.sendMessage(KVLLoader.get703());

if (msgs.contains("Test"))

tc.append("PASS -- Sending 703 results in test pairs");

else

tc.append("FAIL -- Sending 703 does not result in test pairs");

} else {

tc.append("Aborting test procedure. Please connect to server and try again.\n");

}

}

});

}

}

## Appendix B

Source code for the KVL/XML test scripts loader.

package input;

public class KVLLoader {

public static KeyValueList get701A() {

KeyValueList ret = new KeyValueList();

ret.putPair("Scope", "SIS");

ret.putPair("MsgID", "701");

ret.putPair("MessageType", "Reading");

ret.putPair("VoterID", "2674757720");

ret.putPair("CandidateID", "3");

ret.putPair("Test701A", "Test701A");

return ret;

}

public static KeyValueList get701B() {

KeyValueList ret = new KeyValueList();

ret.putPair("Scope", "SIS");

ret.putPair("MsgID", "701");

ret.putPair("MessageType", "Reading");

ret.putPair("VoterID", "2674757721");

ret.putPair("CandidateID", "99");

ret.putPair("Test701B", "Test701B");

return ret;

}

public static KeyValueList get702A() {

KeyValueList ret = new KeyValueList();

ret.putPair("Scope", "SIS");

ret.putPair("MsgID", "702");

ret.putPair("MessageType", "Reading");

ret.putPair("Passcode", "1631");

ret.putPair("N", "2");

ret.putPair("Test702", "Test702");

return ret;

}

public static KeyValueList get702B() {

KeyValueList ret = new KeyValueList();

ret.putPair("Scope", "SIS");

ret.putPair("MsgID", "702");

ret.putPair("MessageType", "Reading");

ret.putPair("Passcode", "9999");

ret.putPair("N", "2");

ret.putPair("Test702", "Test702");

return ret;

}

public static KeyValueList get703() {

KeyValueList ret = new KeyValueList();

ret.putPair("Scope", "SIS");

ret.putPair("MsgID", "703");

ret.putPair("MessageType", "Reading");

//ret.putPair("Sender", "VotingComponent");

ret.putPair("CandidateList", "1,2,3");

ret.putPair("Passcode", "1631");

ret.putPair("Test703", "Test703");

return ret;

}

public static KeyValueList get999() {

KeyValueList ret = new KeyValueList();

ret.putPair("Scope", "SIS");

ret.putPair("MsgID", "999");

ret.putPair("MessageType", "Reading");

return ret;

}

}