System and Unit Test Report

Product name:

FilterShare

Team name:

Team 16

Date:

07/25/16

1. System Test scenarios:

A. User story 1 from Sprint 1: As a social media user, I want to be able to take/select photos so that I can share them with friends.
B. User story 2 from Sprint 1: As a photo editor, I want to make a filter so that I can have customized filters.
C. User story 3 from Sprint 1: As a photo editor, I want a non-intrusive editing page to create and preview filters.
D. User story 4 from Sprint 1: As a software developer, I want an organized backend system so that I can save and retrieve data for filters.

Scenario 1 for Sprint 1:

1. start FilterShare app
2. take a new photo with camera by selecting the ‘camera’ icon
3. wants to take the photo again; select ‘RE-TAKE’
4. select the ‘change camera’ icon and change the camera to ‘selfie’ mode
5. take a new photo in selfie mode
6. select ‘CONFIRM’
7. User should see a page with image preview and eight effect adjustment buttons on the bottom.
8. select one of <brightness, contrast, saturation, fade, temperature, tint, vignette, grain> icons
9. move the slider to another point
10. User should see the change in selected attribute of the image on preview
11. select ‘CANCEL’
12. The preview is restored to the state before adjustment; the bottom bar displays eight attribute buttons again.

Scenario 2 for Sprint 1:

1. start FilterShare app
2. select the ‘gallery’ icon and view photos in gallery

3. select one photo from gallery
4. select ‘CONFIRM’
5. User should see a page with image preview and eight effect adjustment buttons on the bottom.
6. select one of <brightness, contrast, saturation, fade, temperature, tint, vignette, grain> icons
7. move the slider to another point
8. User should see the change in selected attribute of the image on preview
9. select one of <brightness, contrast, saturation, fade, temperature, tint, vignette, grain> icons
10. move the slider to another point
11. User should see the change in selected attribute of the image on preview
12. select ‘DONE’
13. The preview keeps the change made on the image; the bottom bar displays eight attribute buttons again.

E. User story 1 from Sprint 2: As a photo editor, I want a non-intrusive editing page to create and preview filters.
F. User story 2 from Sprint 2: As a social media user, I want to be able to share filters with other users on the app so that I can contribute to the community.

Scenario 1 for Sprint 2:

1. With the image that you’ve chosen, you can make your own filter by selecting any of the following attributes; <Brightness,Contrast,Saturation, Fade,Temperature,Tint,Vignette,Grain>
2. After selecting a single attribute, you can adjust the value associated with that with the slider which appears, to achieve the desired filter effect.
3. If you don’t like the image you’re working on now, you can press ‘Change Image’ to re-choose the image.
4. Once you’re done, press ‘Next’ to share the filter.
5. You can input the name of the filter, your username, and the hashtags you want to describe the filter.
6. Once you’re done, press ‘Share’ to share the filter. After that, you will go to share filter activity. If you don’t like the filter at this moment, press ‘Back’ button to go back to filter making activity.

G. User story 1 from Sprint 3: As a social media user, I want to be able to find filters other people share so that I can make better photos.

Scenario 1 for Sprint 3:

1. Start ‘share_filter’ activity after choosing the image with camera / gallery.
2. Look at the list of filters applied to the image that you’ve chosen. Scroll down to take a look at more filters. Before you reach the end of the scroll, the app will automatically fetch more filters from the server.

3. You can choose the order of filters in the view. You can either choose to see the recent filters first or the most popular filters first. Also, you can see your own filters at this page.
4. If you have chosen a filter in mind, touch that image. Then you will see the
5. specific value of each attributes the filter has. Press ‘SAVE’ button to use the filter to the image that you’ve chosen.

H. User story 2 from Sprint 3: As a user, I want to understand the functions of the application easily so that I can make the best use of it.

Scenario 2 for Sprint 3:

1. Download and start FilterShare app
2. Notice the permissions request for camera and external storage and accept them.
3. Select ‘Ok’, then user should be able to open the app with the camera preview open.
4. As it is the first execution of the app, the tutorial that describes the function of the camera page starts.
5. Touch ‘Got it’, then the user can select the picture and goes on to the next page.
6. The consecutive tutorials that describes other pages start.

2. Unit Tests and Automated Build System.

A. Unit Tests

a. Where to find:
   i. follow a link in the Testing.md file on Github under project root directory
   ii. follow a soft link Testing by entering “cd Testing” on terminal command line in the project root directory

1. Bitmap Processing Test

```
package team16.filtershare;

import android.graphics.Color;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

/**
* Created by chocho on 7/26/16.
*/

public class BitmapProcessingTest {
```

```
@Before
public void setUp() {
}

@Test
public void brightness_FiftyAsArgumentValue_ReturnSameColor() {
    assertEquals(BitmapProcessing.brightness(Color.RED, 50), Color.RED);
}

@Test
public void brightness_RGBOver255_BoundTo255() {
    assertEquals(BitmapProcessing.brightness(0xfffefefe, 100), 0xffffffff);
}

@Test
public void brightness_RGBUnder0_BoundTo0() {
    assertEquals(BitmapProcessing.brightness(0xff010101, 0), 0xff000000);
}

@Test
public void brightness_HigherArgumentValue_ReturnBrighterPixel() {
    assertTrue(BitmapProcessing.brightness(Color.RED, 70) > Color.RED);
}

@Test
public void brightness_LowerArgumentValue_ReturnDarkerPixel() {
    assertTrue(BitmapProcessing.brightness(Color.RED, 20) < Color.RED);
}

@Test
public void contrast_FiftyAsArgumentValue_ReturnSameColor() {
    assertEquals(BitmapProcessing.contrast(Color.RED, 50), Color.RED);
}

@Test
public void fade_ZeroAsArgumentValue_ReturnSameBitmap() {
    assertEquals(BitmapProcessing.fade(Color.RED, 0), Color.RED);
}

@Test
public void fade_HigherArgumentValue_ReturnBrighterPixel() {
    assertTrue(BitmapProcessing.fade(Color.RED, 70) > Color.RED);
}

@Test
public void temperature_FiftyAsArgumentValue_ReturnSameColor() {
    assertEquals(BitmapProcessing.temperature(Color.RED, 50), Color.RED);
}

@Test
public void temperature_ArgumentValueLargerThanFifty_ReturnHigherRLowerGLowerB() {
    int color = 0x808080;
    assertTrue(((BitmapProcessing.temperature(color, 80) >> 16) & 0xff)
```

```
            > ((color >> 16) & 0xff));
        assertTrue(((BitmapProcessing.temperature(color, 80) >> 8) & 0xff)
            < ((color >> 8) & 0xff));
        assertTrue((BitmapProcessing.temperature(color, 80)& 0xff)
            < (color & 0xff));
    }

    @Test
    public void temperature_ArgumentValueLessThanFifty_ReturnLowerRHigherGHigherB() {
        int color = 0x808080;
        assertTrue(((BitmapProcessing.temperature(color, 20) >> 16) & 0xff)
            < ((color >> 16) & 0xff));
        assertTrue(((BitmapProcessing.temperature(color, 20) >> 8) & 0xff)
            > ((color >> 8) & 0xff));
        assertTrue((BitmapProcessing.temperature(color, 20)& 0xff)
            > (color & 0xff));
    }

    @Test
    public void tint_ZeroAsArgumentValue_ReturnSameColor() {
        assertEquals(BitmapProcessing.tint(Color.RED, 0), Color.RED);
    }

    @Test
    public void tint_ArgumentValueLargerThan0_ReturnHigherR() {
        int color = 0x808080;
        assertTrue(((BitmapProcessing.tint(color, 80) >> 16) & 0xff)
            > ((color >> 16) & 0xff));
    }

    @Test
    public void grain_ZeroAsArgumentValue_ReturnSameColor() {
        assertEquals(BitmapProcessing.grain(Color.RED, 0), Color.RED);
    }

    @After
    public void tearDown() {

    }
}
```

2. MainactivityTest

```
package team16.filtershare;

// MainActivityTest.java

import android.os.Build;
import android.widget.ImageButton;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.robolectric.Robolectric;
import org.robolectric.RobolectricGradleTestRunner;
```

```
import org.robolectric.annotation.Config;

import static junit.framework.Assert.assertEquals;
import static junit.framework.Assert.assertNotNull;

/**
* Created by harrykim on 2016. 7. 26..
*/

//This codes are derived from the robolectric tutorial from codepath.com to check the entire activity cycles
@RunWith(RobolectricGradleTestRunner.class)
@Config(constants = BuildConfig.class, sdk = Build.VERSION_CODES.LOLLIPOP)
public class MainActivityTest {
  private MainActivity main_activity;


  @Before
  public void setUp() {
     main_activity = Robolectric.setupActivity(MainActivity.class);
  }


  // Test that simulates the full lifecycle of an activity
  @Test
  public void Button_CaputreButtonExists() {
     //createWithIntent("my extra_value");
     assertNotNull(main_activity);
     ImageButton capture_button = (ImageButton) main_activity.findViewById(R.id.button_capture);
     assertNotNull(capture_button);
     assertEquals(capture_button.getHeight(), 70);

     // ... add assertions ...
  }

  @Test
  public void Button_GalleryButtonExists() {
     //createWithIntent("my extra_value");
     assertNotNull(main_activity);
     ImageButton gallery_button = (ImageButton) main_activity.findViewById(R.id.button_gallery);
     assertNotNull(gallery_button);
     assertEquals(gallery_button.getHeight(), 70);

     // ... add assertions ...
  }

  @Test
  public void Button_ChangeButtonExists() {
     //createWithIntent("my extra_value");
     assertNotNull(main_activity);
     ImageButton change_button = (ImageButton) main_activity.findViewById(R.id.button_gallery);
     assertNotNull(change_button);
     assertEquals(change_button.getHeight(), 70);

     // ... add assertions ...
```

```
    }

    @Test
    public void Rect_AutoFocusRectExists() {
        AutofocusRect mAutofocusRect = (AutofocusRect) main_activity.findViewById(R.id.af_rect);
        int button_height = mAutofocusRect.getHeight();
        assertNotNull(mAutofocusRect);
        assertEquals(button_height, 60);
    }

}
```

B. Automated Build System(Jenkins)

The Jenkins detects the push in the github project and then auto-builds the project.