

Mobile Embedded Devices
Group Contribution Report
UFCFW5-30-2

Andrew Belcher
StudentID:17010347

March 28, 2019

The code discussed in this report regarding the "Spy App" can be found at..

`https://gitlab.uwe.ac.uk/a2-belcher/med`

Contents

1	Task specification	4
1.1	Android App Development	4
1.2	Android OS customization	4
2	Spy App	5
2.1	Develop Image Capture and Audio Recording Functionality . . .	5
2.2	Develop a Login UI	10
2.3	Develop a User Registration Login UI	12
2.4	Store Encrypted Username and Password	14
2.5	Verify Username and Password for Login Process	18
2.6	Develop a UI to Change the Password	22
2.7	Bonus Functionality	25
2.7.1	Correct live camera overlay orientation	25
2.7.2	Correct image view orientation	25
2.7.3	Add hide UI button(s) to hide both live camera view and image view	26
2.7.4	Cover the whole UI with a screenshot from a popular app such as Facebook or Gmail	29
2.7.5	Make ‘Register’ button disabled after user inputs their username and password	30
2.7.6	Changing the user password requires verifying current pass- word	32
3	Android OS Customization	34
3.1	Android Source Code Download Setup	34
3.2	Compile Android Source Code	34
3.3	Custom Boot Animation	35

1 Task specification

1.1 Android App Development

This will involve you writing an Android app with a login interface and secure data storage (encrypted data) to store the username and password. Once the user logs in, the “Spy App” will allow recording of audio and the capture of still images using the built-in microphone and camera. Implementing basic camera features and audio capture will be achieved from the camera and audio worksheet. Additional functional requirements will require group self-directed study of the relevant topics.

1.2 Android OS customization

Following the provided worksheets, download/configure/and compile Lineage OS 14.1 source code(flo) for the Nexus 7 tablet. Demonstrate a custom boot animation by swapping out the corresponding files and rebuilding the OS so that it can be installed on the device. Detail any issues experienced.

2 Spy App

Tasked with developing a "Spy App" for an Android Nexus 7 tablet under API 23, our group was able to complete all requested functional requirements. These ranged from the ability to take pictures and record audio as well as store.

2.1 Develop Image Capture and Audio Recording Functionality

Following the previous image capture worksheet, I contributed my working camera app code to the "Spy App" since it was pretty stable at this point. The code to do this is performed in a few functions..

```
1
2     private android.hardware.Camera.PictureCallback captureMedia =
3         new Camera.PictureCallback()
4     {
5         @Override
6         public void onPictureTaken(byte[] data, Camera camera) {
7             Matrix matrix = new Matrix();
8             matrix.postRotate(90F);
9             Bitmap bitmap = BitmapFactory.decodeByteArray(data, 0,
10                 data.length);
11
12             Bitmap newBitmap = Bitmap.createBitmap(bitmap, 0, 0,
13                 bitmap.getWidth(), bitmap.getHeight(), matrix,
14                 true);
15
16             if (bitmap == null) {
17             } else {
18                 pictureTaken.setImageBitmap(newBitmap);
19                 pictureTaken.setBackgroundResource(R.drawable.
20                     ic_launcher_background);
21                 pictureTaken.setZ(6);
22                 pictureTaken.setVisibility(View.VISIBLE);
23                 Button captureButton = findViewById(R.id.
24                     captureButton);
25                 Button recordButton = findViewById(R.id.
26                     recordButton);
27                 captureButton.setEnabled(false);
28                 recordButton.setEnabled(false);
29                 captureButton.setVisibility(View.INVISIBLE);
30                 recordButton.setVisibility(View.INVISIBLE);
31                 FrameLayout cameraPreview = findViewById(R.id.
32                     camPreview);
33                 cameraPreview.setVisibility(View.INVISIBLE);
34                 ConstraintLayout background = findViewById(R.id.
35                     camBackground);
36                 background.setBackgroundColor(Color.GRAY);
37
38                 // Make directory called RecordData if it doesn't
39                 already exist
```

```

31         File f = new File(Environment.
32             getExternalStorageDirectory(),
33             folder_main);
34         if (!f.exists()) {
35             f.mkdirs();
36         }
37         String path = Environment.
38             getExternalStorageDirectory().toString();
39
40         // enter the following code statement all on one
41         // line
42         String filename = path + "/" + folder_main + "/" +
43             "recordData" + String.format("%d.jpg", System.
44             currentTimeMillis());
45         FileOutputStream outputStream = null;
46         try {
47             outputStream = new FileOutputStream(filename);
48             outputStream.write(data);
49             outputStream.close();
50         } catch (FileNotFoundException e) {
51             e.printStackTrace();
52         } catch (IOException e) {
53             e.printStackTrace();
54         } finally {
55             }
56         }
57         cameraObject.startPreview();
58     }
59 };
60
61 public void captureImage(View view)
62 {
63     cameraObject.takePicture(null, null, captureMedia);
64 }
65
66 public void recordAudio(View view)
67 {
68     if(!recording_running)
69     {
70
71         // Make directory called RecordData if it doesn't
72         // already exist
73         File f = new File(Environment.
74             getExternalStorageDirectory(), folder_main);
75
76         if (!f.exists()) {
77             f.mkdirs();
78         }
79
80         String path = Environment.getExternalStorageDirectory().
81             toString();
82
83         // enter the following code statement all on one line
84         String filename = path + "/" + folder_main + "/" + "
85             recordData" + String.format("%d.3gp", System.
86             currentTimeMillis());
87         recorder = new MediaRecorder();

```

```

78         recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
79         recorder.setOutputFormat(MediaRecorder.OutputFormat.
            THREE_GPP);
80         recorder.setAudioEncoder(MediaRecorder.AudioEncoder.
            AMR_NB);
81         recorder.setOutputFile(filename);
82
83         try {
84             recorder.prepare();
85         } catch (IllegalStateException e) {
86             // TODO Auto-generated catch block
87             e.printStackTrace();
88         } catch (IOException e) {
89             // TODO Auto-generated catch block
90             e.printStackTrace();
91         }
92         recorder.start(); // Recording is now started
93         recorder_open = true;
94     }
95
96     else
97         recorder.stop();
98
99     recording_running = !recording_running;
100 }

```

This aspect of the app also required graceful permission requesting which was accomplished with the following..

```
1
2
3     private static final int REQUEST_CAMERA = 0;
4     private static final int REQUEST_CONTACTS = 1;
5     private static final int REQUEST_EXTERNAL_STORAGE = 2;
6     private static final int AUDIO_RECORD_REQUEST_CODE = 300;
7     private static String[] PERMISSIONS_CONTACT = {Manifest.
        permission.READ_CONTACTS, Manifest.permission.
        WRITE_CONTACTS};
8
9     String[] PERMISSIONS = {
10         android.Manifest.permission.WRITE_EXTERNAL_STORAGE,
11         android.Manifest.permission.RECORD_AUDIO,
12         android.Manifest.permission.CAMERA
13     };
14
15     // CLASS METHODS
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         cur_page = current_page.LOGIN;
20         prev_page = current_page.LOGIN;
21
22         setContentView(R.layout.activity_spy_app_login);
23         initCrypto();
24
25         if(checkAppPermissions())
26             requestAppPermissions();
27
28         enableTextListenier();
29     }
30
31     ...
32
33     private void requestAppPermissions()
34     {
35         Log.i(TAG, "CAMERA/Record audio/write storage permission
36             has NOT been granted. Requesting permission.");
37         ActivityCompat.requestPermissions(SpyApp.this, PERMISSIONS,
38             1);
39     }
40
41     private boolean checkAppPermissions()
42     {
43         return
44         (
45             // we dont have camera privs
46             ActivityCompat.checkSelfPermission(this, Manifest.
47                 permission.CAMERA)
48             != PackageManager.PERMISSION_GRANTED
49
50             // we dont have audio privs
51             || ActivityCompat.checkSelfPermission(this,
52                 Manifest.permission.RECORD_AUDIO)
53             != PackageManager.PERMISSION_GRANTED
```



```
50
51      // we dont have write storage privs
52      || ActivityCompat.checkSelfPermission(this,
53          Manifest.permission.WRITE_EXTERNAL_STORAGE)
54      != PackageManager.PERMISSION_GRANTED);
    }
```

2.2 Develop a Login UI

Now that Camera/Recording functionality was maintained it was time for us to develop a login interface. My work towards this goal was focused on layout switching to focus on displaying a login layout on class creation. As well as input verification involving the login fields and linking different login related buttons to other layouts. Below is the final code of the application that related to login verification.

```
1 public void handleLoginAttempt(View view)
2 {
3     Log.i(TAG, "ON LOGIN CLICK attempts:" + login_attempts);
4
5     EditText userText = (EditText)findViewById(R.id.login_username)
6     ;
7     EditText passText = (EditText)findViewById(R.id.login_password)
8     ;
9     String username = userText.getText().toString();
10    String password = passText.getText().toString();
11
12    // check if user exists!
13    // check if password matches
14    if(checkAppPermissions()) {
15        requestAppPermissions();
16        userText.setText("");
17        passText.setText("");
18    }
19    else {
20        Document doc = getXmlDoc();
21
22        // our credentials file
23        File file = new File(Environment.
24            getExternalStorageDirectory(), folder_main+"/
25            credentials/details.sxml");
26        if (file.exists())
27        {
28            if (username.equals("") || password.equals("")) {
29                if (username.equals(""))
30                    Toast.makeText(getApplicationContext(), "
31                        Username cant be empty", Toast.LENGTH_SHORT
32                        ).show();
33                else
34                    Toast.makeText(getApplicationContext(), "
35                        Password cant be empty", Toast.LENGTH_SHORT
36                        ).show();
37
38                userText.setText("");
39                passText.setText("");
40            }
41
42            else if (usernameExists(username, doc) &&
43                credentialsMatch(username, password, doc)) {
```

```

38         drawApp();
39     }
40     else {
41
42         TextView attempt_text = findViewById(R.id.
43             attempts_view);
44         attempt_text.setVisibility(View.VISIBLE);
45         TextView attempt_num = findViewById(R.id.
46             attempt_num);
47         attempt_num.setVisibility(View.VISIBLE);
48
49         login_attempts--;
50         attempt_num.setText(Integer.toString(
51             login_attempts));
52
53         userText.setText("");
54         passText.setText("");
55
56         if (login_attempts == 0) {
57             attempt_num.setTextColor(Color.RED);
58             attempt_text.setTextColor(Color.RED);
59
60             Button btn = (Button) findViewById(R.id.
61                 login_confirm);
62             btn.setEnabled(false);
63         }
64     }
65 }
66 else {
67 {
68     if (username.equals("") || password.equals("")) {
69         if (username.equals(""))
70             Toast.makeText(getApplicationContext(), "
71                 Username cant be empty", Toast.LENGTH_SHORT
72                 ).show();
73         else
74             Toast.makeText(getApplicationContext(), "
75                 Password cant be empty", Toast.LENGTH_SHORT
76                 ).show();
77     }
78     else {
79         Toast.makeText(getApplicationContext(), "User does
80             not exist, please register!", Toast.
81             LENGTH_SHORT).show();
82     }
83     userText.setText("");
84     passText.setText("");
85 }
86 }
87 }
88 }

```

2.3 Develop a User Registration Login UI

With the ability to connect a login layout to the camera layout and verify input (without credential verification yet) we could now go on to developing a registration page. I was able to develop code to verify fields as well and later on actually update the credentials file.

```
1
2 public void handleRegisterAttempt(View view) {
3
4     EditText userText = (EditText)findViewById(R.id.
5         reg_username);
6     EditText passText = (EditText)findViewById(R.id.
7         passchg_newpass);
8     EditText passConfirmText = (EditText)findViewById(R.id.
9         passchg_newpass_confirm);
10    String username = userText.getText().toString();
11    String password = passText.getText().toString();
12    String passwordConfirm = passConfirmText.getText().toString()
13        ();
14
15    // gain access to filesystem write priv first
16    if (checkAppPermissions()) {
17        requestAppPermissions();
18        userText.setText("");
19        passText.setText("");
20        passConfirmText.setText("");
21    }
22
23    // handle register attempt
24    else
25    {
26
27        File filedir = new File(Environment.
28            getExternalStorageDirectory(), folder_main+"/
29            credentials");
30
31        // our credentials file
32        File file = new File(Environment.
33            getExternalStorageDirectory(), folder_main+"/
34            credentials/details.xml");
35
36        if (username.equals("") || password.equals("")) {
37
38            if (username.equals(""))
39                Toast.makeText(getApplicationContext(), "
40                    Username cant be empty", Toast.LENGTH_SHORT
41                ).show();
42            else if (password.equals(""))
43                Toast.makeText(getApplicationContext(), "
44                    Password cant be empty", Toast.LENGTH_SHORT
45                ).show();
46
47            userText.setText("");
48            passText.setText("");
49            passConfirmText.setText("");
50        }
51    }
```

```

39
40     else if (password.equals(passwordConfirm)) {
41         if (file.exists()) {
42
43             Document doc = getXmlDoc();
44
45             if (!usernameExists(username, doc))
46             {
47                 addNewUser(username, password, doc);
48             }
49             else {
50                 Toast.makeText(getApplicationContext(), "
                    name is already taken", Toast.
                        LENGTH_SHORT).show();
51             }
52         }
53         else {
54
55             filedir.mkdirs();
56
57             String templateCredData = "<?xml version
                =\"1.0\" encoding=\"UTF-8\"?>\n\n" +
58                 "<credentials>\n" +
59                 "    <user id=\"" + username + "\">\n" +
60                 "        <name>%s</name>\n" +
61                 "        <pass>%s</pass>\n" +
62                 "    </user>\n" +
63                 "</credentials>";
64
65             templateCredData = String.format(
                templateCredData, 1, username, password);
66
67             if (writeCredentialsFile(templateCredData)) ;
68             else
69                 Log.i(TAG, "Error in writing encrypted file
                    back!!");
70
71         }
72         onBackPressed();
73     }
74     else
75     {
76         userText.setText("");
77         passText.setText("");
78         passConfirmText.setText("");
79
80         Toast.makeText(getApplicationContext(), "
            Confirmation password does not match!", Toast.
                LENGTH_SHORT).show();
81     }
82 }
83 }
84 }

```

2.4 Store Encrypted Username and Password

To tie most of the needed login functionality together, we needed to store credentials on the file system which would also need to be encrypted for safe storage. My role was to develop code to parse an xml file that is initially encrypted in AES-128-CBC. This would require me to also implement code to encrypt/decrypt the file before storing and retrieving.

```
1
2 // Encrypt a byte array in AES-128-CBC
3 public byte[] encrypt(byte[] data)
4 {
5     try
6     {
7         byte[] iv = new byte[16];
8         cipher.init(ENCRYPT_MODE, skey, new IvParameterSpec(iv));
9         DeflaterInputStream deflaterInput = new
10             DeflaterInputStream(new CipherInputStream(new
11                 DeflaterInputStream(new ByteArrayInputStream(data))
12                 , cipher));
13         Log.i(TAG, "iv:" + byteArrayToHex(iv));
14         return IOUtils.toByteArray(deflaterInput);
15     }
16     catch (Exception e)
17     {
18         e.printStackTrace();
19         return new byte[0];
20     }
21 }
22
23 // Decrypt a byte array in AES-128-CBC
24 public byte[] decrypt(byte[] data)
25 {
26     byte[] result = new byte[0];
27
28     try
29     {
30         byte[] iv = new byte[16];
31         cipher.init(DECRYPT_MODE, skey, new IvParameterSpec(iv));
32
33         InflaterInputStream inflaterStream = new
34             InflaterInputStream(new CipherInputStream(new
35                 InflaterInputStream(new ByteArrayInputStream(data))
36                 , cipher));
37         return IOUtils.toByteArray(inflaterStream);
38     }
39     catch (Exception e)
40     {
41         Log.e(TAG, "decryption out failed: "+Log.
42             getStackTraceString(e));
43         return result;
44     }
45 }
```

```

41 // Pass in document file to convert to StringWriter for xml
    encryption
42 public StringWriter writeXml(Document doc)
43 {
44     StringWriter writer = null;
45
46     try
47     {
48         DOMSource domSource = new DOMSource(doc);
49         writer = new StringWriter();
50         StreamResult result = new StreamResult(writer);
51         TransformerFactory tf = TransformerFactory.newInstance
            ();
52         Transformer transformer = tf.newTransformer();
53         transformer.transform(domSource, result);
54         System.out.println("XML IN String format is: " + writer
            .toString());
55     }
56
57     catch (javax.xml.transform.
        TransformerConfigurationException e)
58     {
59         e.printStackTrace();
60     }
61
62     catch (javax.xml.transform.TransformerException e)
63     {
64         e.printStackTrace();
65     }
66
67     return writer;
68 }
69
70 public InputStream readCredentialsFile()
71 {
72     File file = new File(Environment.
        getExternalStorageDirectory(), folder_main+"/
        credentials/details.sxml");
73     int size = (int) file.length();
74     byte[] sxml = null;
75     byte[] xml = null;
76
77     sxml = new byte[size];
78
79     try
80     {
81         BufferedInputStream buf = new BufferedInputStream(new
            FileInputStream(file));
82         buf.read(sxml, 0, sxml.length);
83         buf.close();
84     }
85
86     catch (FileNotFoundException e)
87     {
88         // TODO Auto-generated catch block
89         e.printStackTrace();
90     }

```

```

91     catch (IOException e)
92     {
93         // TODO Auto-generated catch block
94         e.printStackTrace();
95     }
96
97     xml = decrypt(sxml);
98     Log.i(TAG, "new decrypted data is: "+byteArrayToHex(xml));
99
100    // String str = new String(dectest, "UTF-8"); // for UTF-8
        encoding
101    InputStream decxml = new ByteArrayInputStream(xml);
102    Log.i(TAG, "new decrypted file is: " + decxml);
103
104    return decxml;
105 }
106
107 public boolean writeCredentialsFile(String credFile)
108 {
109     // Make directory called RecordData if it doesn't already
        exist
110    //File f = new File(Environment.getExternalStorageDirectory
        (), folder_main+"/credentials/");
111    String path = Environment.getExternalStorageDirectory().
        toString();
112    String filename = path + "/" + folder_main + "/" + "
        credentials/" + "details.sxml";
113
114    byte[] xml = null;
115    byte[] sxml = null;
116
117    try
118    {
119        Log.i(TAG, "key is: " + byteArrayToHex(key));
120        Log.i(TAG, "skey is: " + byteArrayToHex(skey.getEncoded()
        ));
121        Log.i(TAG, "key length: " + key.length);
122
123        xml = credFile.getBytes();
124        Log.i(TAG, "decrypted data is: " + byteArrayToHex(xml));
125        sxml = encrypt(xml);
126        Log.i(TAG, "encrypted data is: "+byteArrayToHex(sxml));
127        xml = decrypt(sxml);
128        Log.i(TAG, "new decrypted data is: "+byteArrayToHex(xml)
        );
129        String str = new String(xml, "UTF-8"); // for UTF-8
            encoding
130        Log.i(TAG, "new decrypted file is: "+str);
131    }
132
133    catch (java.lang.Exception e)
134    {
135        Log.e(TAG, "Exception: "+Log.getStackTraceString(e));
136    }
137
138    // enter the following code statement all on one line
139    FileOutputStream outputStream = null;

```



```

140
141
142     try
143     {
144         outputStream = new FileOutputStream(filename);
145         outputStream.write(sxml);
146         outputStream.close();
147         return true;
148     }
149     catch (FileNotFoundException e)
150     {
151         e.printStackTrace();
152     }
153     catch (IOException e)
154     {
155         e.printStackTrace();
156     }
157     finally
158     {
159         return false;
160     }
161
162 public Document getXmlDoc()
163 {
164     InputStream credFile = readCredentialsFile();
165
166     Document doc = null;
167
168     try{
169         DocumentBuilderFactory dbFactory =
170             DocumentBuilderFactory.newInstance();
171         DocumentBuilder dBuilder = dbFactory.newDocumentBuilder
172             ();
173         doc = dBuilder.parse(credFile);
174
175         Element element = doc.getDocumentElement();
176         element.normalize();
177     }
178     catch(org.xml.sax.SAXException e)
179     {
180         Log.e(TAG, "sax exception failed: " + Log.
181             getStackTraceString(e));
182     }
183     catch(javax.xml.parsers.ParserConfigurationException e)
184     {
185         e.printStackTrace();
186     }
187     catch(java.io.IOException e)
188     {
189         e.printStackTrace();
190     }
191     return doc;
192 }

```

2.5 Verify Username and Password for Login Process

After implementing functionality needed to store credentials, we could move on to developing username and password verification to complete the login process. I provided code to look through the parsed credentials and determine if the input data matched with a stored users' credentials. First of which was to check if a user existed, and second was to determine if their credentials matched. Also, code for handling the login attempt was developed by me for verifying attempts and input data.

```
1
2
3     public boolean usernameExists(String usernameChecked, Document
4         doc)
5     {
6         Element element = doc.getDocumentElement();
7         element.normalize();
8
9         NodeList nList = doc.getElementsByTagName("user");
10
11         for (int i = 0; i < nList.getLength(); i++) {
12             Node node = nList.item(i);
13             if (node.getNodeType() == Node.ELEMENT_NODE) {
14                 if (node.getNodeName().equals("user")) {
15
16                     Element element2 = (Element) node;
17                     NodeList userCreds = element2.getChildNodes
18                         ();
19
20                     for (int d = 0; d < userCreds.getLength();
21                         d++) {
22                         Node credList = userCreds.item(d);
23
24                         if (credList.getNodeType() == Node.
25                             ELEMENT_NODE) {
26                             if (credList.getNodeName().equals("
27                                 name")) {
28
29                                 Element element3 = (Element)
30                                     credList;
31
32                                 if (usernameChecked.equals(
33                                     element3.getTextContent()))
34                                 {
35                                     Log.i(TAG, "node: " +
36                                         element3.getTextContent
37                                             ());
38                                     return true;
39                                 }
40                             }
41                         }
42                     }
43                 }
44             }
45         }
46     }
```

```

37     return false;
38 }
39
40 public boolean credentialsMatch(String username, String
41     password, Document doc)
42 {
43     Element element = doc.getDocumentElement();
44     boolean onUserEntry = false;
45
46     element.normalize();
47
48     NodeList nList = doc.getElementsByTagName("user");
49
50     for (int i = 0; i < nList.getLength(); i++) {
51         Node node = nList.item(i);
52         if (node.getNodeType() == Node.ELEMENT_NODE) {
53             if (node.getNodeName().equals("user")) {
54
55                 Element element2 = (Element) node;
56                 NodeList userCreds = element2.getChildNodes();
57
58                 for (int d = 0; d < userCreds.getLength(); d++)
59                 {
60                     Node credList1 = userCreds.item(d);
61                     //Node credList2 = userCreds.item(d+);
62
63                     if (credList1.getNodeType() == Node.
64                         ELEMENT_NODE) {
65                         Element element3 = (Element) credList1;
66
67                         if (credList1.getNodeName().equals("
68                             name")) {
69                             if (username.equals(element3.
70                                 getTextContent())) {
71                                 onUserEntry = true;
72                             }
73                         }
74                     }
75                     if(credList1.getNodeName().equals("pass
76                         ") && onUserEntry)
77                     {
78                         if(password.equals(element3.
79                             getTextContent())){
80                             Log.i(TAG, "pass: " + element3.
81                                 getTextContent());
82                             return true;
83                         }
84                     }
85                     onUserEntry = false;
86                 }
87             }
88         }
89     }
90
91     return false;
92 }

```

```

86
87 public void handleLoginAttempt(View view)
88 {
89     Log.i(TAG, "ON LOGIN CLICK attempts:" + login_attempts);
90
91     EditText userText = (EditText)findViewById(R.id.
        login_username);
92     EditText passText = (EditText)findViewById(R.id.
        login_password);
93     String username = userText.getText().toString();
94     String password = passText.getText().toString();
95
96
97     // check if user exists!
98     // check if password matchs
99     if(checkAppPermissions()) {
100         requestAppPermissions();
101         userText.setText("");
102         passText.setText("");
103     }
104     else {
105         Document doc = getXmlDoc();
106
107         // our credentials file
108         File file = new File(Environment.
            getExternalStorageDirectory(), folder_main+"/
            credentials/details.xml");
109         if (file.exists())
110         {
111
112             if (username.equals("") || password.equals("")) {
113                 if (username.equals(""))
114                     Toast.makeText(getApplicationContext(), "
                        Username cant be empty", Toast.
                            LENGTH_SHORT).show();
115
116                 else
117                     Toast.makeText(getApplicationContext(), "
                        Password cant be empty", Toast.
                            LENGTH_SHORT).show();
118
119                 userText.setText("");
120                 passText.setText("");
121             }
122
123             else if (usernameExists(username, doc) &&
                credentialsMatch(username, password, doc)) {
124                 drawApp();
125             }
126             else {
127
128                 TextView attempt_text = findViewById(R.id.
                    attempts_view);
129                 attempt_text.setVisibility(View.VISIBLE);
130                 TextView attempt_num = findViewById(R.id.
                    attempt_num);
131                 attempt_num.setVisibility(View.VISIBLE);

```

```

132
133         login_attempts--;
134         attempt_num.setText(Integer.toString(
135             login_attempts));
136
137         userText.setText("");
138         passText.setText("");
139
140         if (login_attempts == 0) {
141             attempt_num.setTextColor(Color.RED);
142             attempt_text.setTextColor(Color.RED);
143
144             Button btn = (Button) findViewById(R.id
145                 .login_confirm);
146             btn.setEnabled(false);
147         }
148     }
149     else
150     {
151         if (username.equals("") || password.equals("")) {
152             if (username.equals(""))
153                 Toast.makeText(getApplicationContext(), "
154                     Username cant be empty", Toast.
155                         LENGTH_SHORT).show();
156             else
157                 Toast.makeText(getApplicationContext(), "
158                     Password cant be empty", Toast.
159                         LENGTH_SHORT).show();
160         }
161         else {
162             Toast.makeText(getApplicationContext(), "User
163                 does not exist, please register!", Toast.
164                     LENGTH_SHORT).show();
165         }
166         userText.setText("");
167         passText.setText("");
168     }
169 }
170 }

```

2.6 Develop a UI to Change the Password

At this point we had now had our app checking user credentials for logging into the camera section, as well as the ability to register a new user. But we hadn't implemented the functionality of changing an existing users' password. In this section of development, I was able to develop the code necessary to update the password after decrypting and parsing the credentials file, updating and restoring the file afterwards. Verification code for the password update pages fields were written by myself also.

```
1
2 public void handlePassChangeAttempt(View view)
3 {
4     EditText userText = (EditText)findViewById(R.id.
5         passchg_username);
6     EditText oldPassText = (EditText)findViewById(R.id.
7         passchg_prevpass);
8     EditText newPassText = (EditText)findViewById(R.id.
9         passchg_newpass);
10    EditText newPassConfirmText = (EditText)findViewById(R.id.
11        passchg_newpass_confirm);
12    String username = userText.getText().toString();
13    String oldPassword = oldPassText.getText().toString();
14    String newPassword = newPassText.getText().toString();
15    String newPasswordConfirm = newPassConfirmText.getText().
16        toString();
17
18    // gain access to filesystem write priv first
19    if(checkAppPermissions()) {
20        requestAppPermissions();
21
22        userText.setText("");
23        oldPassText.setText("");
24        newPassText.setText("");
25        newPassConfirmText.setText("");
26    }
27    // handle pass change attempt
28    else
29    {
30        Document doc = getXmlDoc();
31
32        if(usernameExists(username,doc) && credentialsMatch(
33            username,oldPassword,doc))
34        {
35            if(newPassword.equals(newPasswordConfirm))
36            {
37                changeUserPassword(username,newPassword,doc);
38                onBackPressed();
39            }
40            else
41            {
42                userText.setText("");
43                oldPassText.setText("");
44                newPassText.setText("");
45                newPassConfirmText.setText("");

```

```

41
42         Toast.makeText(getApplicationContext(), "New
           password does not match confirmation",
           Toast.LENGTH_SHORT).show();
43     }
44
45     }
46     else
47     {
48         if(!usernameExists(username, doc))
49             Toast.makeText(getApplicationContext(), "User
           does not exist", Toast.LENGTH_SHORT).show()
           ;
50         else
51             Toast.makeText(getApplicationContext(), "Old
           password does not match", Toast.
           LENGTH_SHORT).show();
52
53         userText.setText("");
54         oldPassText.setText("");
55         newPassText.setText("");
56         newPassConfirmText.setText("");
57     }
58
59 }
60
61 }
62
63 public void handlePassChangeRequest(View view)
64 {
65     prev_page = current_page.LOGIN;
66     setContentView(R.layout.activity_spy_app_passchange);
67     cur_page = current_page.PASSCHANGE;
68 }
69
70
71 public void changeUserPassword(String username, String password
72     , Document doc)
73 {
74     StringWriter writer = null;
75
76     Element element = doc.getDocumentElement();
77     boolean onUserEntry = false;
78
79     element.normalize();
80
81     NodeList nList = doc.getElementsByTagName("user");
82
83     Log.i(TAG, "in password change\n\n ");
84
85     for (int i = 0; i < nList.getLength(); i++) {
86
87         Node node = nList.item(i);
88         if (node.getNodeType() == Node.ELEMENT_NODE) {
89             if (node.getNodeName().equals("user")) {
90
91                 Element element2 = (Element) node;

```

```

91         NodeList userCreds = element2.getChildNodes();
92
93         for (int d = 0; d < userCreds.getLength(); d++)
94         {
95             Node credList1 = userCreds.item(d);
96             //Node credList2 = userCreds.item(d+);
97
98             if (credList1.getNodeType() == Node.
99                 ELEMENT_NODE)
100             {
101                 Element element3 = (Element) credList1;
102
103                 if (credList1.getNodeName().equals("
104                     name"))
105                 {
106                     if (username.equals(element3.
107                         getTextContent()))
108                     {
109                         Log.i(TAG, "name matchs!: " +
110                             element3.getTextContent());
111                         onUserEntry = true;
112                     }
113                 }
114             }
115         }
116     }
117 }
118
119
120
121 // transform doc back to xml string
122 writer = writeXml(doc);
123
124 if(writeCredentialsFile(writer.toString()));
125 else
126     Log.i(TAG, "Error in writing encrypted file back!!");
127
128 }

```


2.7 Bonus Functionality

Finally, we have a working spy application with a login interface, however our task had not been complete yet. We were given the option of 2 bonus functionality requirement sets. We chose the first one involving more UI related functions that will be detailed in this section.

2.7.1 Correct live camera overlay orientation

First of the bonus requirements, the camera needed to stay fixed and correctly display when rotating the tablet. To do this I offered the solution of locking the layout in the Manifest.

```
1 <activity android:name=".SpyApp" android:screenOrientation="
    portrait">
```

2.7.2 Correct image view orientation

Luckily my offered solution worked for the orientation of the image view as well, since it locked the whole app to portrait mode.

2.7.3 Add hide UI button(s) to hide both live camera view and image view

This feature required us to hide everything on the camera layout with the touch of a button, I developed the code needed to hide the UI and then restore its functionality by tapping the back button with the hidden mode detected.

```
1
2
3     public boolean hidden_ui = false;
4
5     public void hide_ui(View view)
6     {
7         if(cur_page == current_page.CAMERA && !hidden_ui)
8         {
9             Button captureButton = findViewById(R.id.captureButton)
10                ;
11             Button recordButton = findViewById(R.id.recordButton);
12             Button hide_ui_btn = findViewById(R.id.hide_ui_btn);
13             FrameLayout cameraPreview = findViewById(R.id.
14                camPreview);
15             ImageView imgView = findViewById(R.id.imageView);
16             imgView.setVisibility(View.INVISIBLE);
17             cameraPreview.setVisibility(View.INVISIBLE);
18             captureButton.setVisibility(View.INVISIBLE);
19             recordButton.setVisibility(View.INVISIBLE);
20             hide_ui_btn.setVisibility(View.INVISIBLE);
21             getSupportActionBar().hide();
22
23             hidden_ui = true;
24         }
25     }
26     public void unhide_ui()
27     {
28         Button captureButton = findViewById(R.id.captureButton)
29            ;
30         Button recordButton = findViewById(R.id.recordButton);
31         Button hide_ui_btn = findViewById(R.id.hide_ui_btn);
32         FrameLayout cameraPreview = findViewById(R.id.
33            camPreview);
34         ImageView imgView = findViewById(R.id.imageView);
35         // imgView.setVisibility(View.VISIBLE);
36         cameraPreview.setVisibility(View.VISIBLE);
37         captureButton.setVisibility(View.VISIBLE);
38         recordButton.setVisibility(View.VISIBLE);
39         hide_ui_btn.setVisibility(View.VISIBLE);
40         getSupportActionBar().hide();
41
42         hidden_ui = false;
43     }
44
45     @Override
46     public void onBackPressed() {
```

```

47         if(cur_page == current_page.LOGIN)
48         {
49             Intent intent = new Intent(Intent.ACTION_MAIN);
50             intent.addCategory(Intent.CATEGORY_HOME);
51             intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
52             startActivity(intent);
53         }
54         else if(cur_page == current_page.FAKE)
55         {
56             getSupportActionBar().show();
57
58             if(prev_page == current_page.LOGIN) {
59                 setContentView(R.layout.activity_spy_app_login);
60                 cur_page = current_page.LOGIN;
61                 enableTextListenier();
62             }
63             else if(prev_page == current_page.CAMERA) {
64
65                 drawApp();
66
67             }
68             else if(prev_page == current_page.REGISTER) {
69                 setContentView(R.layout.activity_spy_app_register);
70                 cur_page = current_page.REGISTER;
71             }
72             else
73             {
74                 setContentView(R.layout.activity_spy_app_passchange
75                     );
76                 cur_page = current_page.PASSCHANGE;
77             }
78         }
79         else
80         {
81             if(cur_page == current_page.CAMERA && hidden_ui)
82             {
83                 unhide_ui();
84             }
85             else {
86                 login_attempts = 3;
87
88                 cur_page = current_page.LOGIN;
89                 setContentView(R.layout.activity_spy_app_login)
90                     ;
91
92                 enableTextListenier();
93
94                 TextView attempt_text = findViewById(R.id.
95                     attempts_view);
96                 attempt_text.setVisibility(View.INVISIBLE);
97                 TextView attempt_num = findViewById(R.id.
98                     attempt_num);
99                 attempt_num.setVisibility(View.INVISIBLE);
100
101                 Button btn = findViewById(R.id.login_confirm);
102                 btn.setEnabled(true);

```

```
100      }  
101      }  
102  
103      }
```

2.7.4 Cover the whole UI with a screenshot from a popular app such as Facebook or Gmail

In this portion of the bonus functionality I had helped by coming up with a method to trigger a new "incognito" type layout which was provide by my group partner. If so, many taps were experienced on the touch screen at once, then this layout would be swapped to and the rest of the UI hidden as to appear as the user was playing "2048" on their device. I was able to develop the following..

```
1
2
3     @Override
4     public boolean onTouchEvent(MotionEvent event) {
5
6         if(cur_page != current_page.FAKE) {
7             super.onTouchEvent(event);
8
9             Date currentTime = Calendar.getInstance().getTime();
10            Log.d(TAG, "TOUCH!! red time:" + currentTime + "\n\n");
11            if (currentTime == currentTime)
12                touchCount++;
13            if (touchCount > 25) {
14                prev_page = cur_page;
15                setContentView(R.layout.activity_spy_app_fake);
16                cur_page = current_page.FAKE;
17
18
19                ImageView image = findViewById(R.id.imageView3);
20
21                image.setImageResource(R.drawable.screen);
22                getWindow().setFlags(WindowManager.LayoutParams.
23                    FLAG_FULLSCREEN,
24                    WindowManager.LayoutParams.FLAG_FULLSCREEN)
25                    ;
26                getSupportActionBar().hide();
27
28                image.getLayoutParams().height = ViewGroup.
29                    LayoutParams.MATCH_PARENT;
30                image.getLayoutParams().width = ViewGroup.
31                    LayoutParams.MATCH_PARENT;
32                image.setAdjustViewBounds(false);
33                image.setScaleType(ImageView.ScaleType.FIT_XY);
34
35                touchCount = 0;
36            }
37            lasttime = currentTime;
38        }
39        return false;
40    }
```

2.7.5 Make ‘Register’ button disabled after user inputs their username and password

In making the registration button on the login screen unusable, my role was to develop code to detect when the user input their credentials. However i needed to also disable the registration button when both username and password were filled at the same time. To do this I crafted an on text change listener that always checked both fields.

```
1
2
3 public void enableTextListenier()
4 {
5
6     Log.d(TAG,"in listener!");
7
8     final EditText username_et = findViewById(R.id.
9         login_username);
10    final EditText password_et = findViewById(R.id.
11        login_password);
12    final Button regButton = findViewById(R.id.register_enter);
13
14    //final boolean reg_btn_allow = false;
15
16    username_et.addTextChangedListener(new TextWatcher() {
17
18        @Override
19        public void onTextChanged(CharSequence s, int start,
20            int before,
21            int count) {
22            // TODO Auto-generated method stub
23
24            if (s.toString().equals("") && password_et.getText
25                ().toString().equals("")) {
26                Log.d(TAG,"TEXT IS EMPTY!");
27                // reg_btn_allow = true;
28                regButton.setVisibility(View.VISIBLE);
29            } else {
30                Log.d(TAG,"TEXT IS something!");
31                //reg_btn_allow = false;
32                regButton.setVisibility(View.INVISIBLE);
33            }
34        }
35
36        @Override
37        public void beforeTextChanged(CharSequence s, int start
38            , int count,
39            int after) {
40            // TODO Auto-generated method stub
41
42        }
43
44        @Override
45        public void afterTextChanged(Editable s) {
46            // TODO Auto-generated method stub
47
48        }
49    });
50 }
```

```

44     });
45
46     password_et.addTextChangedListener(new TextWatcher() {
47
48         @Override
49         public void onTextChanged(CharSequence s, int start,
50             int before,
51                 int count) {
52             // TODO Auto-generated method stub
53
54             if (s.toString().equals("") && username_et.getText
55                 ().toString().equals("")) {
56                 Log.d(TAG, "TEXT IS EMPTY!");
57                 // reg_btn_allow = true;
58                 regButton.setVisibility(View.VISIBLE);
59             } else {
60                 Log.d(TAG, "TEXT IS something!");
61                 // reg_btn_allow = false;
62                 regButton.setVisibility(View.INVISIBLE);
63             }
64         }
65
66         @Override
67         public void beforeTextChanged(CharSequence s, int start
68             , int count,
69                 int after) {
70             // TODO Auto-generated method stub
71         }
72
73         @Override
74         public void afterTextChanged(Editable s) {
75             // TODO Auto-generated method stub
76         }
77     });
78 }

```

2.7.6 Changing the user password requires verifying current password

Much of the backend work for this requirement I had already personally developed, so it didn't take much more work to verify the user's credentials in order to grant them access to change their password. In the code below I did this along with other field verification for current username, current password, new password, and confirm new password.

```
1
2
3 public void handlePassChangeAttempt(View view)
4 {
5     EditText userText = (EditText)findViewById(R.id.
6         passchg_username);
7     EditText oldPassText = (EditText)findViewById(R.id.
8         passchg_prevpass);
9     EditText newPassText = (EditText)findViewById(R.id.
10        passchg_newpass);
11     EditText newPassConfirmText = (EditText)findViewById(R.id.
12        passchg_newpass_confirm);
13     String username = userText.getText().toString();
14     String oldPassword = oldPassText.getText().toString();
15     String newPassword = newPassText.getText().toString();
16     String newPasswordConfirm = newPassConfirmText.getText().
17         toString();
18
19     // gain access to filesystem write priv first
20     if(checkAppPermissions()) {
21         requestAppPermissions();
22
23         userText.setText("");
24         oldPassText.setText("");
25         newPassText.setText("");
26         newPassConfirmText.setText("");
27     }
28     // handle pass change attempt
29     else
30     {
31         Document doc = getXmlDoc();
32
33         if(usernameExists(username,doc) && credentialsMatch(
34             username,oldPassword,doc))
35         {
36             if(newPassword.equals(newPasswordConfirm))
37             {
38                 changeUserPassword(username,newPassword,doc);
39                 onBackPressed();
40             }
41             else
42             {
43                 userText.setText("");
44                 oldPassText.setText("");
45                 newPassText.setText("");
46                 newPassConfirmText.setText("");
47             }
48         }
49     }
50 }
```



```

43         Toast.makeText(getApplicationContext(), "New
           password does not match confirmation",
           Toast.LENGTH_SHORT).show();
44     }
45
46 }
47 else
48 {
49     if(!usernameExists(username, doc))
50         Toast.makeText(getApplicationContext(), "User
           does not exist", Toast.LENGTH_SHORT).show()
           ;
51     else
52         Toast.makeText(getApplicationContext(), "Old
           password does not match", Toast.
           LENGTH_SHORT).show();
53
54     userText.setText("");
55     oldPassText.setText("");
56     newPassText.setText("");
57     newPassConfirmText.setText("");
58 }
59
60 }
61
62 }

```

3 Android OS Customization

After we had completed our work on developing a "Spy App" for an android device. We were then tasked with compiling/installing/and modifying Lineage OS code for our tablet. The compilation and setup took place on my machine, and the modification of the boot animation was performed with the help of my partner.

3.1 Android Source Code Download Setup

To start, I needed to download the source for Lineage OS on my Linux 16.04 virtual machine. Using `repo sync` I was able to do this but ended up missing proprietary blobs when I went to compile. So to fix this, I used a repo on GitHub known as TheMuppets located here.

https://github.com/TheMuppets/proprietary_vendor_asus/tree/lineage-16.0/flo

After acquiring the repo I was able to re sync the source code with it after editing the file `.repo/local_manifests/roomservice.xml` with the correct blobs repo.

```
1 <project name="TheMuppets/proprietary_vendor_asus" path="vendor/
  asus" depth="1" />
```

Also running `breakfast flo` again resolved any errors about missing blob files.

3.2 Compile Android Source Code

Now that all the necessary code has been collected for compiling, I was able to compile without any errors regarding missing sources. However new errors were experienced regarding my virtual machines' memory parameters, as well as environmental variables for the jack server. The jack server is an android toolchain needed to compile the Java source code for our target OS and in my case it didn't have enough memory to continue. By allocating it more memory via setting the correct env variables I was able to get the image to finally compile. This just required me updating my `bashrc` file for my current user.

```
1 export ANDROID_JACK_VM_ARGS="-Dfile.encoding=UTF-8 -XX:+
  TieredCompilation -Xmx4G"
2 export JACK_SERVER_VM_ARGUMENTS="-Dfile.encoding=UTF-8 -XX:+
  TieredCompilation -Xmx4g"
3 USE_CCACHE=1
4 ccache -M 50G
```

Also I would need to restart the jack server to apply the changes.

```
1 jack-admin kill-server
2 jack-admin start-server
```

3.3 Custom Boot Animation

With the final Lineage OS image in hand, I could now install it on the tablet to test it. To complete the task however, we had to change the boot animation on the device. With help of my partner, he supplied me with a customized boot animation he had modified to get sound and I was able to replace within the lineage OS directory..

```
android/lineage/out/target/product/flo/system/media/bootanimation.zip
```

Afterwards I rebuilt the image and installed it through the recovery menu as well as tested. Everything worked fine and our task had been completed.