

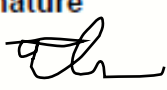
## Assignment/Project Coversheet

<b>Campus</b>	Durbanville		
<b>Faculty</b>	Information Technology		
<b>Module code</b>	ITJA321	<b>Group number</b>	3A
<b>Student surname</b>	Bekker		
<b>Student name</b>	Johann		
<b>eVision number</b>		<b>Student number</b>	Y.JPCWYF29

Turnitin report attached:      Yes   ☐      No   ☐

### Declaration

"I declare that this assignment is my own original work except for source material explicitly acknowledged, and that the same or related material has not been previously, or is being simultaneously, submitted for this or any other course. I also acknowledge that I am aware of the Institution's policy and regulations on honesty in academic work as set out in the Pearson Institute of Higher Education Conditions of Enrolment, and of the disciplinary guidelines applicable to breaches of such policy and regulations."

<b>Student signature</b>

<b>Date</b> 02/10/2020

### Marker Comments

---



---



---



---

<b>Marker name</b>		<b>Internal marker result</b>	%
<b>Marker name</b>		<b>External marker result</b>	%

## Contents

<b>Question 1 .....</b>	<b>2</b>
1.1 .....	2
1.2 .....	3
<b>Question 2 .....</b>	<b>4</b>
<b>Login Activity.....</b>	<b>4</b>
activity_login.xml file .....	8
Login Activity User Interface (Screenshots).....	10
<b>Registration Activity.....</b>	<b>14</b>
activity_registration.xml file .....	20
Registration Activity User Interface (Screenshots) .....	23
<b>SQLite Database .....</b>	<b>27</b>
SQLiteOpenHelper Class.....	27
Bank User Class .....	31
<b>Main Page Activity.....</b>	<b>33</b>
activity_main_page.xml file .....	36
Main Page Activity User Interface (Screenshots) .....	38
<b>View Account Balance Activity.....</b>	<b>41</b>
activity_view_account_balance.xml file .....	44
View Account Balance Activity User Interface (Screenshots) .....	46
<b>Transfer Between Accounts Activity.....</b>	<b>47</b>
activity_transfer_between_accounts.xml file .....	53
custom_spinner_transfer_options.xml file .....	56
Transfer Between Accounts Activity User Interface (Screenshots) .....	57
<b>Additional Code .....</b>	<b>60</b>
AndroidManifest.xml file .....	60
<b>References .....</b>	<b>61</b>

## Question 1

### 1.1

An Android intent can be described as a passive data structure or object that carries a message or description of an operation that is to be performed (Izuchukwu, 2017). This message is sent between the various core application components (Izuchukwu, 2017). Thus, it can communicate operations from a single core component to another (Izuchukwu, 2017). These core components include broadcast receivers, services and activities (Izuchukwu, 2017).

**Intents have several uses in Android development. Intents can be used to:** start an activity, start a service, deliver a broadcast or even transfer data to another component. There are many more uses.

An activity can be described as a representation of one screen in an application (Vogel, 2020). A new instance of an activity can be started by using the `startActivity()` method (Vogel, 2020). The activity to be started is described by the intent (Vogel, 2020). Also, the intent holds any data that may be required and is passed within this method (Vogel, 2020).

A service can be described as a component that runs and executes operations in the background. This component lacks an interface (Vogel, 2020). To start a service, the `startService()` method can be used to start the service component and to perform an operation such as uploading a file (Vogel, 2020). The intent is passed and defines the service to be started (Vogel, 2020). Also, the intent holds any data that may be required by the service (Vogel, 2020).

Broadcasts are messages that can be received by any application (Vogel, 2020). Several broadcasts are delivered by the system for events. For example, when the system boots up, the system delivers a broadcast (Vogel, 2020). This broadcast can be read by an application to see that the system has booted up (Vogel, 2020). A broadcast can be sent/delivered to an application by using either the `sendOrderedBroadcast()` method or the `sendBroadcast()` method (Vogel, 2020).

To transfer data from one component to another, an instance of an intent should be used (Vogel, 2020). `putExtra()` methods can then be used to add data to the intent instance (Vogel, 2020). Thereafter, the intent can be passed to the desired component (Vogel, 2020). To access the data in the receiving component, an instance of an intent should be created (Vogel, 2020). Methods such as `getIntent()`, `getStringExtra()`, `getAction()` and `getData()` can then be used to access the intent and its data depending on the type of data/information received (Vogel, 2020).

**In Android development, there are two types of intents:** explicit intents and implicit intents.

Explicit intents are used when you define which of the various components should be opened by the Android System (Izuchukwu, 2017). Normally, explicit intents are used to start Android components in an application (Izuchukwu, 2017). For example, `startService()` can be used to explicitly start the service component.

Implicit intents are used when you define the action you would like to perform without defining which of the components will perform the action (Izuchukwu, 2017). Therefore, implicit intents declare general actions (Izuchukwu, 2017). These general actions can be used or handled by any of the components from the application (Izuchukwu, 2017). Components from other applications can also use or handle these general actions (Izuchukwu, 2017). For example, one application can use implicit intents to pass coordinates to another application.

## 1.2

### **SQLiteOpenHelper class:**

The SQLiteOpenHelper class is located within the android.database.sqlite package (Khan, 2015). This class can be used to create, open and manage an SQLite database and tables (Khan, 2015). Also, the SQLite database is only opened or created when either the getReadableDatabase() or getWritableDatabase() method is called (Khan, 2015). To use the SQLiteOpenHelper class, a class should be created and extended using 'extends SQLiteOpenHelper' (Khan, 2015). This will give the extended class the ability to implement the various SQLiteOpenHelper CRUD (Create, Read, Update and Delete) methods (Khan, 2015). These methods can then be used to create, read, update and delete an SQLite database and tables (Khan, 2015).

### **SQLiteCursor class:**

The SQLiteCursor class can be used to invoke methods that execute SQLite statements and it can retrieve data from a query on a SQLiteDatabase (Stroud, 2016). Therefore, the SQLiteCursor class can be used to perform various CRUD operations on the actual data within an SQLite database (Stroud, 2016). To use the SQLiteCursor class, an instance of this class should be created (Stroud, 2016). This instance can then be used to invoke the various methods that execute SQLite statements (Stroud, 2016).

To conclude, the SQLiteOpenHelper class is used to manage the structure of the database and tables. The SQLiteCursor class is used to manage the actual data within the database and tables.

## Question 2

### Login Activity

```
/*
 * This is the login activity. It allows the user to login to their accounts on the
 * SQLite database. The user logs in using user details.
 */

package com.example.bankingapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class LoginActivity extends AppCompatActivity {

    // Local variables
    EditText emailEdit, passwordEdit;
    Button loginButton;
    TextView registerHereTextView;
    DatabaseOpenHelper db;
    String emailEntered, passwordEntered;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        // Creates a new instance of the SQLiteOpenHelper class
        db = new DatabaseOpenHelper(this);

        emailEdit = (EditText)findViewById(R.id.editText_email);
        passwordEdit = (EditText)findViewById(R.id.editText_password);
        loginButton = (Button)findViewById(R.id.button_loginBTN);
        registerHereTextView = (TextView) findViewById(R.id.textView_register_here);

        /*
        An action listener is placed on the registerHereTextView. Once clicked, it directs the
        user to the registration screen using an intent.
        */
        registerHereTextView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent registrationIntent =
                    new Intent(LoginActivity.this, RegistrationActivity.class);
                startActivity(registrationIntent);
            }
        })
    }
}
```

```

});

// Adds an action listener to the 'LOGIN' button
loginButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // The initialise() method is called to initialise variables with user input
        initialise();

        // The validation() method is called to validate user inputs
        // If the validation() method returns 'true' - all inputs are valid
        if (validation()) {
            /*
             The authenticateUser() method in the OpenDatabaseHelper class is called. This
             method checks to see if the entered email and password is found on the database.
            */
            String result = db.authenticateUser(emailEntered, passwordEntered);

            // If the email and password is found on the database
            if (result == "UserFound") {

                // Direct user to the main application interface using an intent
                Intent mainScreen =
                    new Intent(LoginActivity.this, MainPageActivity.class);

                // The email entered is added to the intent and passed to the MainPageActivity
                mainScreen.putExtra("loggedInUserEmail", emailEntered);
                startActivity(mainScreen);
                // Finish() exits application if user is on main screen and back button is pressed
                finish();
                // Show toast message to user indicating that the login attempt is successful
                Toast.makeText(LoginActivity.this,
                    "Login successful", Toast.LENGTH_SHORT).show();

                // If the entered is incorrect
            } else if (result == "PasswordIncorrect"){
                // Set error on text field
                passwordEdit.setError("Incorrect password");
                // Show toast message indicating that the password entered is incorrect
                Toast.makeText(LoginActivity.this,
                    "Password is incorrect", Toast.LENGTH_SHORT).show();

                // If the email entered is not found on the database
            } else{
                // Set error on text field
                emailEdit.setError("Email does not exist");
                // Show toast message indicating that the email entered is incorrect
                Toast.makeText(LoginActivity.this,
                    "Email is incorrect", Toast.LENGTH_SHORT).show();
            }
        }
    }
});
}

```

```

// This method is used to initialise the local variables
public void initialise(){
    emailEntered = emailEdit.getText().toString().trim();
    passwordEntered = passwordEdit.getText().toString().trim();
}

/*
This validation() method is used to perform validation on all user inputs. It returns 'true' if
all inputs are valid.
*/
public boolean validation(){
    boolean validInput = true;

    // If all text fields are empty
    if(emailEntered.isEmpty() && passwordEntered.isEmpty()){
        // Show toast message prompting the user to not leave the fields empty
        Toast.makeText(LoginActivity.this,
            "Enter an email address and password",Toast.LENGTH_SHORT).show();
        validInput = false;

        // If all of the text fields are not left empty
    }else{

        // If the email text field is left empty
        if(emailEntered.isEmpty()){
            // Show toast message prompting the user to enter an email
            Toast.makeText(LoginActivity.this,
                "Enter an email address",Toast.LENGTH_SHORT).show();
            validInput = false;
        }

        // If the email text field is not empty and the input does not match the regex pattern
        if(!emailEntered.isEmpty() && !emailEntered.matches("[a-z0-9._-]+@[a-z]+\.[a-z]+")){
            // Show toast message prompting the user to enter a valid email
            Toast.makeText(LoginActivity.this,
                "Enter a valid email address",Toast.LENGTH_SHORT).show();
            validInput = false;
        }

        // If the password text field is left empty
        if(passwordEntered.isEmpty()){
            // Show toast message prompting the user to enter a password
            Toast.makeText(LoginActivity.this,
                "Enter a password",Toast.LENGTH_SHORT).show();
            validInput = false;
        }

        // If the entered password is less than 5 characters long
        if(!passwordEntered.isEmpty() && passwordEntered.length()<5){
            // Show toast message prompting the user to enter a longer password
            Toast.makeText(LoginActivity.this,
                "Password too short",Toast.LENGTH_SHORT).show();
            validInput = false;
        }
    }
}
// Return validation result

```

```
        return validInput;
    }
}
```



## activity\_login.xml file

```
<?xml version="1.0" encoding="utf-8"?>

<!--This file is used to set the design and layout of the login screen-->

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    tools:context=".LoginActivity">

    <!--Logo picture-->
    <ImageView
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:layout_marginTop="10dp"
        app:srcCompat="@drawable/logo" />

    <!--Greeting/heading on login screen-->
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Welcome to the Sisonke Bank App"
            android:textColor="#156775"
            android:textSize="18dp" />
    </LinearLayout>

    <!--Email text field-->
    <EditText
        android:id="@+id/editText_email"
        android:layout_width="280dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="25dp"
        android:hint="@string/email" />

    <!--Password text field-->
    <!--Password is hidden while it is being entered-->
    <EditText
        android:id="@+id/editText_password"
        android:layout_width="280dp"
        android:layout_height="wrap_content"
```

```

        android:inputType="textPassword"
        android:hint="@string/password" />

<!--Login button-->
<Button
    android:id="@+id/button_loginBTN"
    android:layout_width="280dp"
    android:layout_height="50dp"
    android:layout_marginTop="20dp"
    android:background="#156775"
    android:text="@string/loginBTN"
    android:textColor="#ffffff" />

<!--Asks user if he/she has an account-->
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:orientation="horizontal">

    <!--Question section-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14dp"
        android:text="@string/no_account"
        android:textColor="#000000" />

    <!--'Register Here' section-->
    <TextView
        android:id="@+id/textView_register_here"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14dp"
        android:layout_marginLeft="5dp"
        android:text="@string/register_here"
        android:textColor="#000000" />
</LinearLayout>

</LinearLayout>

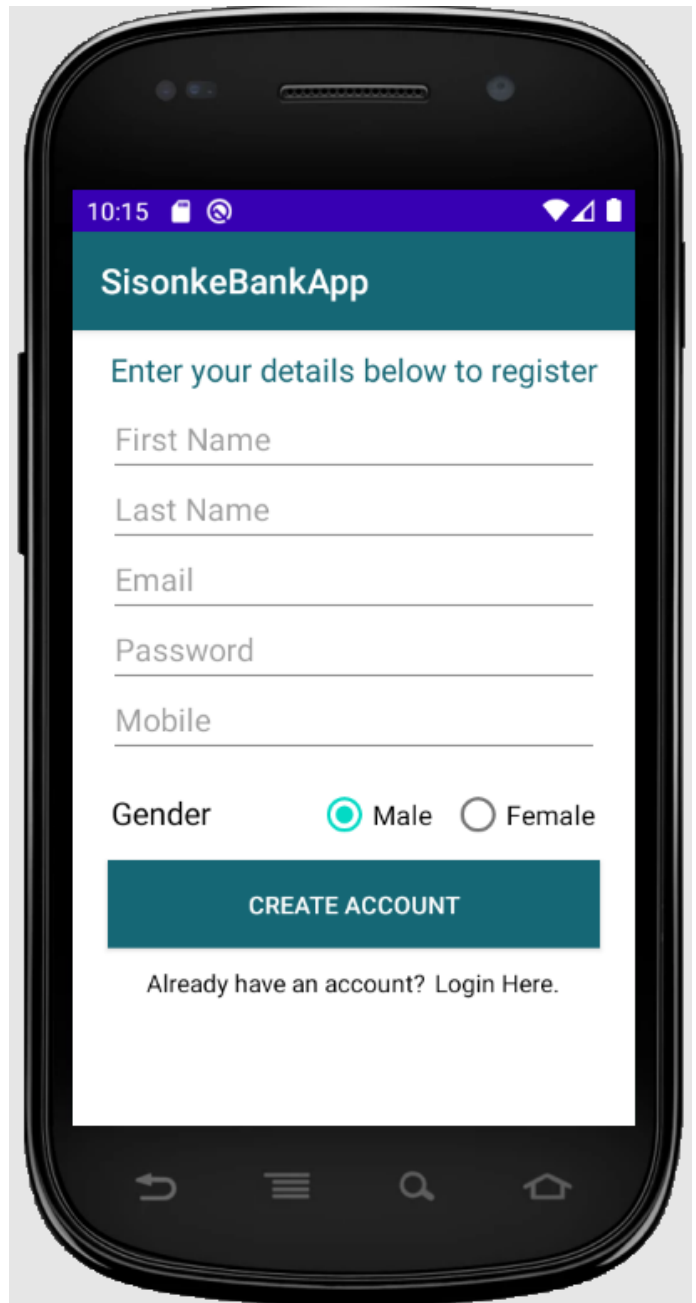
```

## Login Activity User Interface (Screenshots)

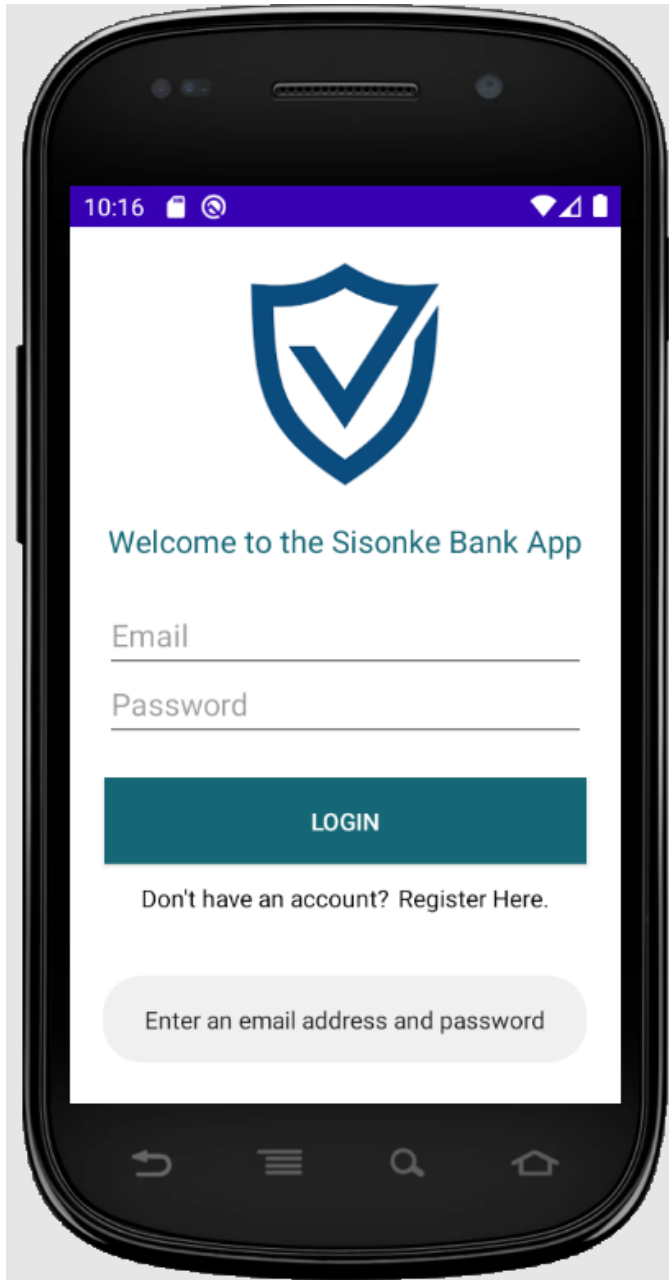
Below: The user interface of the Login Activity. When a user clicks on 'Register Here', he/she will be redirected to the registration page. This is done using intents.



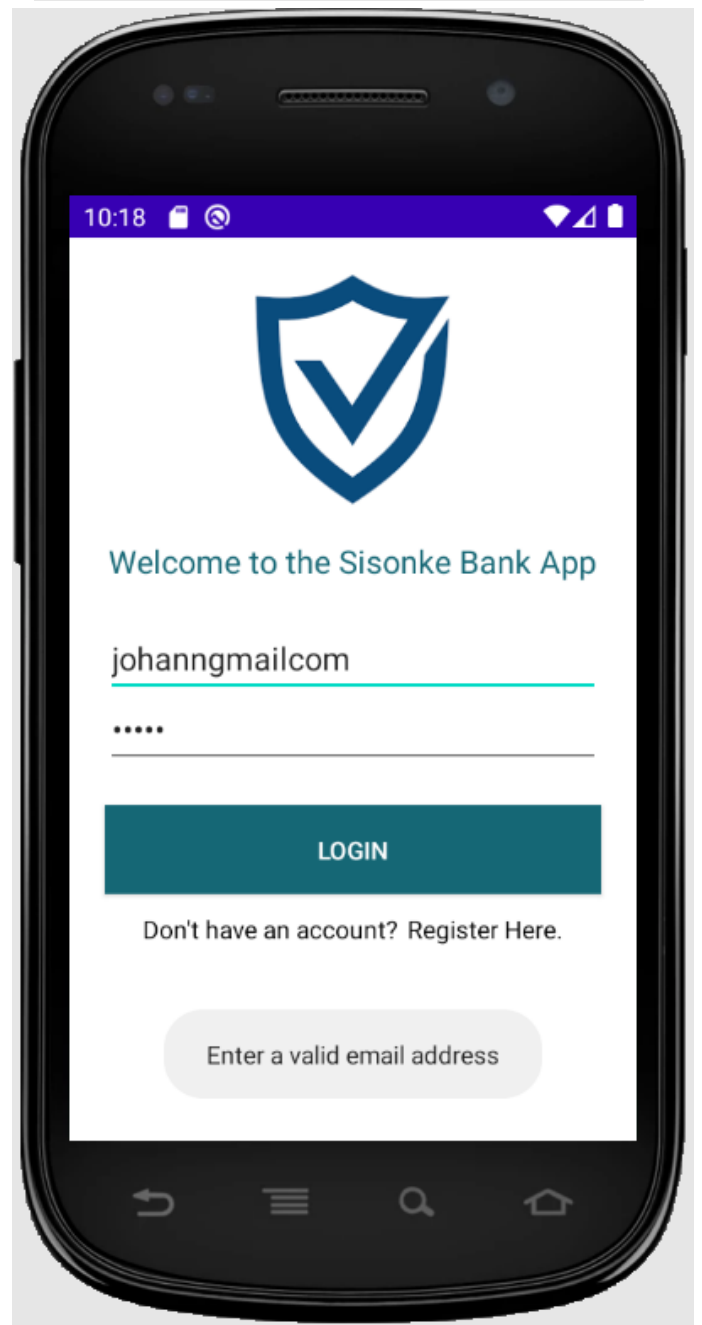
Below: The user interface of the Registration Activity.



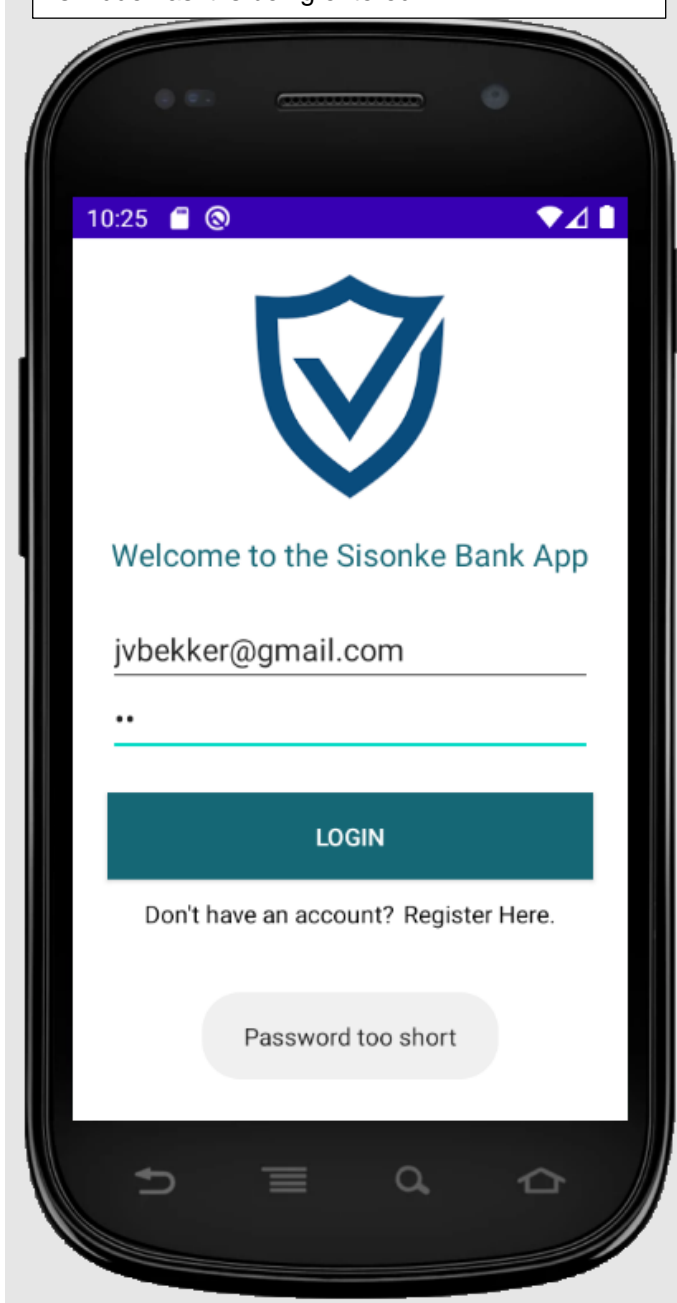
Below: Validation is used to ensure that the user must enter an email and password before logging in. If a user does not enter these details before logging in, a toast message is shown. This toast message asks the user to enter an email and password.



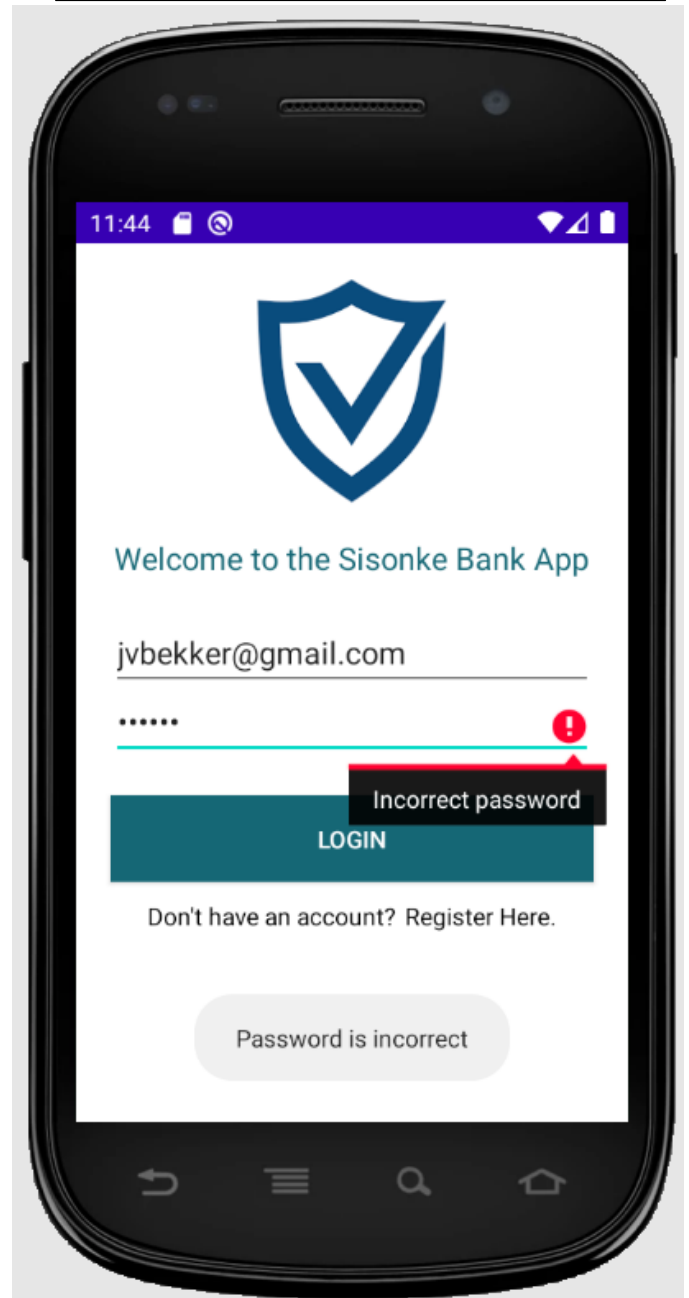
Below: Validation is used to ensure that the entered email has the correct format. If the email does not have the correct format and the user tries to log in, a toast message is shown. This toast message asks the user to enter a valid email.



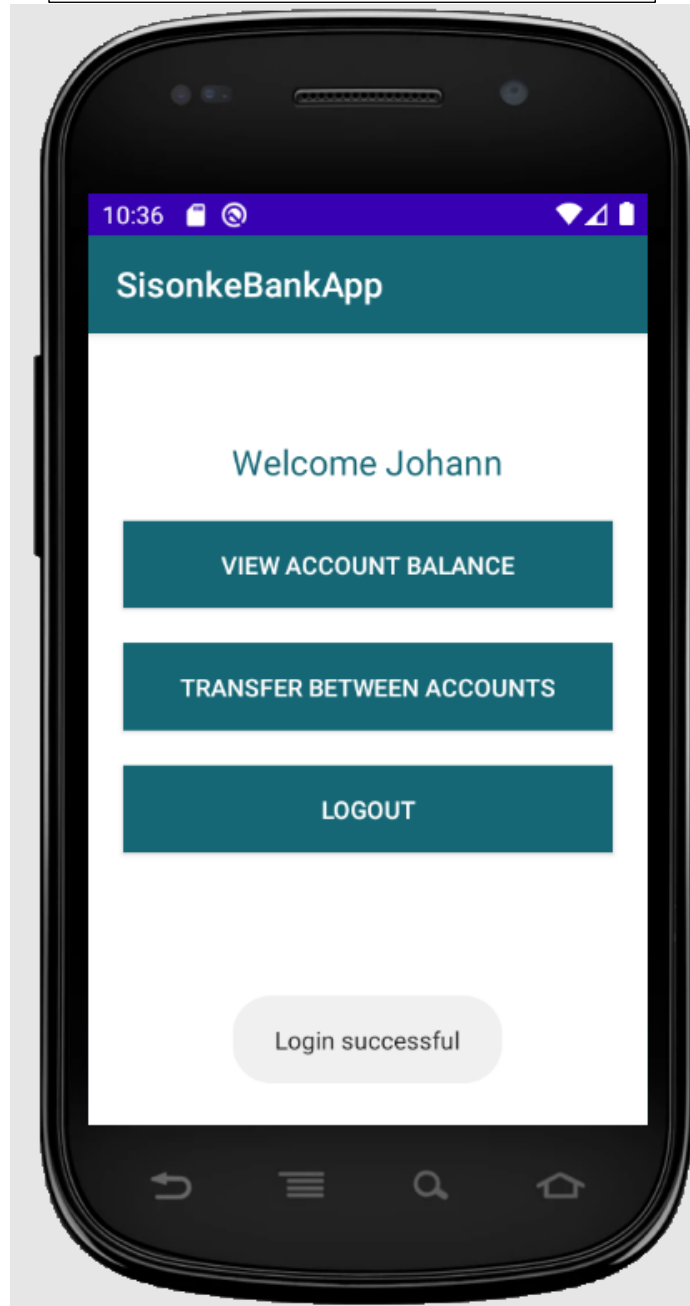
Below: Validation is used to ensure that the entered password is at least 5 characters long. If the entered password is shorter than 5 characters and the user tries to log in, a toast message is shown. This toast message tells the user that the entered password is too short. Also, the password is hidden as it is being entered.



Below: Validation is used to ensure that the entered email and password is correct. If either the email or password is incorrect, a toast message will appear to inform the user. In this example, this password entered is incorrect.



Below: After a successful login, the user is directed to the main user interface. Also, a toast message appears which informs the user that the login was successful.



## Registration Activity

```
/*
 * This is the registration activity. It allows the user to register on the
 * SQLite database.
 */

package com.example.bankingapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

public class RegistrationActivity extends AppCompatActivity {

    // Local variables
    DatabaseOpenHelper db;
    EditText firstNameEdit, lastNameEdit, emailEdit, passwordEdit, mobileEdit;
    String firstNameEntered, lastNameEntered, emailEntered, passwordEntered, mobileEntered,
    genderSelected;
    double currentAccountBalance, savingsAccountBalance;
    Button createAccountButton;
    TextView loginHereTextView;
    RadioButton genderButton;
    RadioGroup radioGroup;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registration);

        // Sets the title for the actionbar
        getSupportActionBar().setTitle("SisonkeBankApp");

        // Creates a new instance of the SQLiteOpenHelper class
        db = new DatabaseOpenHelper(this);

        // Text fields used to capture user input on registration screen
        firstNameEdit = (EditText) findViewById(R.id.firstNameT);
        lastNameEdit = (EditText) findViewById(R.id.lastNameT);
        emailEdit = (EditText) findViewById(R.id.emailT);
        passwordEdit = (EditText) findViewById(R.id.passwordT);
        mobileEdit = (EditText) findViewById(R.id.mobileT);
        radioGroup = (RadioGroup) findViewById(R.id.radioGroup);
        createAccountButton = (Button) findViewById(R.id.createAccountBTN);
```

```

// Adds an action listener to the 'CREATE ACCOUNT' button
createAccountButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // The initialise() method is called to initialise variables with user input
        initialise();

        // The validation() method is called to validate user inputs
        // If the validation() method returns 'true' - all inputs are valid
        if (validation()) {

            /*
            The checkIfEmailAlreadyUsed() method is called to determine if the entered
            user email already exists on the database or not. This method will call the
            registerUser() method to add the account to the database if the entered email
            does not exist.
            */
            checkIfEmailAlreadyUsed();
        }
    }
});

/*
An action listener is placed on the loginHereTextView. Once clicked, it directs the
user to the login screen using an intent.
*/
loginHereTextView = (TextView) findViewById(R.id.textView_login_here);
loginHereTextView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent loginIntent =
            new Intent(RegistrationActivity.this, LoginActivity.class);
        startActivity(loginIntent);
    }
});
}

// This method is used to get the selected gender using radio buttons
public void checkButton(View v){
    int id = radioGroup.getCheckedRadioButtonId();
    genderButton = findViewById(id);
    genderSelected = genderButton.getText().toString();
}

/*
The registerUser() method registers a new user on the database. It uses the BankUser class
and its getter and setter methods to do so.
*/
public void registerUser() {

    /*
    Creates a new instance of the BankUser class called 'user'. This instance is given the
    values inserted by the user.
    */
    BankUser user = new BankUser(firstNameEntered, lastNameEntered,

```



```

        emailEntered, passwordEntered, mobileEntered, genderSelected,
        currentAccountBalance, savingsAccountBalance);

// Adds the user to the database
boolean dataInserted = db.addUser(user);

// Setter methods used in BankUser class
user.setName(firstNameEntered);
user.setSurname(lastNameEntered);
user.setEmail(emailEntered);
user.setPassword(passwordEntered);
user.setMobile(mobileEntered);
user.setGender(genderSelected);
user.setCurrentAccountBalance(currentAccountBalance);
user.setSavingsAccountBalance(savingsAccountBalance);

// If the user is added to the database
if (dataInserted == true) {
    // Show toast message to user indicating that the account creation was successful
    Toast.makeText(RegistrationActivity.this, "Account creation successful",
        Toast.LENGTH_LONG).show();
    // Direct user to login screen using an intent
    Intent goToLoginScreen =
        new Intent(RegistrationActivity.this, LoginActivity.class);
    startActivity(goToLoginScreen);

// If the user is not added to the database
} else {
    // Show toast message indicating that the account creation was not successful
    Toast.makeText(RegistrationActivity.this, "Account not created",
        Toast.LENGTH_LONG).show();
}
}

// This method is used to initialise the local variables
public void initialise(){

    // User text field inputs
    firstNameEntered = firstNameEdit.getText().toString().trim();
    lastNameEntered = lastNameEdit.getText().toString().trim();
    emailEntered = emailEdit.getText().toString().trim();
    passwordEntered = passwordEdit.getText().toString().trim();
    mobileEntered = mobileEdit.getText().toString().trim();

    // Radio button selected
    int id = radioGroup.getCheckedRadioButtonId();
    genderButton = findViewById(id);
    genderSelected = genderButton.getText().toString();

    // Variables storing account balances
    currentAccountBalance = 50400.0;
    savingsAccountBalance = 121543.0;
}

```

```

/*
This validation() method is used to perform validation on all user inputs. It returns 'true' if
all inputs are valid.
*/
public boolean validation(){
    boolean validInput = true;

    // If all text fields are empty
    if(firstNameEntered.isEmpty() && lastNameEntered.isEmpty() && emailEntered.isEmpty()
        && passwordEntered.isEmpty() && mobileEntered.isEmpty()){

        // Set errors on fields
        firstNameEdit.setError("Enter first name");
        lastNameEdit.setError("Enter last name");
        emailEdit.setError("Enter email");
        passwordEdit.setError("Enter password");
        mobileEdit.setError("Enter mobile number");

        // Show toast message prompting the user to not leave the fields empty
        Toast.makeText(RegistrationActivity.this,"Enter the required details",
            Toast.LENGTH_SHORT).show();
        validInput = false;

        // If all of the text fields are not left empty
    }else {

        // If the first name text field is left empty
        if (firstNameEntered.isEmpty()) {
            // Set error on text field
            firstNameEdit.setError("Enter first name");
            // Show toast message prompting the user to enter a first name
            Toast.makeText(RegistrationActivity.this,"Enter your first name",
                Toast.LENGTH_SHORT).show();
            validInput = false;
        }

        // If the last name text field is left empty
        if (lastNameEntered.isEmpty()) {
            // Set error on text field
            lastNameEdit.setError("Enter last name");
            // Show toast message prompting the user to enter a last name
            Toast.makeText(RegistrationActivity.this,"Enter your last name",
                Toast.LENGTH_SHORT).show();
            validInput = false;
        }

        // If the email text field is left empty
        if (emailEntered.isEmpty()) {
            // Set error on text field
            emailEdit.setError("Enter email");
            // Show toast message prompting the user to enter an email
            Toast.makeText(RegistrationActivity.this,"Enter your email address",
                Toast.LENGTH_SHORT).show();
            validInput = false;
        }
    }
}

```

```

// If the email text field is not empty and the input does not match the regex pattern
if (!emailEntered.isEmpty() && !emailEntered.matches("[a-z0-9._-]+@[a-z]+\\.+[a-z]+")) {
    // Set error on text field
    emailEdit.setError("Enter valid email address");
    // Show toast message prompting the user to enter a valid email
    Toast.makeText(RegistrationActivity.this, "Enter a valid email address",
        Toast.LENGTH_SHORT).show();
    validInput = false;
}

// If the password text field is left empty
if (passwordEntered.isEmpty()) {
    // Set error on text field
    passwordEdit.setError("Enter password");
    // Show toast message prompting the user to enter a password
    Toast.makeText(RegistrationActivity.this, "Enter a password",
        Toast.LENGTH_SHORT).show();
    validInput = false;
}

// If the entered password is less than 5 characters long
if (!passwordEntered.isEmpty() && passwordEntered.length() < 5) {
    // Set error on text field
    passwordEdit.setError("Password too short");
    // Show toast message prompting the user to enter a longer password
    Toast.makeText(RegistrationActivity.this,
        "Password must contain at least 5 characters", Toast.LENGTH_SHORT).show();
    validInput = false;
}

// If no mobile number is entered
if (mobileEntered.isEmpty()) {
    // Set error on text field
    mobileEdit.setError("Enter mobile number");
    // Show toast message prompting the user to enter a mobile number
    Toast.makeText(RegistrationActivity.this, "Enter your mobile number",
        Toast.LENGTH_SHORT).show();
    validInput = false;
}

/*
If a mobile number is entered that does not start with 0 or is 10 digits long
*/
if (!mobileEntered.isEmpty() && !mobileEntered.matches("^0\\d{9}$")) {
    // Set error on text field
    mobileEdit.setError("Invalid mobile number");
    // Show toast message prompting the user to enter a valid mobile number
    Toast.makeText(RegistrationActivity.this, "Enter a valid mobile number",
        Toast.LENGTH_SHORT).show();
    validInput = false;
}
}

// Return validation result
return validInput;

```

```

}

/*
The checkIfEmailAlreadyUsed() method is used to determine if the entered
user email already exists on the database or not. This method will call the
registerUser() method to add the account to the database if the entered email
does not exist.
*/
public void checkIfEmailAlreadyUsed(){

    /*
    The checkIfEmailExists() method in the DatabaseOpenHelper class is used
    to determine whether the entered email already exists on the database or not. If the
    method returns 'false', the new account is created.
    */
    Boolean result = db.checkIfEmailExists(emailEntered);

    // If the email does not exist on the database
    if(result == false){
        // Create the account and register the user
        registerUser();

        // If the email entered is already used
    }else{
        // Set error on text field
        emailEdit.setError("This email is already used");
        // Show toast message prompting the user to enter a different email
        Toast.makeText(RegistrationActivity.this,"Enter a different email",
            Toast.LENGTH_SHORT).show();
    }
}
}
}

```

## activity\_registration.xml file

```
<?xml version="1.0" encoding="utf-8"?>

<!--This file is used to set the design and layout of the registration screen-->

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".RegistrationActivity"
    android:orientation="vertical"
    android:gravity="center_horizontal"
    android:background="#ffffff">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:textSize="18dp"
            android:layout_height="wrap_content"
            android:textColor="#156775"
            android:text="Enter your details below to register"/>
    </LinearLayout>

    <!--First name text field-->
    <EditText
        android:id="@+id/firstNameT"
        android:layout_width="280dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:hint="@string/first_nameR" />

    <!--Last name text field-->
    <EditText
        android:id="@+id/lastNameT"
        android:layout_width="280dp"
        android:layout_height="wrap_content"
        android:hint="@string/last_nameR"/>

    <!--Email text field-->
    <EditText
        android:id="@+id/emailT"
        android:layout_width="280dp"
        android:layout_height="wrap_content"
        android:hint="@string/emailR"/>

    <!--Password text field-->
    <!--The password is hidden when it is entered-->
    <EditText
        android:id="@+id/passwordT"
        android:layout_width="280dp"
        android:layout_height="wrap_content"
```

```

        android:inputType="textPassword"
        android:hint="@string/passwordR"/>

<!--Mobile number text field-->
<!--Only digits can be entered-->
<EditText
    android:id="@+id/mobileT"
    android:layout_width="280dp"
    android:layout_height="wrap_content"
    android:digits="0123456789"
    android:hint="@string/mobileR"/>

<!--Gender radio buttons-->
<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:id="@+id/radioGroup">

    <TextView
        android:layout_width="wrap_content"
        android:textColor="#000000"
        android:layout_height="25dp"
        android:textSize="18dp"
        android:layout_marginTop="15dp"
        android:text="Gender"/>

    <!--Male radio button-->
    <RadioButton
        android:id="@+id/radioMale"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Male"
        android:layout_marginTop="15dp"
        android:layout_marginLeft="60dp"
        android:checked="true"
        android:onClick="checkButton"
        android:textSize="15dp" />

    <!--Female radio button-->
    <RadioButton
        android:id="@+id/radioFemale"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Female"
        android:layout_marginTop="15dp"
        android:layout_marginLeft="10dp"
        android:onClick="checkButton"
        android:textSize="15dp" />
</RadioGroup>

<!--Create account button-->
<Button
    android:id="@+id/createAccountBTN"
    android:layout_width="280dp"
    android:layout_height="50dp"

```

```

        android:background="#156775"
        android:textColor="#ffffff"
        android:layout_marginTop="10dp"
        android:text="@string/create_accountBTNtextR" />

<!--Asks user if he/she already has an account-->
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:orientation="horizontal">

    <!--Question section-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:textSize="14dp"
        android:text="Already have an account?"/>

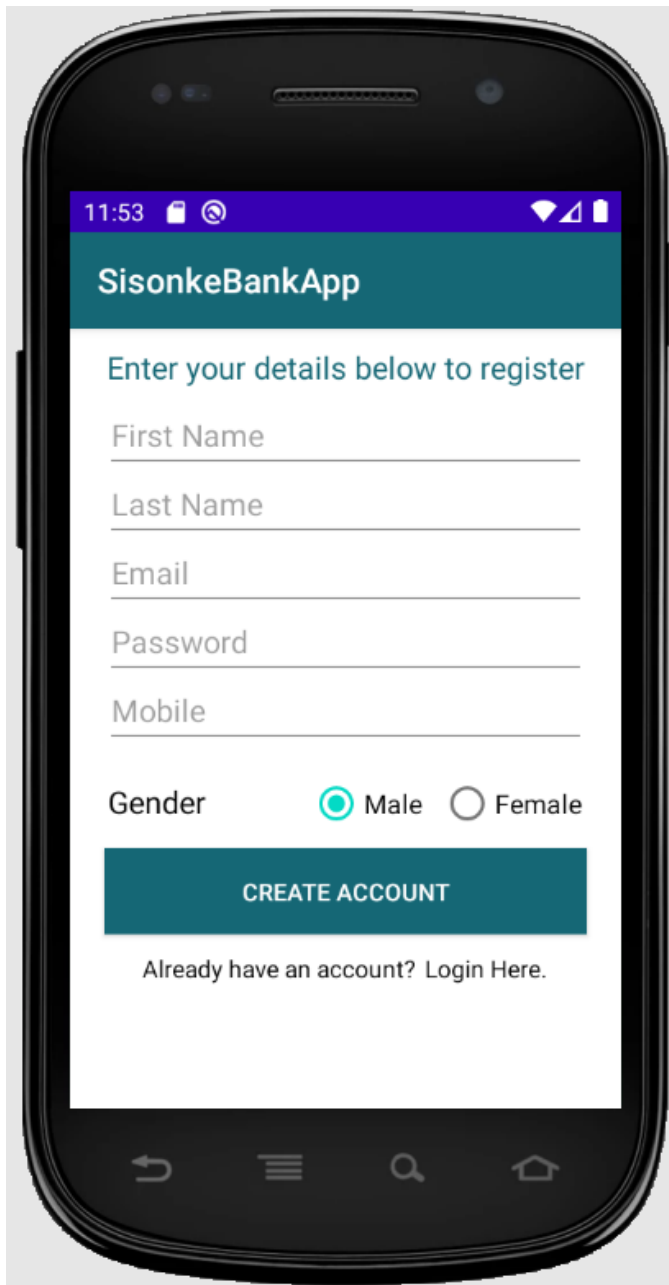
    <!--'Login Here' section-->
    <TextView
        android:id="@+id/textView_login_here"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:textSize="14dp"
        android:paddingLeft="5dp"
        android:text="Login Here."/>
    </LinearLayout>

</LinearLayout>

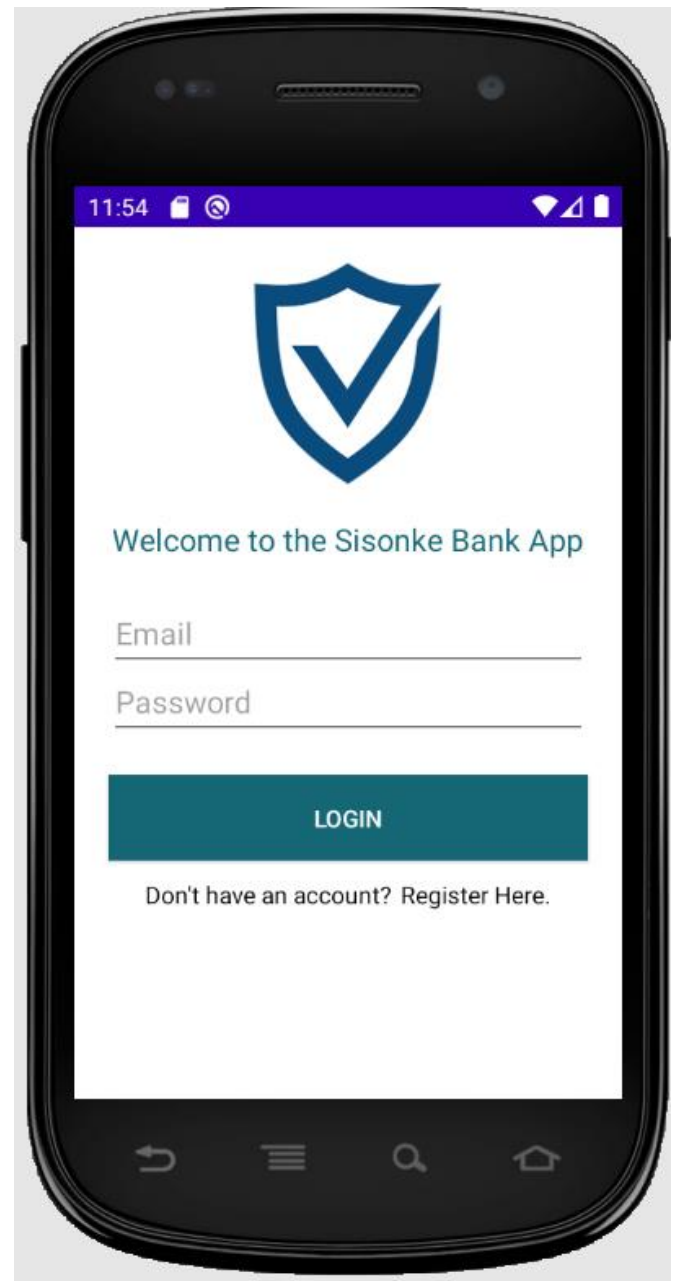
```

## Registration Activity User Interface (Screenshots)

Below: The user interface of the Registration Activity. When a user clicks on 'Login Here', he/she will be redirected to the login screen. This is done using intents.



Below: The login screen.





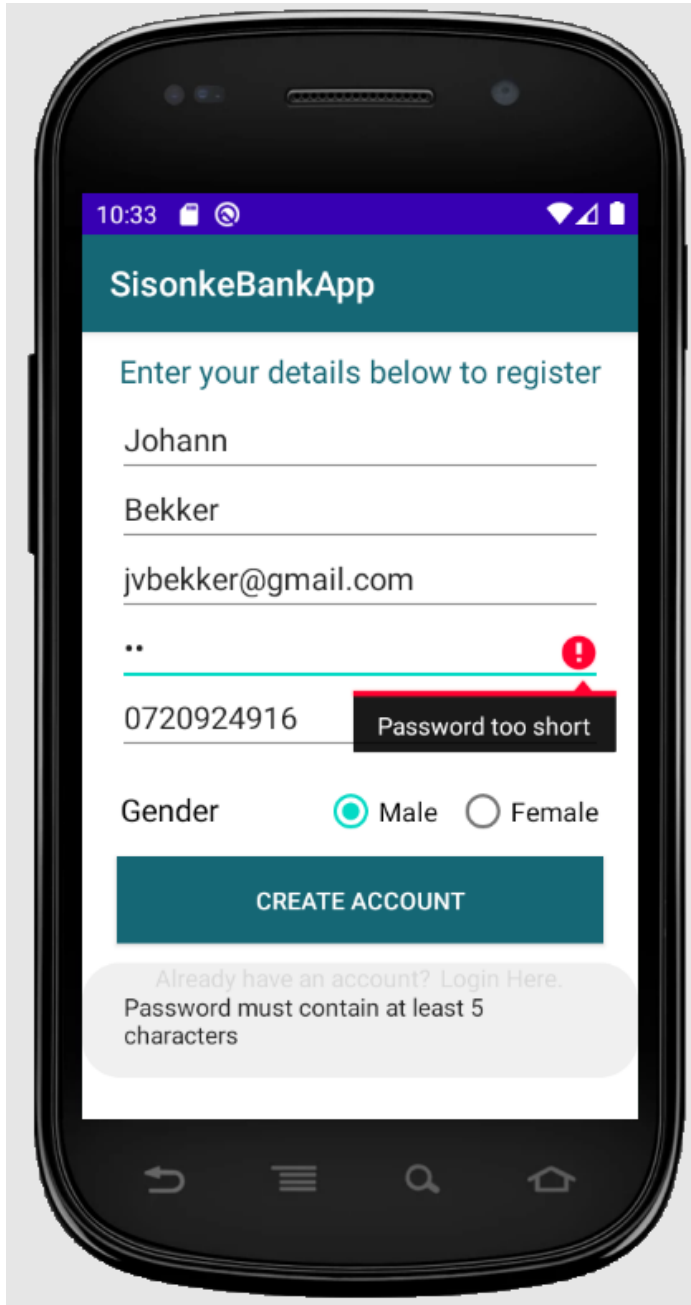
Below: The registration activity has validation which ensures that all required details are entered before an account is created. If a user tries to register without entering the details, a toast message will appear. This toast message will inform the user to enter the details before an account can be created.

The screenshot shows the registration screen of the SisonkeBankApp. The title bar is teal with the text "SisonkeBankApp". Below the title bar, the text "Enter your details below to register" is displayed. There are five input fields: "First Name", "Last Name", "Email", "Password", and "Mobile". Each field has a red exclamation mark icon to its right, indicating a validation error. Below these fields are two radio buttons for "Gender", with "Male" selected. At the bottom, there is a teal button labeled "CREATE ACCOUNT". Below the button, the text "Already have an account? Login Here." is displayed. At the very bottom, there is a light gray button labeled "Enter the required details".

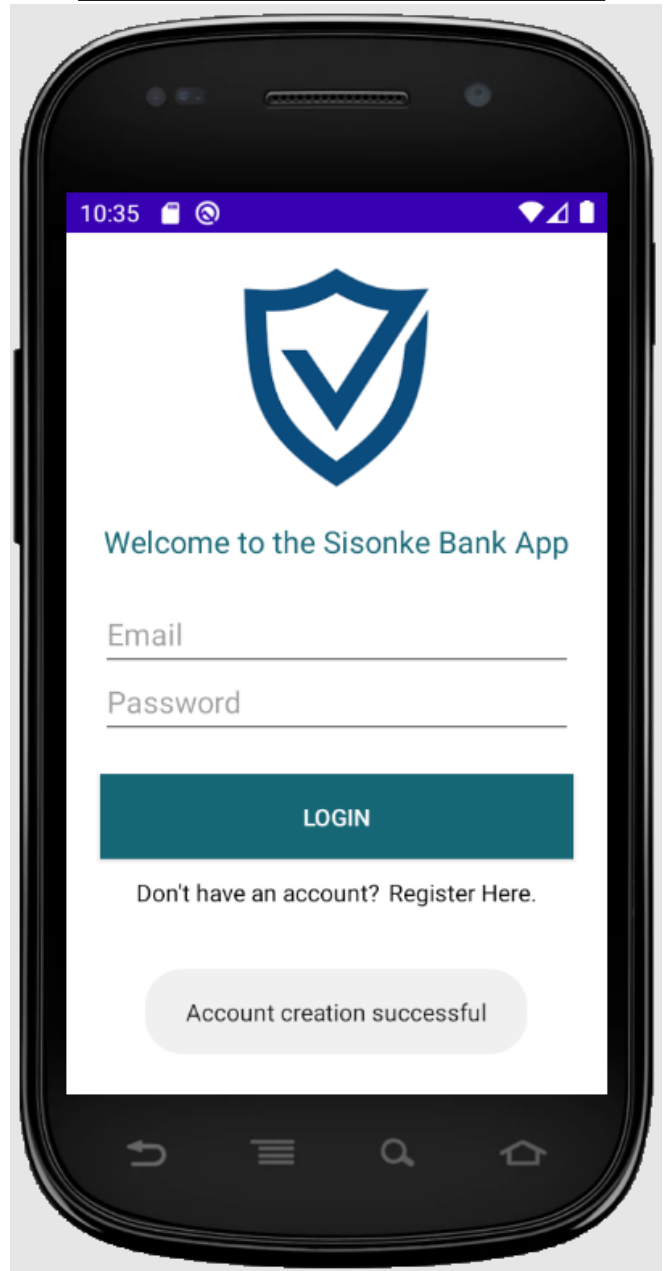
Below: Validation is used to ensure that the entered email has the correct format. If the entered email does not have the correct format, a toast message appears. This toast message informs the user that the email should have the correct format. Also, the mobile number field only accepts numbers.

The screenshot shows the registration screen of the SisonkeBankApp. The title bar is teal with the text "SisonkeBankApp". Below the title bar, the text "Enter your details below to register" is displayed. There are five input fields: "First Name" (containing "Johann"), "Last Name" (containing "Bekker"), "Email" (containing "jybekker@gmail.com"), "Password" (containing "....."), and "Mobile" (containing "0720924916"). The "Email" field has a red exclamation mark icon to its right, indicating a validation error. Below these fields are two radio buttons for "Gender", with "Male" selected. At the bottom, there is a teal button labeled "CREATE ACCOUNT". Below the button, the text "Already have an account? Login Here." is displayed. At the very bottom, there is a light gray button labeled "Enter a valid email address".

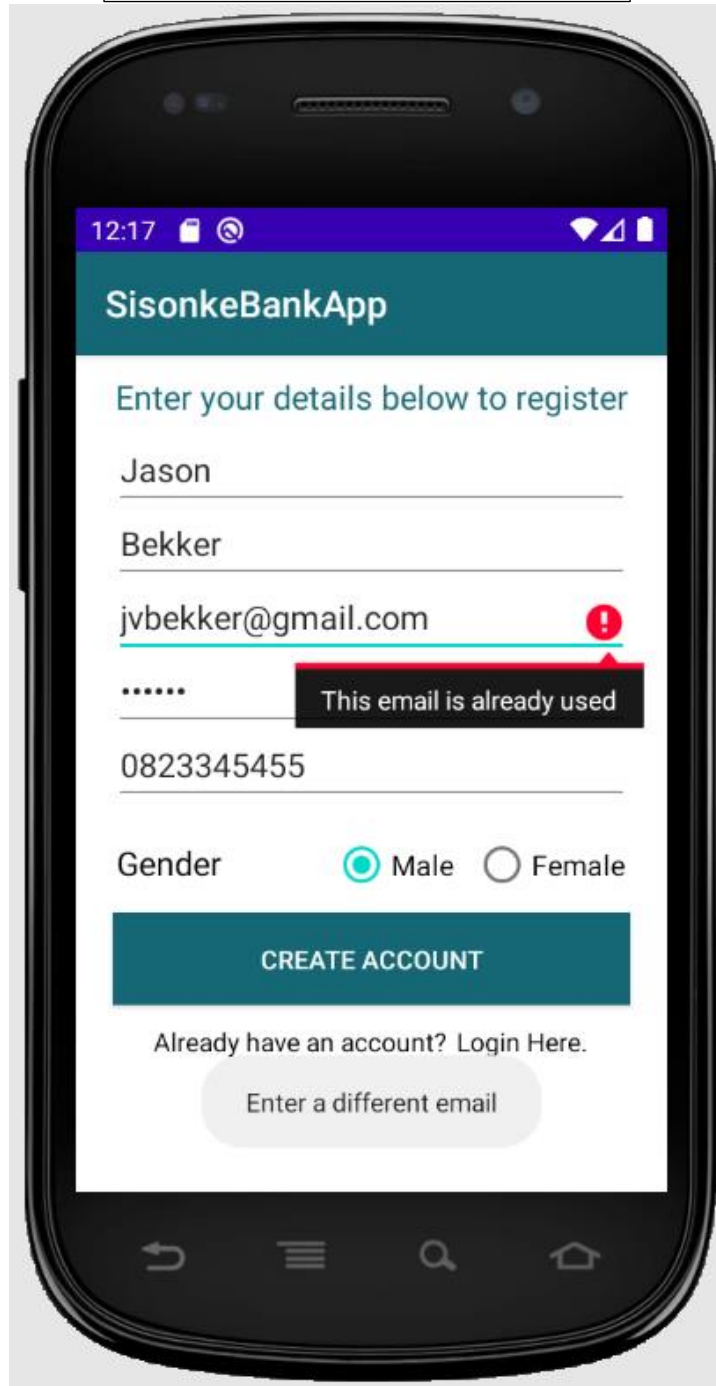
Below: Validation is used to ensure that the entered password is at least 5 characters long. If the password is not at least 5 characters long, a toast message appears to inform the user. Also, the password is hidden while it is being entered.



Below: Once all details are correctly entered, the new account is created. The user is redirected to the login activity and a toast message is displayed. This toast message informs the user that the account has been created.



Below: The registration activity also checks to see if the entered email already exists in the database. If it is already used/in the database, a toast message appears. This toast message asks the user to use a different email address.



## SQLite Database

### SQLiteOpenHelper Class

```
package com.example.bankingapplication;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;

import androidx.annotation.Nullable;

// The SQLite helper class that is used to perform CRU (Create, Read & Update) operations
public class DatabaseOpenHelper extends android.database.sqlite.SQLiteOpenHelper {
    public static final String DATABASE_NAME = "Banking.db";
    public static final String TABLE_NAME = "users";

    public static final String COLUMN_1 = "userID";
    public static final String COLUMN_2 = "userFirstName";
    public static final String COLUMN_3 = "userLastName";
    public static final String COLUMN_4 = "userEmail";
    public static final String COLUMN_5 = "UserPassword";
    public static final String COLUMN_6 = "userMobile";
    public static final String COLUMN_7 = "userGender";
    public static final String COLUMN_8 = "userCurrentAccountBalance";
    public static final String COLUMN_9 = "userSavingsAccountBalance";

    // A database named 'Banking.db' is created
    public DatabaseOpenHelper(@Nullable Context context) {
        super(context, DATABASE_NAME, null, 1);
    }

    // Creates a table named 'users'
    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        sqLiteDatabase.execSQL("create table "+ TABLE_NAME +" (userID INTEGER PRIMARY KEY AUTOINCREMENT," +
            "userFirstName TEXT, userLastName TEXT, userEmail TEXT, userPassword TEXT, userMobile TEXT," +
            "userGender TEXT, userCurrentAccountBalance TEXT, userSavingsAccountBalance TEXT)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);
        onCreate(sqLiteDatabase);
    }

    // The addUser() method is used to add user details on registration screen to the database
```

```

public boolean addUser(BankUser u){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues CV = new ContentValues();

    // The getter methods in the BankUser class is used to retrieve the user's details
    CV.put(COLUMN_2, u.getName());
    CV.put(COLUMN_3, u.getSurname());
    CV.put(COLUMN_4, u.getEmail());
    CV.put(COLUMN_5, u.getPassword());
    CV.put(COLUMN_6, u.getMobile());
    CV.put(COLUMN_7, u.getGender());
    CV.put(COLUMN_8, u.getCurrentAccountBalance());
    CV.put(COLUMN_9, u.getSavingsAccountBalance());

    // Gets the result of the data insertion
    long result = db.insert(TABLE_NAME,null, CV);
    if(result == -1)
        return false;
    else
        return true;
}

/*
The authenticateUser() method authenticates the user who tries to login from the login screen
*/
public String authenticateUser(String email, String password){

    String returnValue = "";
    String[] columns ={COLUMN_1};
    SQLiteDatabase db = getReadableDatabase();
    String selection = COLUMN_4 + "=? " + " and " + COLUMN_5 + "=?";
    String[] selectionArguements = {email, password};

    // Gets user details from the database using cursor
    Cursor cursor = db.query(TABLE_NAME, columns,selection,selectionArguements,
        null,null,null);

    int count = cursor.getCount();

    // If a record exists
    if(count>0){
        returnValue = "UserFound";
        cursor.close();
        db.close();
    }

    // If the username and password is not found in the database
    if(count==0){
        // Determine if the email entered exists
        Boolean result = checkIfEmailExists(email);

        // If the email exists
        if(result == true){
            // The password entered is incorrect
            returnValue = "PasswordIncorrect";
        }
    }
}

```

```

    }else{
        returnValue = "AccountNotExists";
    }
    // Close cursor and database
    cursor.close();
    db.close();
}
return returnValue;
}

```

/\*

The CheckIfEmailExists() method checks to see if the email entered on the registration activity already exists in the database. If the email does exist, a 'true' is returned.

\*/

```

public boolean checkIfEmailExists(String emailEntered) {
    String[] columns = {COLUMN_1};
    SQLiteDatabase db = getReadableDatabase();
    String selection = COLUMN_4 + "=?";
    String[] selectionArguments = {emailEntered};

    // Gets user details from the database using cursor
    Cursor cursor = db.query(TABLE_NAME, columns, selection, selectionArguments,
        null, null, null);

    int count = cursor.getCount();
    cursor.close();
    db.close();

    if(count>0){
        return true;
    }else{
        return false;
    }
}

```

/\*

The getUserDetails() method gets all user details from the database

\*/

```

public Cursor getUserDetails(String email)throws SQLException {

    SQLiteDatabase db = getReadableDatabase();

    String[] selectionArguments = {email};

    // Get user details from the database using cursor
    Cursor result = db.rawQuery("select * from " + TABLE_NAME +
        " where " + COLUMN_4 + "=?", selectionArguments);

    // Return details
    return result;
}

```

```

/*
The updateBalance() method is used to update the user's account balances in the database
*/
public boolean updateBalance(String user, double newCurrentBal, double newSavingsBal){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues CV = new ContentValues();

    // Add new balances to ContentValues instance
    CV.put(COLUMN_8, newCurrentBal);
    CV.put(COLUMN_9, newSavingsBal);

    db.insert(TABLE_NAME,null, CV);
    db.update(TABLE_NAME, CV,"userEmail = ?", new String[] {user});

    return true;
}
}

```

## Bank User Class

```
package com.example.bankingapplication;
```

```
// This BankUser class is used to handle data transfers to and from the SQLite database.
```

```
public class BankUser {

    String name;
    String surname;
    String email;
    String password;
    String mobile;
    String gender;
    double currentAccountBalance;
    double savingsAccountBalance;

    // Constructor
    public BankUser(String name, String surname, String email, String password, String mobile,
                    String gender, double currentAccountBalance, double savingsAccountBalance) {

        this.name = name;
        this.surname = surname;
        this.email = email;
        this.password = password;
        this.mobile = mobile;
        this.gender = gender;
        this.currentAccountBalance = currentAccountBalance;
        this.savingsAccountBalance = savingsAccountBalance;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public String getName()
    {
        return this.name;
    }

    public void setSurname(String surname)
    {
        this.surname = surname;
    }

    public String getSurname()
    {
        return this.surname;
    }

    public void setEmail(String email)
    {
        this.email = email;
    }
}
```



```

public String getEmail()
{
    return this.email;
}

public void setPassword(String password)
{
    this.password = password;
}

public String getPassword()
{
    return this.password;
}

public void setMobile(String mobile)
{
    this.mobile = mobile;
}

public String getMobile()
{
    return this.mobile;
}

public void setGender(String gender)
{
    this.gender = gender;
}

public String getGender()
{
    return this.gender;
}

public void setCurrentAccountBalance(double currentAccountBalance)
{
    this.currentAccountBalance = currentAccountBalance;
}

public double getCurrentAccountBalance()
{
    return this.currentAccountBalance;
}

public void setSavingsAccountBalance(double savingsAccountBalance)
{
    this.savingsAccountBalance = savingsAccountBalance;
}

public double getSavingsAccountBalance()
{
    return this.savingsAccountBalance;
}
}

```

## Main Page Activity

```
/*
 * This is the main page activity. It is the first screen the user
 * sees when he/she logs in.
 */

package com.example.bankingapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainPageActivity extends AppCompatActivity{

    // Local variables
    DatabaseOpenHelper db;
    Button viewBalancesButton, transferBetweenAccountsButton, logoutButton;
    TextView userEmail;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_page);

        // Sets the title for the actionbar
        getSupportActionBar().setTitle("SisonkeBankApp");

        // Creates a new instance of the SQLiteOpenHelper class
        db = new DatabaseOpenHelper(this);

        // TextView used to greet user
        userEmail = (TextView) findViewById(R.id.welcomeVIEW);

        // Buttons on the main page
        viewBalancesButton = (Button) findViewById(R.id.viewAccountBalanceBTN);
        transferBetweenAccountsButton = (Button) findViewById(R.id.transferBetweenAccountsBTN);
        logoutButton = (Button) findViewById(R.id.logoutMainBTN);

        // The greetUser() method gets the user's first name from the database and displays a welcome
        message
        greetUser();

    }

    /*
    An action listener is placed on the 'VIEW BALANCES' button. Once clicked, it directs the
    user to the view balances activity using an intent. The logged in user's email is passed to
    the view balances activity through the intent.
    */
}
```

```

*/
viewBalancesButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent balancesScreen = new Intent(MainPageActivity.this,
            ViewAccountBalanceActivity.class);
        Intent intent = getIntent();
        String user = intent.getStringExtra("loggedInUserEmail");
        balancesScreen.putExtra("loggedInUserEmail", user);
        startActivity(balancesScreen);
    }
});

/*
An action listener is placed on the 'TRANSFER BETWEEN ACCOUNTS' button. Once clicked,
it directs the user to the transfer between accounts activity using an intent. The
logged in user's email is passed to the transfer between accounts activity through the intent.
*/
transferBetweenAccountsButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {
        Intent transferScreen = new Intent(MainPageActivity.this,
            TransferBetweenAccountsActivity.class);
        Intent intent = getIntent();
        String user = intent.getStringExtra("loggedInUserEmail");
        transferScreen.putExtra("loggedInUserEmail", user);
        startActivity(transferScreen);
    }
});

/*
An action listener is placed on the 'LOGOUT' button. Once clicked,
it directs the user to the login activity using an intent. finish() is used to ensure that
the user cannot log back in by pressing the back button.
*/
logoutButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        finish();
        Intent loginScreen = new Intent(MainPageActivity.this,
            LoginActivity.class);
        startActivity(loginScreen);

        // Toast message informs user that he/she is logged out
        Toast.makeText(MainPageActivity.this,
            "Logged Out", Toast.LENGTH_SHORT).show();
    }
});
}

/*
The greetUser() method gets the user's first name from the database and displays
a welcome message. The user's details is retrieved from the database using the getUserDetails()
method in the DatabaseOpenHelper class. The logged in user's email is obtained from an intent.
*/

```

```

private void greetUser() {

    // Obtains the logged in user's email is obtained from an intent.
    Intent intent = getIntent();
    String user = intent.getStringExtra("loggedInUserEmail");

    // Get user details from database using the getUerDetails() method and user email
    Cursor cursor = db.getUserDetails(user);

    // If no user details are found
    if(cursor.getCount() == 0){
        return;
    }

    // If the cursor is not null and move to first
    if( cursor != null && cursor.moveToFirst() ){

        // Get the user's name in the 'userFirstName' column
        String current = cursor.getString(cursor.getColumnIndex("userFirstName"));

        // Set greeting message using the retrieved first name
        userEmail.setText("Welcome " + current);

        // Close cursor
        cursor.close();
    }
}
}

```

## activity\_main\_page.xml file

```
<?xml version="1.0" encoding="utf-8"?>

<!--This file is used to set the design and layout of the main page-->

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp"
        android:orientation="horizontal">
        <TextView
            android:id="@+id/welcomeVIEW"
            android:layout_width="wrap_content"
            android:textSize="20dp"
            android:layout_height="wrap_content"
            android:textColor="#156775"
            android:textAlignment="center"
            android:gravity="center_horizontal" />
    </LinearLayout>

    <!--View balances button-->
    <Button
        android:id="@+id/viewAccountBalanceBTN"
        android:layout_width="280dp"
        android:layout_height="50dp"
        android:background="#156775"
        android:textColor="#ffffff"
        android:layout_marginTop="20dp"
        android:text="@string/viewAccountBalanceBTN" />

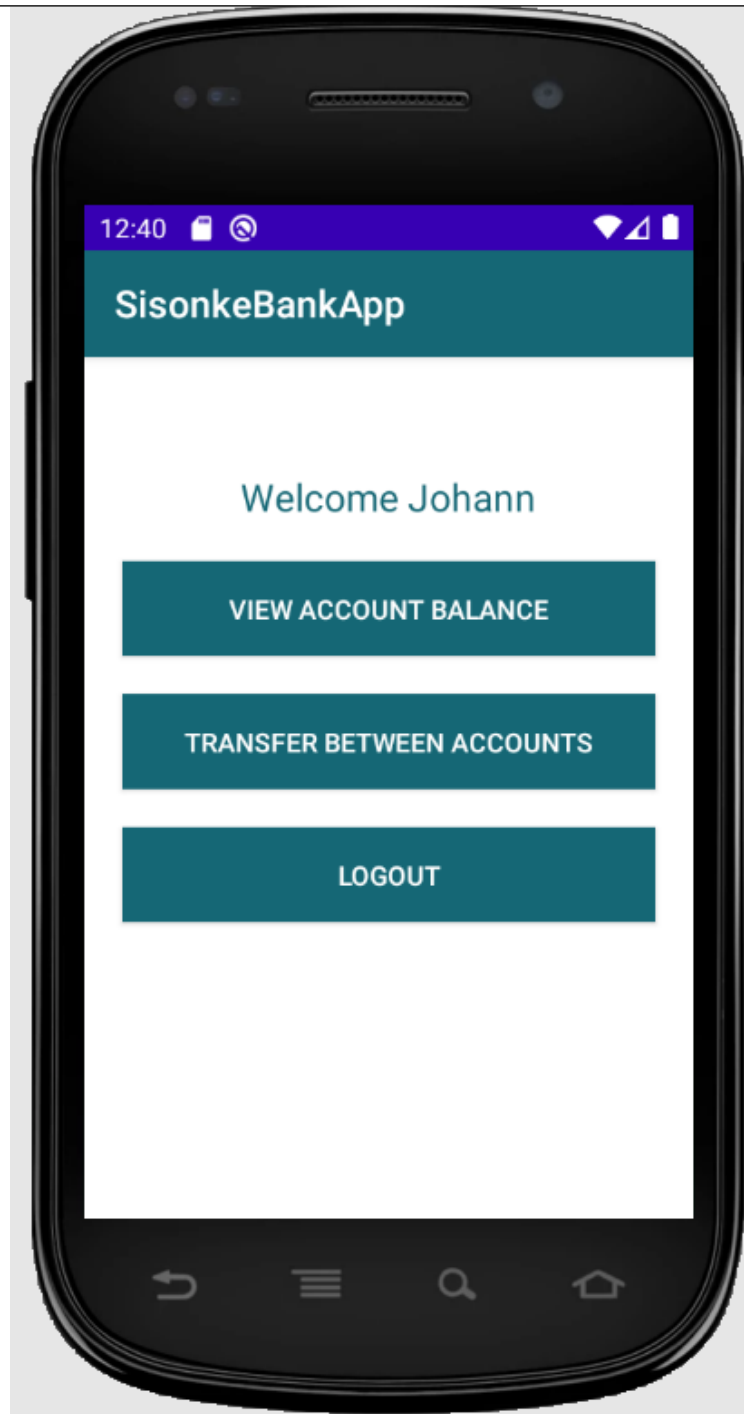
    <!--Transfer between accounts button-->
    <Button
        android:id="@+id/transferBetweenAccountsBTN"
        android:layout_width="280dp"
        android:layout_height="50dp"
        android:background="#156775"
        android:textColor="#ffffff"
        android:layout_marginTop="20dp"
        android:text="@string/transferBetweenAccountsBTN" />

    <!--Logout button-->
    <Button
        android:id="@+id/logoutMainBTN"
```

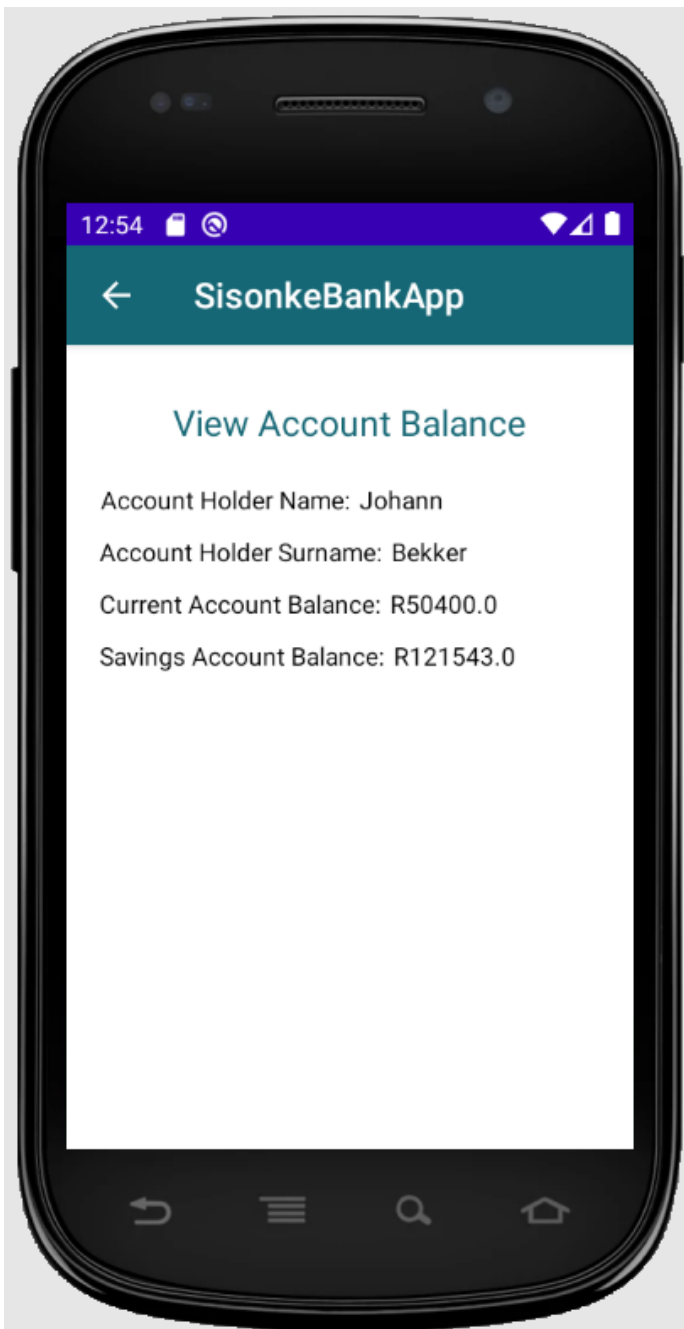
```
        android:layout_width="280dp"
        android:layout_height="50dp"
        android:background="#156775"
        android:textColor="#ffffff"
        android:layout_marginTop="20dp"
        android:text="@string/logoutMainBTN" />
</LinearLayout>
```

## Main Page Activity User Interface (Screenshots)

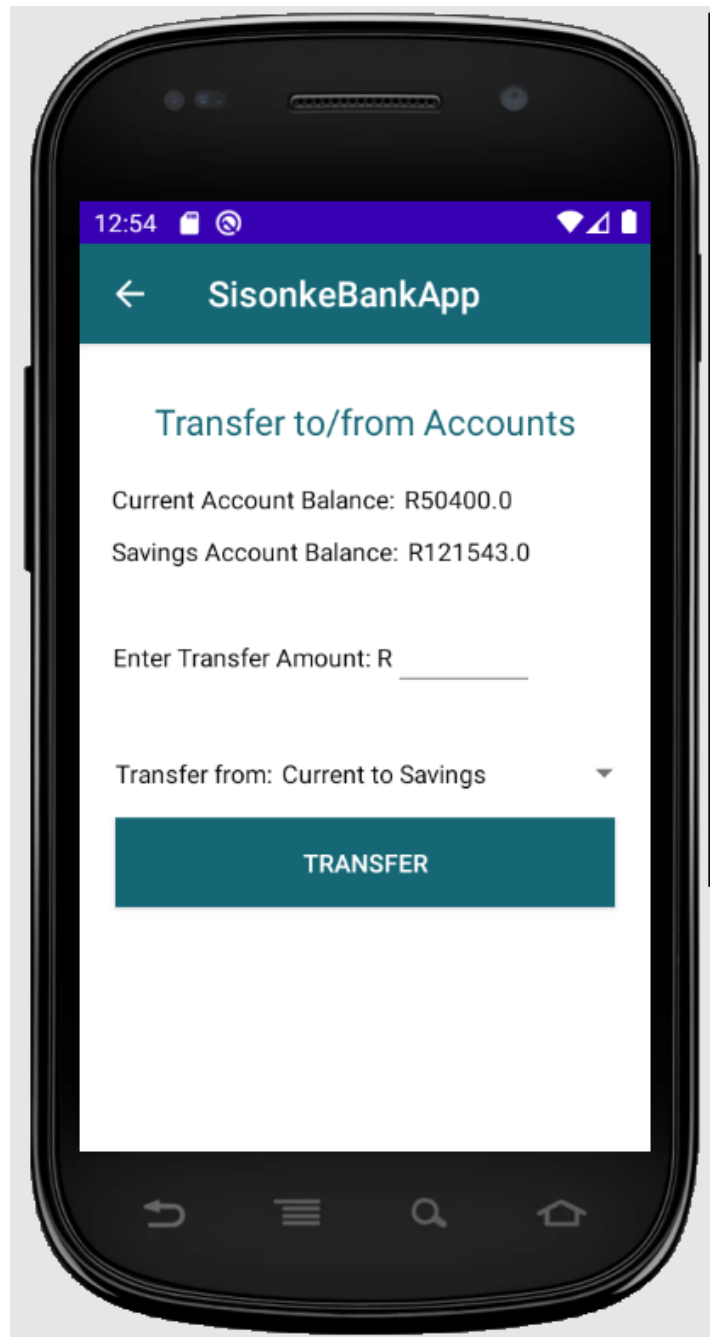
Below: The main page activity user interface. The logged in user's name is retrieved from the database and displayed in a TextView widget. The 'VIEW ACCOUNT BALANCE' button directs the user to the view account balance screen (please refer to Figure A). The 'TRANSFER BETWEEN ACCOUNTS' button directs the user to the transfer between accounts screen (please refer to Figure B). Intents are used to manage the movement from one activity to another. The 'LOGOUT' button logs out the user. Once logged out, a toast message appears. This toast message informs the user that he/she has been logged out (please refer to Figure C). The user cannot log back in by simply pressing the back button.



Below: Figure A



Below: Figure B





Below: Figure C



## View Account Balance Activity

```
/*
 * This is the view account balance activity. It allows the user to see the balances of his/her
 * accounts on the SQLite database.
 */

package com.example.bankingapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

public class ViewAccountBalanceActivity extends AppCompatActivity {

    // Local variables
    TextView firstNameTextView, lastNameTextView, currentBalanceTextView,
    savingsBalanceTextView;
    DatabaseOpenHelper db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_account_balance);

        // Sets the title for the actionbar
        getSupportActionBar().setTitle("SisonkeBankApp");

        // Shows back arrow in action bar
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        // TextViews used to display the user details
        firstNameTextView = (TextView) findViewById(R.id.textView_account_holder_name);
        lastNameTextView = (TextView) findViewById(R.id.textView_account_holder_surname);
        currentBalanceTextView = (TextView) findViewById(R.id.textView_account_holder_current);
        savingsBalanceTextView = (TextView) findViewById(R.id.textView_account_holder_savings);

        // Shows user details in TextViews
        showUserDetails();
    }

    /*
    The showUserDetails() method shows the user's first name, last name and account balances
    in TextViews. It uses the getUserDetails() method in the SQLiteOpenHelper class to obtain these
    details. The logged in user's email is obtained from an intent and passed as a parameter to this
    method.
    */
    private void showUserDetails() {
```

```

// Creates a new instance of the SQLiteOpenHelper class
db = new DatabaseOpenHelper(this);

// Obtains the logged in user's email is obtained from an intent.
Intent intent = getIntent();
String user = intent.getStringExtra("loggedInUserEmail");

// Get user details from database using the getUerDetails() method and user email
Cursor cursor = db.getUserDetails(user);

// If no user details are found
if(cursor.getCount() == 0){
    return;
}

// If the cursor is not null and move to first
if( cursor != null && cursor.moveToFirst() ){

    // Get the user's first name from the 'userFirstName' column
    String firstName = cursor.getString(cursor.getColumnIndex("userFirstName"));
    // Get the user's last name from the 'userLastName' column
    String lastName = cursor.getString(cursor.getColumnIndex("userLastName"));
    // Get the user's current account balance from the 'userCurrentAccountBalance' column
    String currentAccBalance =
cursor.getString(cursor.getColumnIndex("userCurrentAccountBalance"));
    // Get the user's savings account balance from the 'userSavingsAccountBalance' column
    String savingsAccBalance =
cursor.getString(cursor.getColumnIndex("userSavingsAccountBalance"));

    // Sets TextViews to display user details
    firstNameTextView.setText(firstName );
    lastNameTextView.setText(lastName);
    currentBalanceTextView.setText("R" + currentAccBalance);
    savingsBalanceTextView.setText("R" + savingsAccBalance);

    // Close cursor
    cursor.close();
}
}

/*
This section gives the back arrow the ability to go to the main page activity when clicked.
*/
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            finish();
            return true;
    }
    return super.onOptionsItemSelected(item);
}

public boolean onCreateOptionsMenu(Menu menu) {

```

```
    return true;  
  }  
}
```

## activity\_view\_account\_balance.xml file

```
<?xml version="1.0" encoding="utf-8"?>

<!--This file is used to set the design and layout of the view account balance screen activity-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    tools:context=".ViewAccountBalanceActivity">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:textSize="20dp"
            android:layout_height="wrap_content"
            android:textColor="#156775"
            android:text="View Account Balance"/>
    </LinearLayout>

    <!--User's first name-->
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="14dp"
            android:text="Account Holder Name:"
            android:textColor="#000000" />
        <TextView
            android:id="@+id/textView_account_holder_name"
            android:layout_width="135dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5dp"
            android:textSize="14dp"
            android:textColor="#000000" />
    </LinearLayout>

    <!--User's last name-->
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:orientation="horizontal">
        <TextView
```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14dp"
        android:text="Account Holder Surname:"
        android:textColor="#000000" />
<TextView
    android:id="@+id/textView_account_holder_surname"
    android:layout_width="118dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="5dp"
    android:textSize="14dp"
    android:textColor="#000000" />
</LinearLayout>

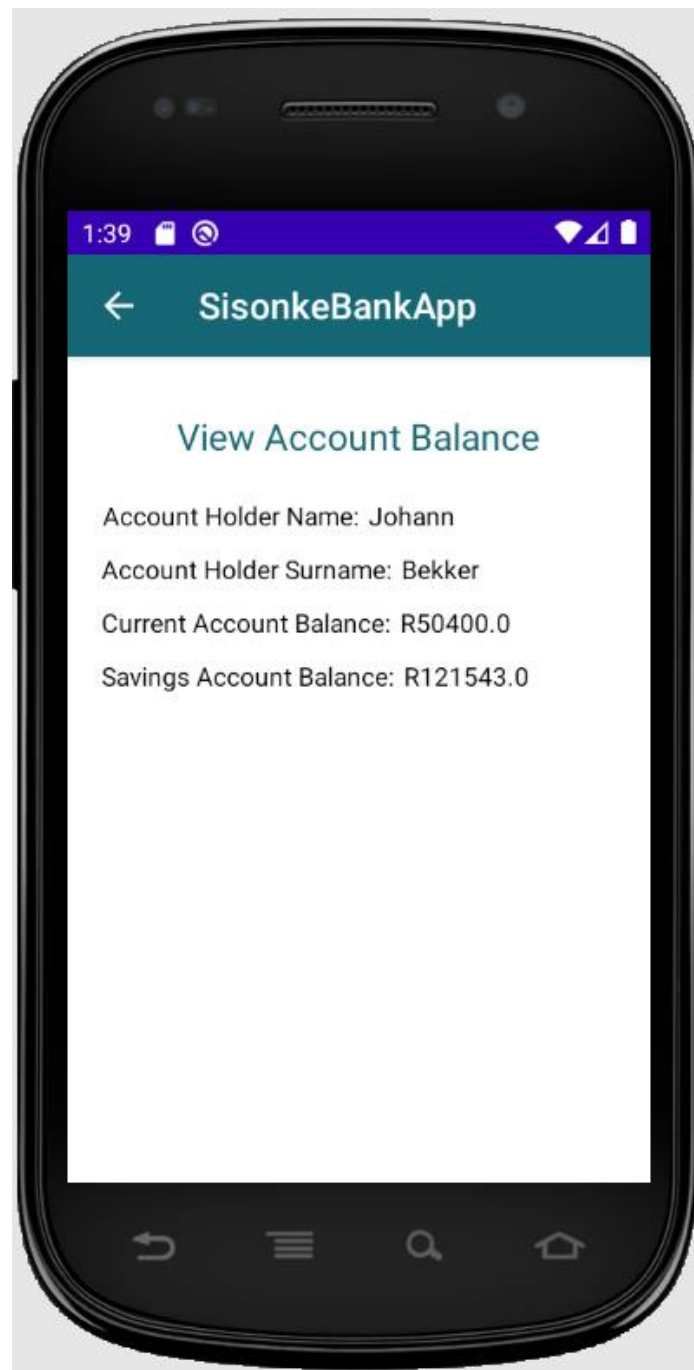
<!--User's current account balance-->
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:orientation="horizontal">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14dp"
        android:text="Current Account Balance:"
        android:textColor="#000000" />
    <TextView
        android:id="@+id/textView_account_holder_current"
        android:layout_width="118dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dp"
        android:textSize="14dp"
        android:textColor="#000000" />
</LinearLayout>

<!--User's savings account balance-->
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:orientation="horizontal">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14dp"
        android:text="Savings Account Balance:"
        android:textColor="#000000" />
    <TextView
        android:id="@+id/textView_account_holder_savings"
        android:layout_width="116dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dp"
        android:textSize="14dp"
        android:textColor="#000000" />
</LinearLayout>
</LinearLayout>

```

## View Account Balance Activity User Interface (Screenshots)

Below: The logged in user's details are displayed on the view account balance screen. These details are retrieved from the database. This activity also has a back arrow which allows the user to go back to the Main Page activity.



## Transfer Between Accounts Activity

```
/*
 * This is the transfer between accounts activity. It allows the user to
 * transfer funds from one account to another.
 */

package com.example.bankingapplication;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.List;

public class TransferBetweenAccountsActivity extends AppCompatActivity implements
    AdapterView.OnItemClickListener {

    // Local variables
    TextView currentBalance, savingsBalance;
    Button transferFundsBTN;
    EditText amountToTransferEdit;
    DatabaseOpenHelper db;
    String transferOptionSelected = "", amountEnteredString = "";
    double newCurrentBal = 0;
    double newSavingsBal = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_transfer_between_accounts);

        // Sets the title for the actionbar
        getSupportActionBar().setTitle("SisonkeBankApp");

        // Shows back arrow in action bar
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        // Transfer funds button
        transferFundsBTN = (Button)findViewById(R.id.transferFundsBTN);

        // Amount to transfer text field
```



```

amountToTransferEdit = (EditText) findViewById(R.id.editText_amountToTransfer);

// Balance TextViews
currentBalance = (TextView) findViewById(R.id.textView_account_current_transfer);
savingsBalance = (TextView) findViewById(R.id.textView_account_savings_transfer);

// Spinner widget
Spinner spinner = (Spinner) findViewById(R.id.spinner);
spinner.setOnItemSelectedListener(this);

// Creates ArrayList and adds transfer options
List<String> transferOption = new ArrayList<String>();
transferOption.add("Current to Savings");
transferOption.add("Savings to Current");

// Creates an adapter for the spinner widget
ArrayAdapter<String> dataAdapter =
    new ArrayAdapter<String>(this,
        R.layout.custom_spinner_transfer_options, transferOption);

// Sets the style of the spinner widget's layout
dataAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

// Adds the adapter to the spinner widget
spinner.setAdapter(dataAdapter);

// Transfers funds from one account to another
transferFunds();

// Shows user's balances in TextViews
showBalances();
}

// Spinner methods
@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
    // Get selected transfer option
    transferOptionSelected = parent.getItemAtPosition(position).toString();
}

public void onNothingSelected(AdapterView<?> arg0) {
    // Nothing
}

/*
This transferFunds() method is used to transfer funds between the user's accounts
*/
private void transferFunds() {

    // New instance of the DatabaseOpenHelper class
    db = new DatabaseOpenHelper(this);

```

```

// Adds an action listener to the 'TRANSFER' button
transferFundsBTN.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        // If the user entered an amount to transfer
        if (amountToTransferEdit.getText().toString().trim().length() != 0) {

            // Get logged in user's email from intent
            Intent intent = getIntent();
            String user = intent.getStringExtra("loggedInUserEmail");

            // Get user details from database using the getUserDetails() method in the
            DatabaseOpenHelper class
            Cursor cursor = db.getUserDetails(user);

            // If no details are found
            if (cursor.getCount() == 0) {
                return;
            }

            // If cursor is not null nad move to first
            if (cursor != null && cursor.moveToFirst()) {

                // Get the user's current account balance from the 'userCurrentAccountBalance'
                column
                double currentAccountBal =
                cursor.getDouble(cursor.getColumnIndex("userCurrentAccountBalance"));
                // Get the user's savings account balance from the 'userSavingsAccountBalance'
                column
                double savingsAccountBal =
                cursor.getDouble(cursor.getColumnIndex("userSavingsAccountBalance"));

                // Get amount entered in text field
                amountEnteredString = amountToTransferEdit.getText().toString().trim();

                // Convert entered amount to double
                double amountEntered = Double.parseDouble(amountEnteredString);

                // If the current to savings option is selected
                if (transferOptionSelected == "Current to Savings") {

                    // If the amount entered is more than what is available in the current account
                    if (currentAccountBal < amountEntered) {
                        // Show toast message to inform user that the transfer amount cannot be more
                        than what is available
                        Toast.makeText(TransferBetweenAccountsActivity.this,
                            "Insufficient funds in Current Account", Toast.LENGTH_SHORT).show();

                        // If the amount entered is less than what is available in the current account
                    } else {

                        // Calculate the new account balances
                        newCurrentBal = currentAccountBal - amountEntered;
                        newSavingsBal = savingsAccountBal + amountEntered;
                    }
                }
            }
        }
    }
});

```

```

        /*
        Use the updateBalance() method in the DatabaseOpenHelper class to
        update the account balances in the database.
        */
        Boolean result = db.updateBalance(user, newCurrentBal, newSavingsBal);

        // If the transfer was successful
        if (result == true) {
            // Show toast message to inform user that the transfer was successful
            Toast.makeText(TransferBetweenAccountsActivity.this,
                "Transfer completed successfully", Toast.LENGTH_SHORT).show();

            // Show account balances after transfer
            showBalances();
        }
    }

    // If the savings to current option is selected
    if (transferOptionSelected == "Savings to Current") {

        // If the amount entered is more than what is available in the savings account
        if (savingsAccountBal < amountEntered) {
            // Show toast message to inform user that the transfer amount cannot be more
            // than what is available
            Toast.makeText(TransferBetweenAccountsActivity.this,
                "Insufficient funds in Savings Account", Toast.LENGTH_SHORT).show();

            // If the amount entered is less than what is available in the savings account
        } else {

            // Calculate the new account balances
            newSavingsBal = savingsAccountBal - amountEntered;
            newCurrentBal = currentAccountBal + amountEntered;

            /*
            Use the updateBalance() method in the DatabaseOpenHelper class to
            update the account balances in the database.
            */
            Boolean result = db.updateBalance(user, newCurrentBal, newSavingsBal);

            // If the transfer was successful
            if (result == true) {
                // Show toast message to inform user that the transfer was successful
                Toast.makeText(TransferBetweenAccountsActivity.this,
                    "Transfer completed successfully", Toast.LENGTH_SHORT).show();

                // Show account balances after transfer
                showBalances();
            }
        }
    }

    // Close cursor
    cursor.close();
}

```

```

        // If the user did not enter an amount to transfer
    }else{
        Toast.makeText(TransferBetweenAccountsActivity.this,
            "Enter an amount", Toast.LENGTH_SHORT).show();
    }
}
});
}

/*
This showBalances() method is used to display the balances of the user's accounts. It uses the
getUserDetails() method in the DatabaseOpenHelper class to get the user's current account
balances
in the database. The logged in user's email is obtained from an intent and passed as a parameter
to the
getUserDetails() method.
*/
public void showBalances(){

    // Creates an instance of the DatabaseOpenHelper class
    db = new DatabaseOpenHelper(this);

    // Obtains the logged in user's email from the intent
    Intent intent = getIntent();
    String user = intent.getStringExtra("loggedInUserEmail");

    // Gets all of the user's details from the database
    Cursor cursor = db.getUserDetails(user);

    // If no details are found
    if(cursor.getCount() == 0){
        return;
    }

    // If the cursor is not null and move to first
    if( cursor != null && cursor.moveToFirst() ){

        // Get the user's current account balance from the 'userCurrentAccountBalance' column
        String current = cursor.getString(cursor.getColumnIndex("userCurrentAccountBalance"));
        // Get the user's current account balance from the 'userSavingsAccountBalance' column
        String savings = cursor.getString(cursor.getColumnIndex("userSavingsAccountBalance"));

        // Sets TextViews to display the user's account balances
        currentBalance.setText("R"+current);
        savingsBalance.setText("R"+savings);

        // Close the cursor
        cursor.close();
    }
}

/*
This section gives the back arrow the ability to go to the main page activity when clicked.
*/

```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            finish();
            return true;
    }
    return super.onOptionsItemSelected(item);
}
public boolean onCreateOptionsMenu(Menu menu) {
    return true;
}
}
```

## activity\_transfer\_between\_accounts.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<!--This file is used to set the design and layout of the transfer activity screen-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    tools:context=".TransferBetweenAccountsActivity">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:textSize="20dp"
            android:layout_height="wrap_content"
            android:textColor="#156775"
            android:text="Transfer to/from Accounts"/>
    </LinearLayout>

    <!--Current account-->
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="14dp"
            android:text="Current Account Balance:"
            android:textColor="#000000" />
        <TextView
            android:id="@+id/textView_account_current_transfer"
            android:layout_width="120dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5dp"
            android:textSize="14dp"
            android:textColor="#000000" />
    </LinearLayout>

    <!--Savings account-->
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="14dp"
```

```

        android:text="Savings Account Balance:"
        android:textColor="#000000" />
<TextView
    android:id="@+id/textView_account_savings_transfer"
    android:layout_width="118dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="5dp"
    android:textSize="14dp"
    android:textColor="#000000" />
</LinearLayout>

```

```

<!--Transfer amount-->
<!--The transfer amount can only contain numbers-->

```

```

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="30dp"
    android:orientation="horizontal">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14dp"
        android:text="Enter Transfer Amount:"
        android:textColor="#000000" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14dp"
        android:text=" R"
        android:textColor="#000000" />
    <EditText
        android:id="@+id/editText_amountToTransfer"
        android:layout_width="80dp"
        android:layout_height="wrap_content"
        android:digits="0123456789"
        android:textSize="14dp"
        android:textColor="#000000" />
    <TextView
        android:layout_width="45dp"
        android:layout_height="wrap_content"
        android:textSize="14dp"
        android:textColor="#000000" />
</LinearLayout>

```

```

<!--Spinner holding transfer options-->

```

```

<LinearLayout
    android:layout_width="280dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="30dp"
    android:orientation="horizontal">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14dp"
        android:text="Transfer from:"
        android:textColor="#000000" />

```

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="210dp"
    android:layout_height="wrap_content"/>
</LinearLayout>

<!--Transfer button-->
<Button
    android:id="@+id/transferFundsBTN"
    android:layout_width="280dp"
    android:layout_height="50dp"
    android:background="#156775"
    android:textColor="#ffffff"
    android:layout_marginTop="10dp"
    android:text="TRANSFER" />

</LinearLayout>
```

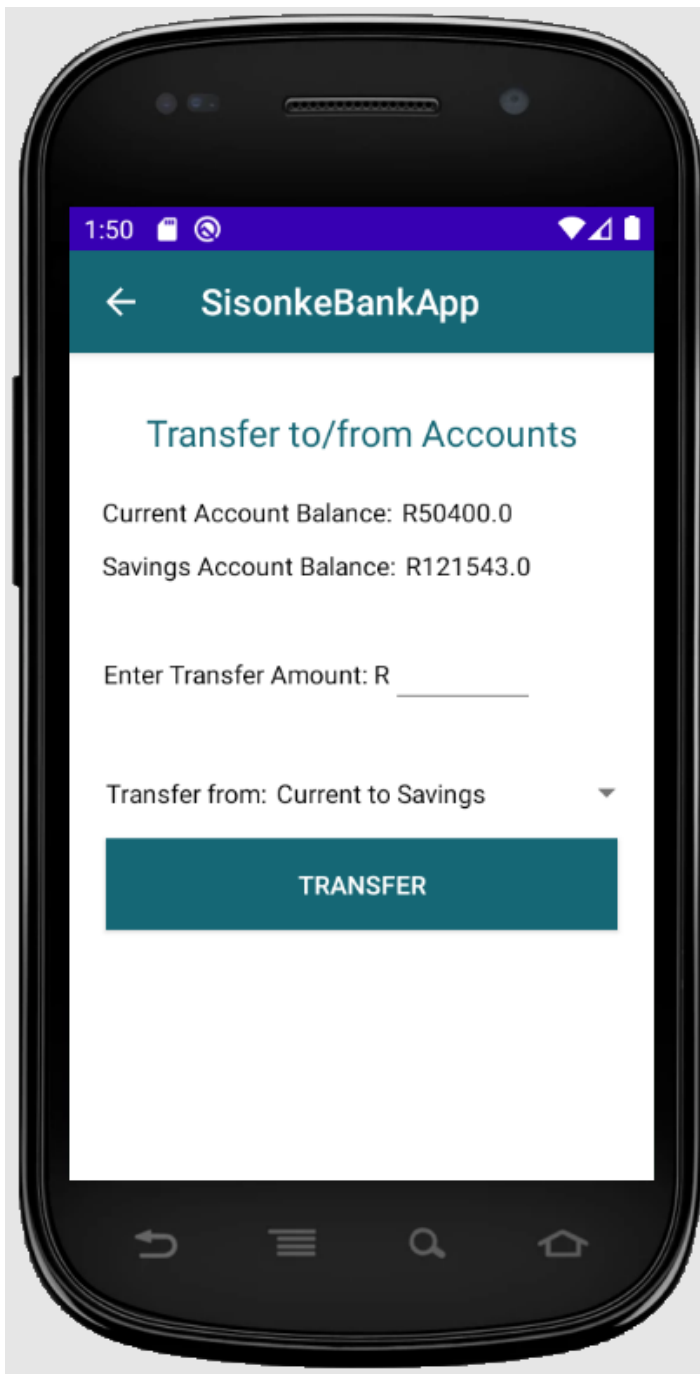


## **custom\_spinner\_transfer\_options.xml file**

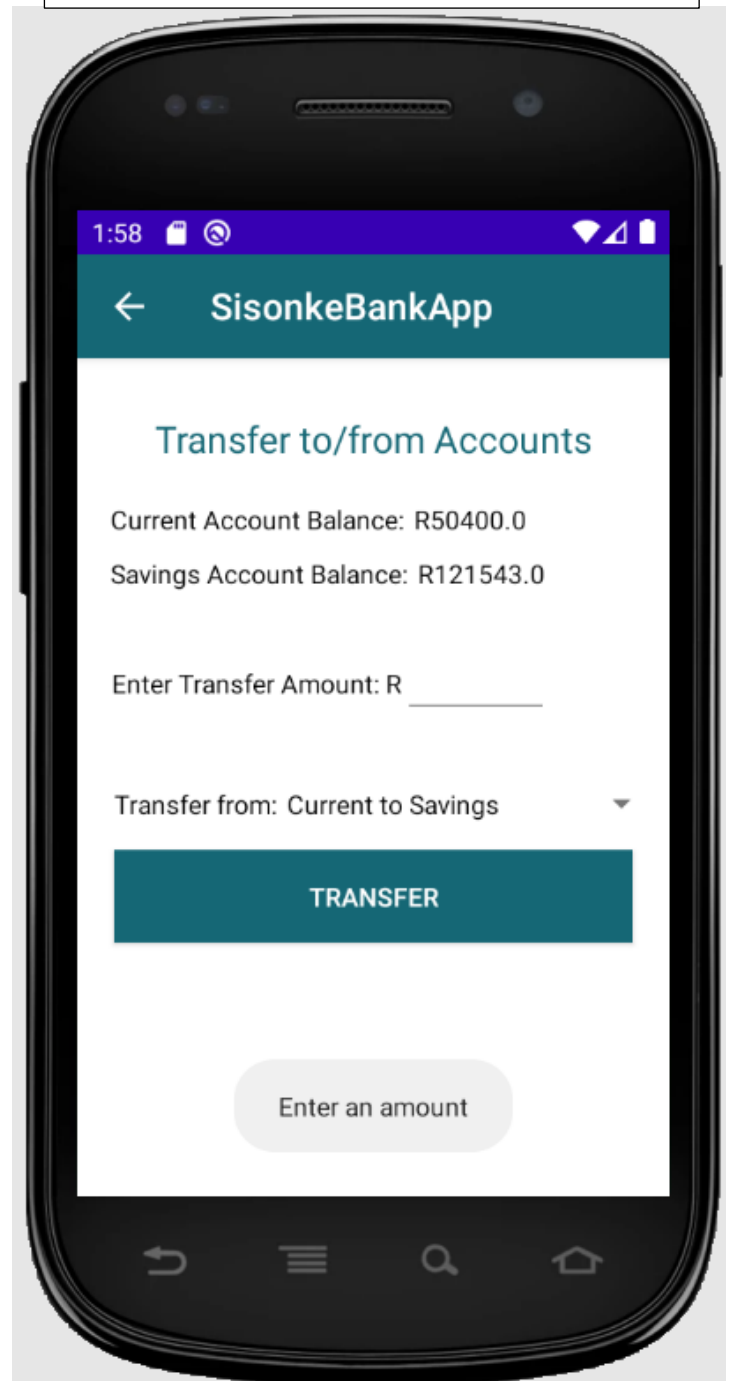
```
<?xml version="1.0" encoding="utf-8"?>
<!--This file is used to set the design and layout of the spinner that is used on the transfer page-->
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="14dp"
    android:gravity="left"
    android:padding="5dip"
    android:textColor="#000000"
/>
```

## Transfer Between Accounts Activity User Interface (Screenshots)

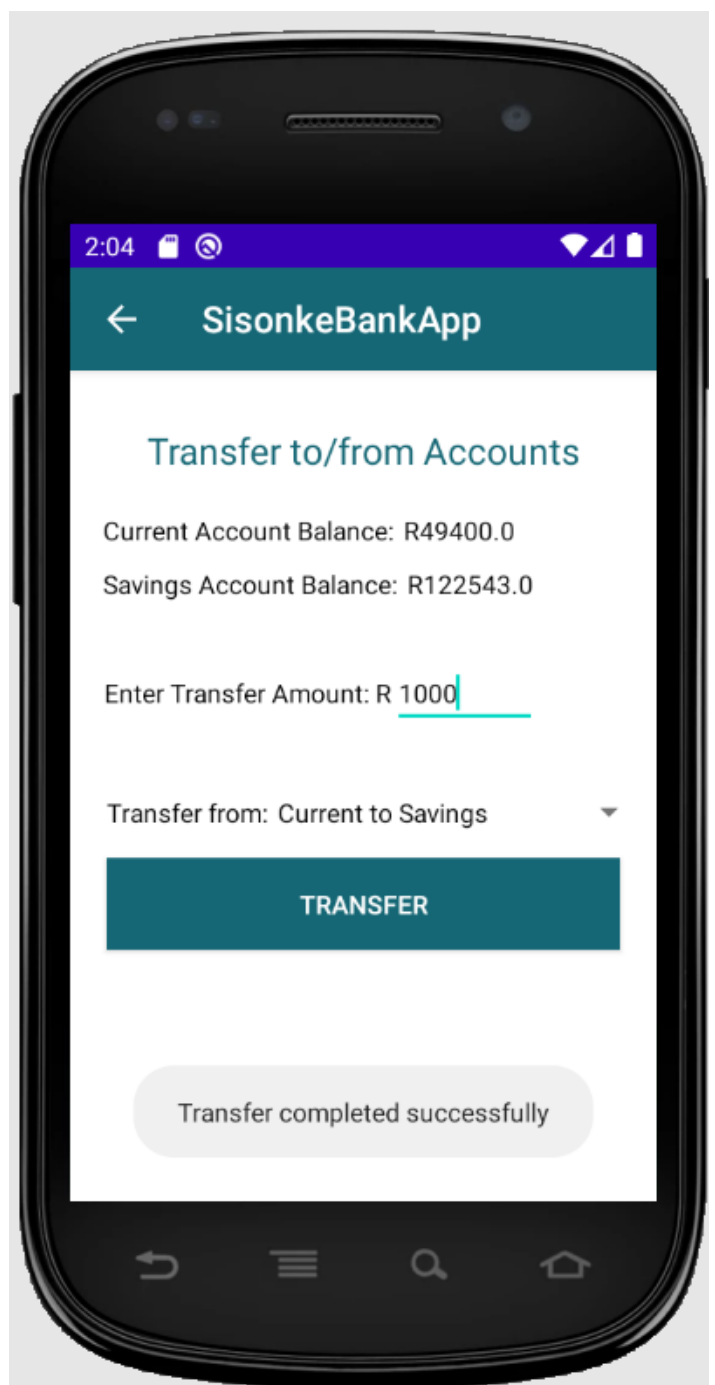
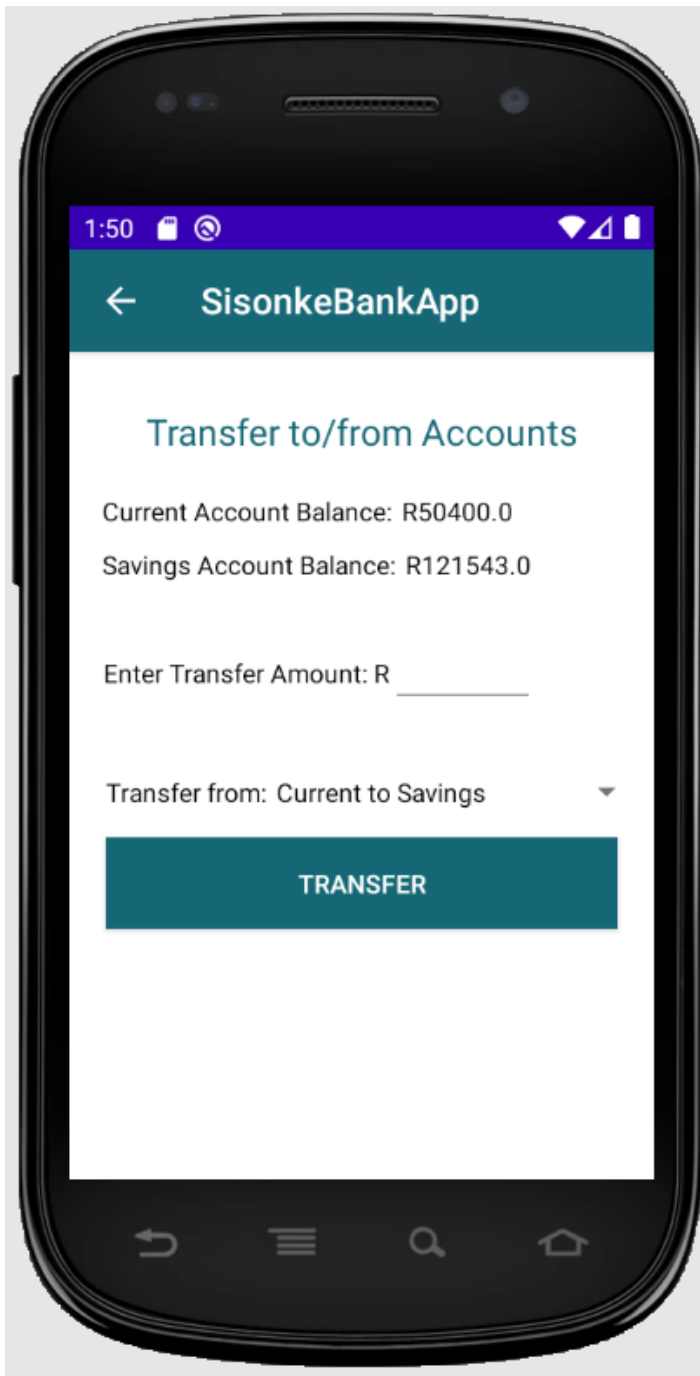
Below: The transfer between accounts screen. The user's balances are retrieved from the database. The user can enter an amount to transfer in the EditText field. This field only accepts numbers. The spinner widget allows the user to select one of two transfer options. When the 'TRANSFER' button is pressed, the entered amount is transferred from one account to the other. Also, this activity has a back arrow which directs the user back to the Main Page activity when pressed.



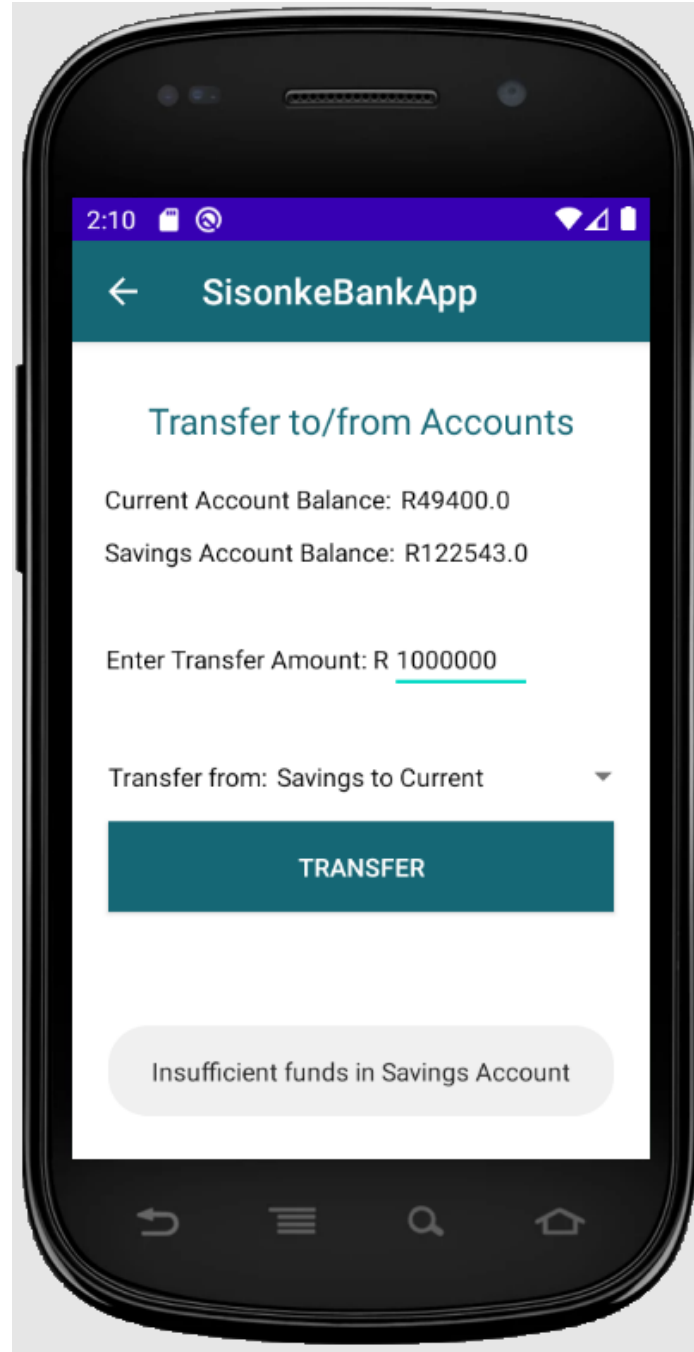
Below: This activity uses validation to ensure that a value has been entered to transfer into the EditText field. If no value has been entered and the 'TRANSFER' button is pressed, a toast message appears. This toast message informs the user to enter an amount.



Below: In this example, R1000 is transferred from the user's current account to his/her savings account. A toast message appears to inform the user that the transfer was successful. The balances of both accounts are updated in the database.



Below: This activity ensures that a user cannot transfer more money than what is available. If the user tries to do this, a toast message appears. This toast message informs the user that he/she does not have sufficient funds available to perform the transaction.



## Additional Code

### AndroidManifest.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bankingapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".TransferBetweenAccountsActivity"> </activity>
        <activity android:name=".ViewAccountBalanceActivity"> </activity>
        <activity android:name=".MainPageActivity" />
        <activity android:name=".RegistrationActivity" />
        <activity android:name=".LoginActivity"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

## References:

1. Erkec, E. 2020. Learn To Write Basic SQL Queries. [Online] sqlshack.com. Available at: <<https://www.sqlshack.com/learn-to-write-basic-sql-queries/>> [Accessed 3 September 2020].
2. Izuchukwu, C. 2017. What Are Android Intents?. [Online] tutspus.com. Available at: <<https://code.tutspus.com/tutorials/what-are-android-intents--cms-29335>> [Accessed 3 September 2020].
3. Kamel, M. 2018. Tips For Neater Android Development. [Online] proandroiddev.com. Available at: <<https://proandroiddev.com/tips-for-neater-android-development-part-1-34d3250b7943>> [Accessed 3 September 2020].
4. Khan, B. 2015. Sqliteopenhelpr Tutorial – Performing CRUD In Sqlite. [Online] simplifiedcoding.net. Available at: <<https://www.simplifiedcoding.net/sqliteopenhelpr-tutorial/>> [Accessed 3 September 2020].
5. Maheshwari, M. 2020. How To Add Custom Spinner In Android?. [Online] geeksforgeeks.org. Available at: <<https://www.geeksforgeeks.org/how-to-add-custom-spinner-in-android/>> [Accessed 3 September 2020].
6. Muntenescu, F. 2018. Spantastic Text Styling With Spans. [Online] medium.com. Available at: <<https://medium.com/androiddevelopers/spantastic-text-styling-with-spans-17b0c16b4568>> [Accessed 3 September 2020].
7. Nautiyal, D. 2018. The Application Manifest File | Android. [Online] geeksforgeeks.org. Available at: <<https://www.geeksforgeeks.org/application-manifest-file-android/>> [Accessed 3 September 2020].
8. Ogbo, O. 2016. Using A Simple Sqlite Database In Your Android App. [Online] androidauthority.com. Available at: <<https://www.androidauthority.com/creating-sqlite-databases-in-your-app-719366/>> [Accessed 3 September 2020].
9. Poetker, B. 2019. 17 Android Development Tips From The Pros. [Online] learn.g2.com. Available at: <<https://learn.g2.com/android-development-tips>> [Accessed 3 September 2020].
10. Reddy, A. 2018. How Many Types Of Intent Are In Android?. [Online] tutorialspoint.com. Available at: <<https://www.tutorialspoint.com/how-many-types-of-intent-are-in-android>> [Accessed 3 September 2020].
11. Sha, A. 2020. 7 Best Android Emulators For Windows You Can Use. [Online] beebom.com. Available at: <<https://beebom.com/best-android-emulators-windows/>> [Accessed 3 September 2020].
12. Shekhar, A. 2019. Learning Android Development In 2020 - A Practical Guide. [Online] medium.com. Available at: <<https://medium.com/mindorks/learning-android-development-in-2019-a-practical-guide-ddc71e008696>> [Accessed 3 September 2020].
13. Singh, C. 2020. 12 Best Android Emulators For Windows PC And Mac. [Online] fossbytes.com. Available at: <<https://fossbytes.com/best-android-emulators-pc/>> [Accessed 3 September 2020].
14. Singh, R. 2018. Java ArrayList Tutorial With Examples. [Online] callicoder.com. Available at: <<https://www.callicoder.com/java-arraylist/>> [Accessed 3 September 2020].
15. Stroud, A. 2016. Working With Databases In Android. [Online] informit.com. Available at: <<https://www.informit.com/articles/article.aspx?p=2731932&seqNum=4>> [Accessed 3 September 2020].
16. Vogel, L. 2020. Android Intents - Tutorial. [Online] vogella.com. Available at: <<https://www.vogella.com/tutorials/AndroidIntent/article.html>> [Accessed 3 September 2020].