Project

Faculty Name:	Information Technology
Module Code:	ITJA321
Module Name:	Java and Distributed Systems Semester 2
Module Leader:	Mr Sheunesu Makura
Internal Moderator:	Dr Michael Ajayi
Copy Editor:	Mr Kevin Levy
Total Marks:	200
Submission Date:	19/10/2020 – 23/10/2020
Resources Required:	Android Studio

This module is presented on NQF level 7.

Mark deduction of 5% per day will be applied to late submission, up to a maximum of three days. Projects submitted later than three days after the deadline or not submitted will get 0%. [1]

This is an individual project.

This project contributes 10% towards the final mark.

Instructions to Student

1. Remember to keep a copy of all submitted projects.
2. All work must be typed.
3. All work must be submitted through Turnitin[2] and the full Originality Report should be attached to the final assignment. Negative marking will be applied if you are found guilty of plagiarism, poor writing skills or if you have applied incorrect or insufficient referencing.

[1] Under no circumstances will assignments be accepted for marking after the assignments of other students have been marked and returned to the students.
[2] Refer to the PIHE Policy for Intellectual Property, Copyright and Plagiarism Infringement, which is available on *my*LMS.

(See the table at the end of this document where the application of negative marking is explained.)

4. Each project must include a cover page, table of contents and full bibliography, based on the referencing method applicable to your faculty as applied at Pearson Institute of Higher Education.
5. Use the cover sheet template for the project; this is available from *my*LMS.
6. Students are not allowed to offer their work for sale or to purchase the work of other students. This includes the use of professional assignment writers and websites, such as Essay Box. If this should happen, Pearson Institute of Higher Education reserves the right not to accept future submissions from a student.

Assignment Format

Students must follow the requirements when writing and submitting assignments as follows:

- Use Arial, font size 10.
- Include page numbers.
- Include a title page.
- Print submissions on both sides of the page.
- Write no more than the maximum word limit.
- Ensure any diagrams, screenshots and PowerPoint presentations fit correctly on the page and are referenced.
- Include a table of contents.
- Use the accurate referencing method throughout the assignment.
- Include a bibliography based on the applicable referencing method at the end of the assignment.
- Include the completed Assessment/Project Coversheet (available on *my*LMS).
- Check spelling, grammar and punctuation.
- Run the assignment through Turnitin software.

Essential Embedded Knowledge and Skills Required of Students

- Report-writing skills
- Ability to analyse scenarios/case studies
- Understanding of subject field concepts and definitions
- Ability to apply theoretical knowledge to propose solutions to real-world problems
- Referencing skills

Resource Requirements

- A device with Internet access for research
- A desktop or personal computer for typing assignments
- Access to a library or resource centre
- Prescribed reading resources

Delivery Requirements (evidence to be presented by students)

- A typed assignment[3]
- A Turnitin Originality Report

Minimum Reference Requirements

At least five references for first year, ten references for second year and fifteen references for third year.

Additional reading is required to complete this assignment successfully. You need to include the following additional information sources:

- Printed textbooks/e-books
- Printed/online journal articles
- Academic journals in electronic format accessed via PROQUEST or other databases
- Periodical articles e.g. business magazine articles
- Information or articles from relevant websites
- Other information sources e.g. geographic information (maps), census reports, interviews

Note
• It is crucial that students reference all consulted information sources, by means of in-text referencing and a bibliography, according to the applicable referencing method. • Negative marking will be applied if a student commits plagiarism i.e. using information from information sources without acknowledgement and reference to the original source. • In such cases, negative marking, also known as 'penalty scoring', refers to the practice of subtracting marks for insufficient/incorrect referencing. • Consult the table at the end of this document, which outlines how negative marking will be applied as well as the way in which it will affect the assignment mark.

[3] Refer to the Conditions of Enrolment for more guidance (available on *my*LMS).

Section A

Learning Objective

This project focuses mainly on LOs 1, 2 and 6. Using Java, XML and SQLite, students are required to come up with an individual Android based project as specified in the questions below. In this project, students must demonstrate an understanding of inheritance, polymorphism and apply the concept of parameter passing of Java objects. Furthermore, the project requires the students to apply their knowledge of intents, usage of `SQLiteOpenHelper` & `SQLiteCursor` classes and SQL execution.

Project Topic

Android Development

Scope

The project looks into developing an Android app that makes use of an SQLite database. You should focus on how to perform CRUD operations in Android using Java. You should also focus on how to use intents to handle movement from one activity to another in an Android application.

Technical Aspects

You should use Harvard referencing for the discussion question.

Marking Criteria

Marks will be awarded for the following:

- Detailed discussion with correct referencing of sources. Note, you must have at least three references for the discussion question.
- Originality - evidence of your own ideas
- Development of a fully functional Android application according to specifications.

Question 1 — 20 Marks

1.1 Discuss intents and their uses in Android development. Make sure you mention the two types of intents in your discussion. (10 Marks)

1.2 Give a description of the `SQLiteOpenHelper` and `SQLiteCursor` classes specifically their uses in performing CRUD operations. (10 Marks)

[Sub Total 20 Marks]

End of Question 1

Question 2 — 180 Marks

Note: The following question requires the use of **Android Studio**. Ensure that you have it installed and have a working Android emulator or alternatively an Android Phone to test your application.

Study the scenario and complete the question(s) that follow:

Android Development of a Bank App

Sisonke Bank is a new and emerging bank in South Africa. They recently recruited you as a Graduate Android Developer after completing your studies at the Pearson Institute of Higher Education. They have given you the task of developing the first prototype of their bank application. The bank application should allow Sisonke's bank clients using Android, to create an account, login, view account balances and transfer between accounts. The Senior Android Developer at Sisonke has provided you with a top-of the range computer installed with Android Studio.

Source: Makura S.M (2020)

Using the information provided in the above scenario, create an Android application for Sisonke Bank. The app must consist of the following activities:

- **Registration**: this is where the user is able to register to make use of the app.
- **Login**: this is where the user logs in using the user details
- **Main page**: the interface seen after the user logs in. It should consist of three other activities stated below
- **View account balance**: this should show details of the balances of the various accounts the user has
- **Transfer funds**: this should provide the functionality to the user to transfer money from one account to another
- **Logout**: this should allow the user to logout of the app

When the app launches, it should launch the login activity as shown in the interface below:

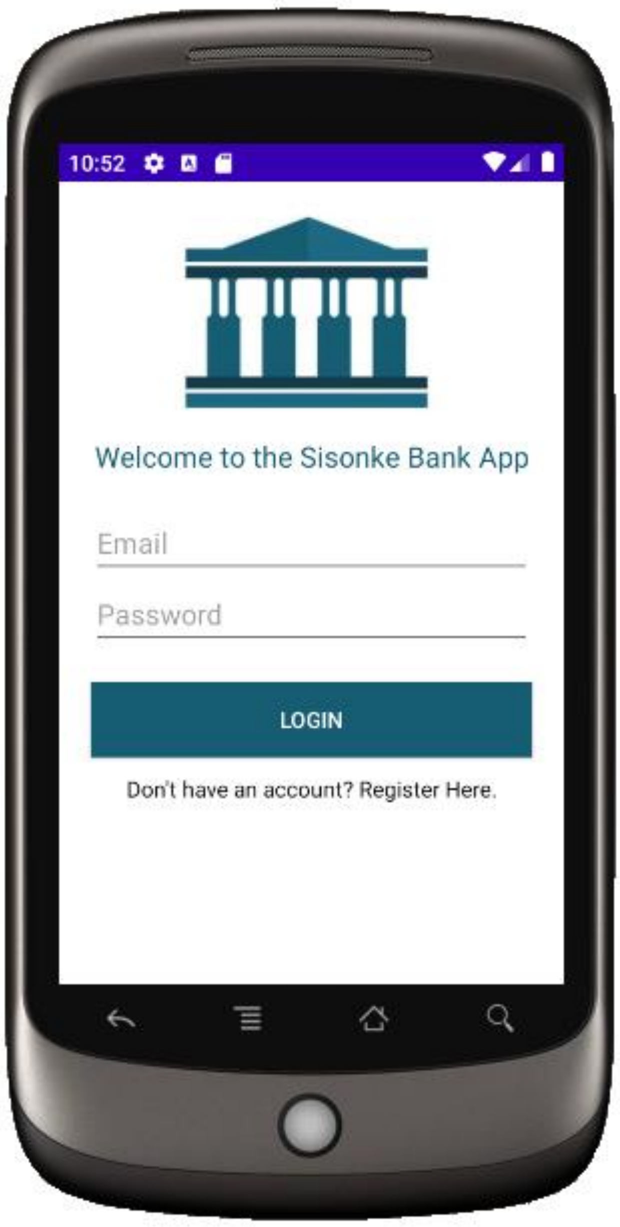

10:52
Welcome to the Sisonke Bank App
Email
Password
LOGIN
Don't have an account? Register Here.

Figure 1- Login Activity User Interface

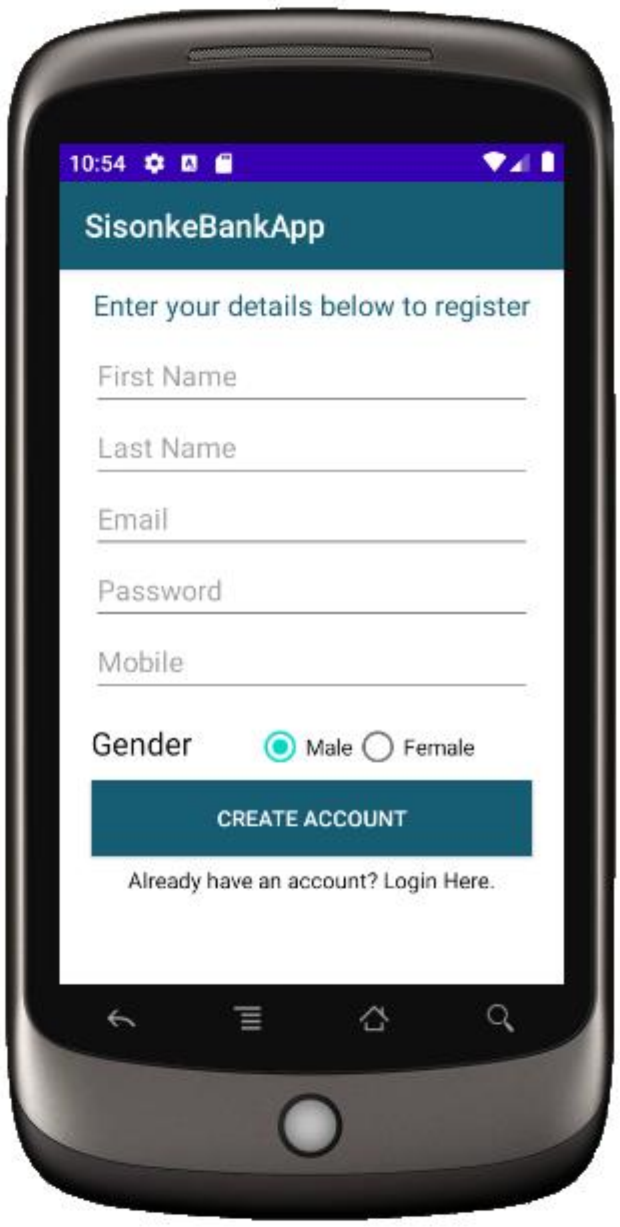


Figure 2- Register Activity User interface

Login Activity (Figure 1):

Take note of the following regarding the login activity:

- The logo picture can be a picture of your choice.
- It should handle validation to ensure that a user enters an email and password before logging in. Otherwise, a toast message should be displayed to the user to inform the user to enter these before logging in.
- The password must be hidden when it is being entered.
- There should be validation to ensure that the email entered is in the correct format.
- There should be validation to ensure that the password entered is at least 5 characters long.

- There should be validation in the event that either the user name or password is incorrect. A toast message should be displayed to the user if either one of them is incorrect.
- The login page should be linked to an SQLite database (see SQLite Database section below) which should authenticate the user.
- There should be a `TextView` widget on the login page that allows a new user who does not have an account to register (see Figure 1). Once clicked, it should direct the user to the registration page (make use of intents).
- After a successful login, a toast message should be displayed to inform the user that the login was successful and the app should direct the user to the main UI (make use of intents).

(23 Marks)

Registration Activity (Figure 2)

Take note of the following regarding the registration activity:

- As with the login activity, the registration activity should also handle validation of all fields to ensure that the user enters all details required before creating an account. Otherwise, a toast message should be displayed to the user to inform the user to enter these before creating an account.
- The `EditTextfield` for the mobile number should accept numbers only.
- There should be validation to ensure that the email entered is in the correct format.
- There should be validation to ensure that the password entered is at least 5 characters long.
- The password must be hidden when it is being entered
- Once all fields are entered correctly, the app should create an account for the user. To do this, it should make use of an SQLite database which is to be used to store the user details (see the SQLite database section for more information regarding the database).
- The register activity should check if the email entered does not already exist in the database before allowing a user to register. Make use of a method to do this (see database section for more information)
- If a user account has been successfully created, the app should automatically redirect the user to the login activity and display a toast message the user account creation was successful.
- To simulate a bank deposit, ensure that during account creation, you create two variables used store values for the balances for two accounts, namely (i) current account (ii) savings account. The values can be any amount of your choice.

- There should be a `TextView` widget on the registration UI that allows a user which already have an account to login (Figure 2). Once clicked, it should direct the user to the login page (make use of intents).

(40 Marks)

SQLite Database:

- Create an SQLite database with a name of your choice. To do this, create an `SQLiteOpenHelper` class which will handle the database and tables creation. In this project, you will perform the CRU (Create, Read & Update) operations.
- Your database should have a table called users to store user details entered on the registration activity. Make use of an `addUser()` method to add user details to the database.
- Your database should have a method to check if an email entered on the registration activity already exists in the database. If it does, it should not allow the user to register with that email. Make use of the `Cursor` implementation that exposes results from a query on a `SQLiteDatabase.`
- Your database should also have a method to authenticate the user who tries to login from the login UI. Make use of the `Cursor` implementation that exposes results from a query on a `SQLiteDatabase`.
- The database should have a `getUserDetails()` method which gets specific user details that might be requested from a particular activity. Note the Main, View Account Balance and Transfer Funds Activities will need this method. Make use of the `Cursor` implementation that exposes results from a query on a `SQLiteDatabase`.
- Your database should also have a method to update the balance of the current and savings account. This is explained later on the Transfer Funds Activity
- To aid the database, create a Bank User class with a constructor, with get and setter methods. Use these to handle transfer of data to and from the login/registration activity to the SQLite database.

(52 Marks)

Once a user logs in, the main UI should look as follows (Figure 3):

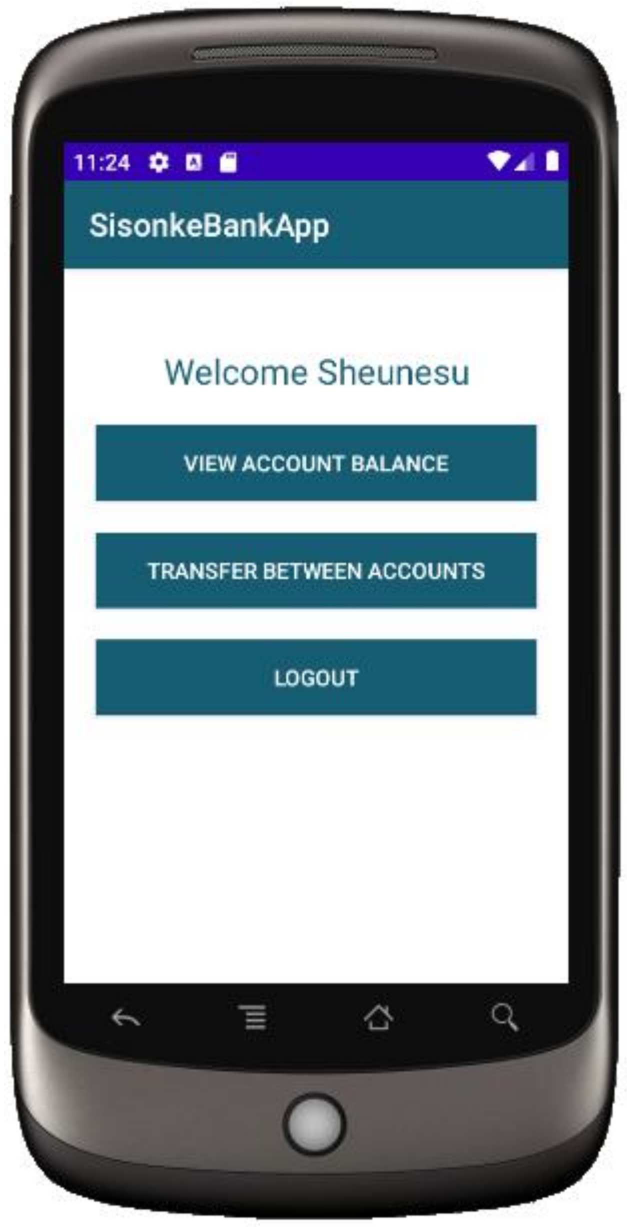
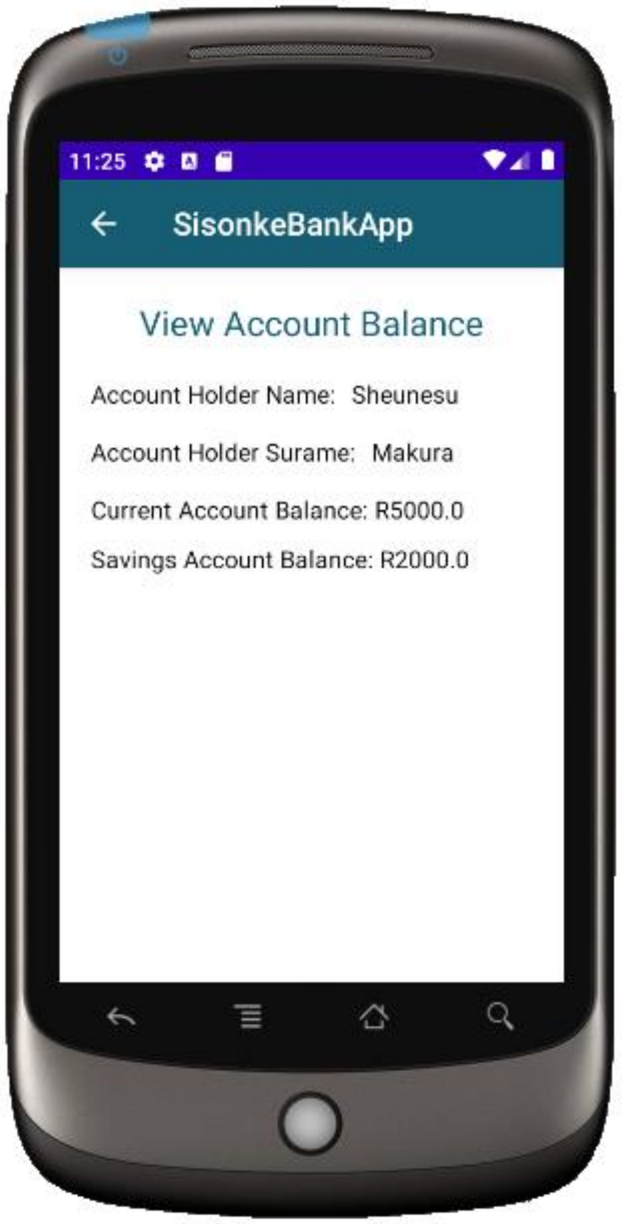


Figure 3- Main Page Activity UI



Account Holder Name: Sheunesu

Account Holder Surame: Makura

Current Account Balance: R5000.0

Savings Account Balance: R2000.0

Figure 4- View Account Balance Activity UI

Main Page Activity:

Take note of the following regarding the main page activity:

- Once a user logs in, there should be a method to get user details, specifically the first name from the database and display it in a `TextView` widget in form of a welcome message as shown in Figure 3. Make use of the `getUserDetails()` method in your `SQLIiteOpenHelper` class to get these details.
- The view account balance and transfer between accounts should direct user to a view account balance (Figure 4) and a transfer between accounts page (Figure 5). Make use of intents to handle the movement between one activity to another.

- The logout button should simply logout the user and direct him/her to the login activity (make use of intents). Ensure that once logged out, the user cannot press the back button to simply log back in.
- Once successfully logged out, a toast message should be displayed to inform the user that he/she has been logged out.

(16 Marks)

View Account Balance Activity:

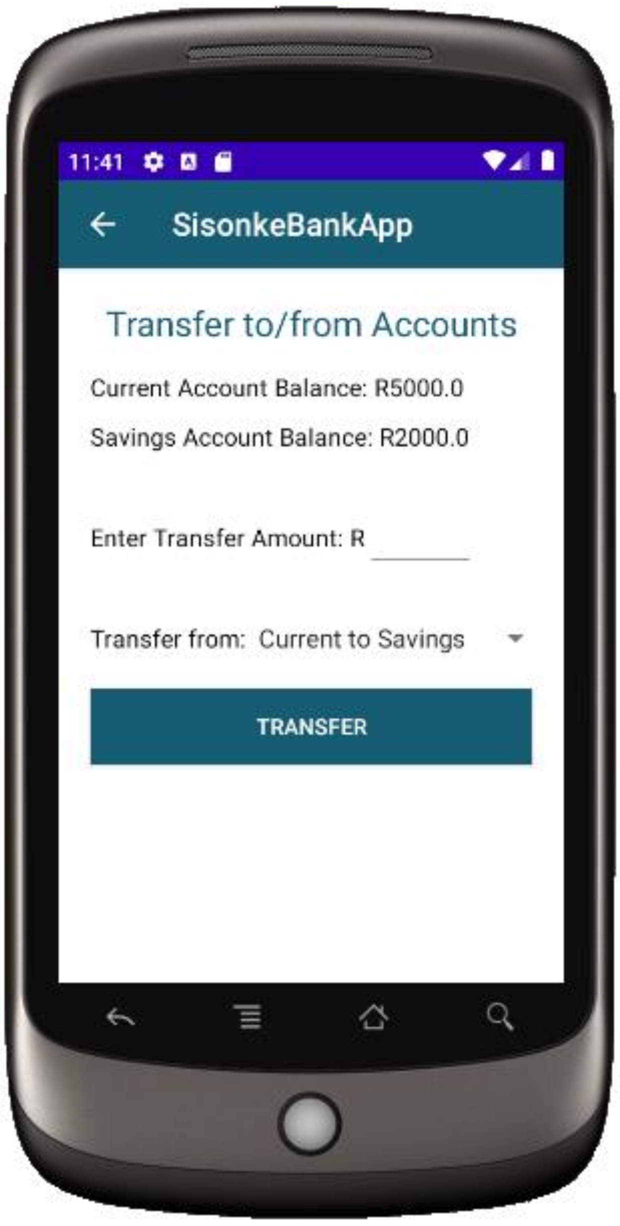
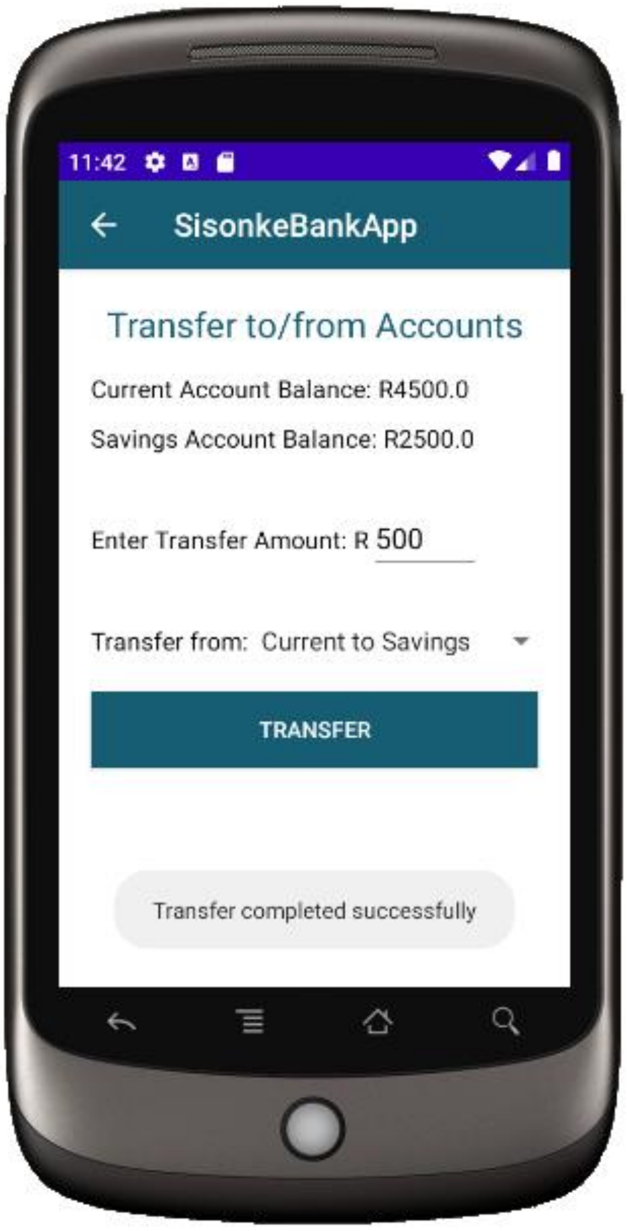
Take note of the following regarding the View account balance activity:

- This activity should simply retrieve user details from the database specifically the account holder name, surname, current account balance & savings account balance (Figure 4). Make use of the `getUserDetails()` method in your `SQLIiteOpenHelper` class to get these details. **Note:** the amounts of the accounts should be created and intialised during the registration activity to ensure that they can be displayed on this activity.
- All details can be displayed in `TextView` widgets. You are welcome to use other widgets if need be.
- The activity should have a back arrow (Figure 4) which, when tapped, allows the user to go back to the Main Page activity.

(11 Marks)

Transfer between accounts Activity

This activity should have the following user interface (Figure 5):

 11:41 SisonkeBankApp Transfer to/from Accounts Current Account Balance: R5000.0 Savings Account Balance: R2000.0 Enter Transfer Amount: R Transfer from: Current to Savings TRANSFER	 11:42 SisonkeBankApp Transfer to/from Accounts Current Account Balance: R4500.0 Savings Account Balance: R2500.0 Enter Transfer Amount: R 500 Transfer from: Current to Savings TRANSFER Transfer completed successfully
Figure 5- Transfer Between Accounts UI	**Figure 6- Toast message displayed after a successful transfer**

Take note of the following regarding the Transfer between accounts activity:

- The Transfer between accounts activity should initially get the current and savings account balances from the database (Make use of the `getUserDetails()` method in your `SQLIiteOpenHelper` class to get these details).These should then be displayed on the UI using `TextView` widgets. You are welcome to use other widgets of your choice.
- There should be an `EditText` field that allows the user to specify the amount to transfer. This field when selected should allow only numbers to be entered

- There should be a `Spinner` widget which gives two options (i) Transfer from current to savings (ii) Transfer from savings to current.
- Once the transfer button is tapped, the specified amount must be transferred from one account to the other based on the option specified on the spinner. For example, as shown in Figure 6, R500 was transferred from the current account to the savings account Make use of a method to handle transfer between the accounts.
- There should be validation to ensure that the `EditText` field has a value entered before the transfer button is clicked. Otherwise, a toast message should be displayed to inform the user of this.
- Once a transfer is successful, a toast message should be displayed to inform the user that the transfer was successful.
- The activity should then update the current and savings account balances in the database and show the new balances on the same activity (Figure 6). Hint, make use of an `updateBalance()` method in your `SQLIiteOpenHelper` class to handle updating of the database entries.
- The activity should ensure that that a user cannot transfer more funds than what is already available. If this is the case, a toast message should be displayed to inform the user that this operation is not possible.
- The activity should have a back arrow (Figure 5/6) which, when tapped, should allow the user to go back to the Main Page activity.

(38 Marks)

[Sub Total 180 Marks]

Once you are done with your project, zip your project files (including the apk file) and submit it with your documentation which should contain screenshots and your source code.

PS: If you do very well in this project, you can include it in your portfolio which will increase the chances of you getting hired as a Graduate/Junior Android Developer once you are done with your studies 😊. Good luck!

End of Question 2

Section B

Plagiarism and Referencing

Pearson Institute of Higher Education places high importance on honesty in academic work submitted by students, and adopts a policy of zero tolerance on cheating and plagiarism. In academic writing, any source material e.g. journal articles, books, magazines, newspapers, reference material (dictionaries), online resources (websites, electronic journals or online newspaper articles), must be properly acknowledged. Failure to acknowledge such material is considered plagiarism; this is deemed an attempt to mislead and deceive the reader, and is unacceptable.

Pearson Institute of Higher Education adopts a zero-tolerance policy on plagiarism, therefore, any submitted assessment that has been plagiarised will be subject to severe penalties. Students who are found guilty of plagiarism may be subject to disciplinary procedures and outcomes may include suspension from the institution or even expulsion. Therefore, students are strongly encouraged to familiarise themselves with referencing techniques for academic work. Students can access the PIHE Guide to Referencing on *my*LMS.

Negative Marking

Third-year Students

- A minimum of 15 additional information sources must be consulted and correctly cited.
- If no additional information sources have been used, a full 15% must be deducted.
- Deduct 1% per missing resource of the required 15. For example:
- If only five resources cited, deduct 10%.
- If only three resources cited, deduct 12%.
- Markers to apply the penalties for Category A for insufficient sources and incorrect referencing style.
- To determine the actual overall similarity percentage and plagiarism, markers must interpret the Turnitin Originality Report with reference to credible sources used and then apply the penalties as per the scale in the PIHE Policy for Intellectual Property, Copyright and Plagiarism Infringement.
- The similarity report alone is not an assessment of whether work has or has not been plagiarised. Careful examination of both the submitted paper/assignment/project and the suspect sources must be done.

Category A

Minimum reference requirements	Deduction of final mark
No additional information sources have been used or referenced.	15%