(GROUP 27)
DILENDRA (190101034)
EKESHWAR (190101036)
KSHITIJ (190101046)
VIVEK (190101100)

Project – Business Management Software
Code Testing Document

1. Mapping between DFD and functions:-

- **module1**
  - module1.1 → void createNewAccount(String userEmail, String password)
  - module1.2 → void loginUserAccount(String userEmail, String password)
  - module1.3
    - module1.3.1 → void changePassword(String userEmail, String oldPassword, String newPassword)
    - module1.3.2 → void forgotPassword(String userEmail)
    - module1.3.3 → void setNewPassword(String userEmail, int OTP, String newPassword)

- **module2**
  - module2.1 → List<ObjStockRecord> viewStockData()
  - module2.2 → ObjStockRecord searchStockItem(String itemName)
  - module2.3
    - module2.3.1 → void addNewItem(String itemName, int itemPrice, int quantity)
    - module2.3.2 → void updateItemDetails(String itemName, String newItem, String price)
    - module2.3.3 → void deleteItem(String itemName)
  - module2.4 → List<ObjPurchaseData> viewPurchaseList()
  - module2.5 → void updatePurchaseList(String itemName, int quantity)

- **module3**
  - module3.1 → void addSoldItemDetail(String itemName, int quantity)
  - module3.2 → List<ObjSalesRecord> viewSalesRecord(String itemName, Date date)
  - module3.3 → void addPurchasedItem(String itemName, int quantity, boolean paymentStatus)

- module3.4 → List<ObjPurchaseRecord> viewPurchaseRecord(String itemName, Date date)

- **module 4**
  - module4.1 → void addRecord(String customerName, int amount, int phoneNumber)
  - module4.2 → void updateRecord(String customerName, int amount, int phoneNumber)
  - module4.3 → void deleteRecord(String customerName, int phoneNumber)
  - module4.4 → ObjCustomerRecord searchRecord(String customerName)
  - module4.5 → List<ObjDebtRecord> showDebtRecord()

- **module 5**
  - module5.1 → void createNewEmployee(String employeeName, String fathersName, int age, String phoneNumber, String address)
  - module5.2 → void deleteEmployee(String employeeName)
  - module5.3 → void markEmployeeAttendance(String employeeName, boolean attendanceStatus)
  - module5.4 → List<ObjAttendanceReport> viewAttendanceRecord(String employeeName)

2. Choosing Three Important Functions:-

We have selected following three functions corresponding to three most important use cases of our system for code testing:-

1. loginUserAccount
2. searchStockItem
3. addSoldItemDetail

Reason for choosing above mentioned functions:-

1. **loginUserAccoun**t function logs in the user to the system's home page. It is important as the user needs to login first in order to access the system. So it is a frequent use case function of our system. Also it is important for security of the software.

2. **searchStockItem** function searches for a particular stock item in the database. Whenever a customer comes and asks for any item, the first thing that the shopkeeper need to know is whether the item is available or not. So this is a necessary function related to another most important use case.

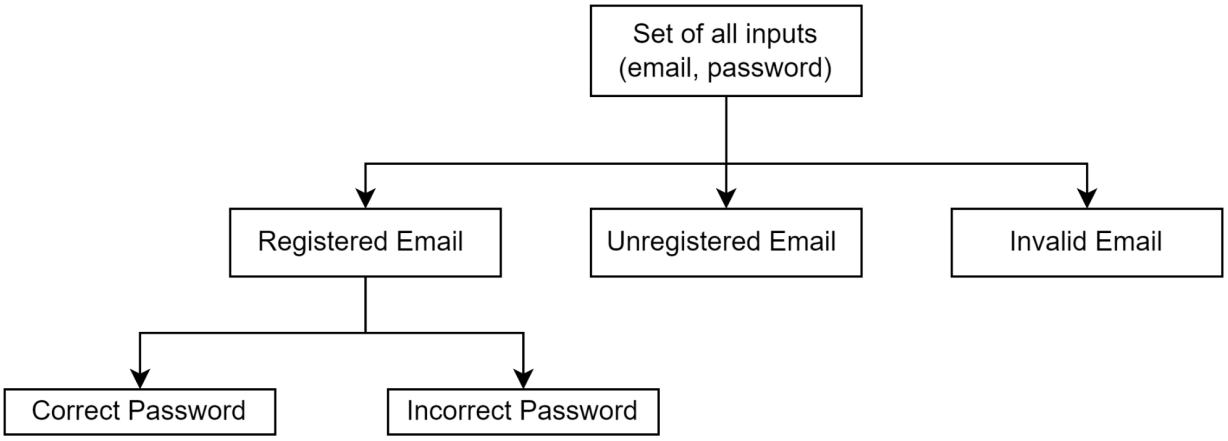
3. **addSoldItemDetail** function is used to update stock quantity in the database and the overall account balance as per the sold item detail. It is also an frequently used feature of the system as stock details need to be updated after each shopping activity and so the total account balance.

Note:- The snapshot of code is added in the white box testing section where they are required. Also for the testing, we have added some recording of our implemented app for ease. We have implemented all those above mentioned basic features in our app.

Black Box Testing Report

Function 1: loginUserAccount

The equivalence classes for the function are the leaf level classes shown in the below figure:-



Equivalence classes: Valid Password, Invalid Passwords, Unregistered Email, and Invalid Email (emails without @gmail.com suffix).
Now selecting one representative value from each equivalence class, we have the required set of test suits {((input), output)} :-
{((ek@gmail.com, 12345678910@e), homePage), ((ek@gmail.com, 12345678901), loginFailed), ((vivek@gmail.com, 01#), loginFailed), ((ek#mail.com, 23!), loginFailed)}

Boundary Value Analysis:-
The boundary values for this function are not defined as we are not using statements with comparison based on input.

Overall Test Suite For Function:-
{((ek@gmail.com, 12345678910@e), homePage), ((ek@gmail.com, 12345678901), loginFailed), ((vivek@gmail.com, 01#), loginFailed), ((ek#mail.com, 23!), loginFailed)}

Testing:-
Youtube Link:- https://youtu.be/kXTyL1LQkpo

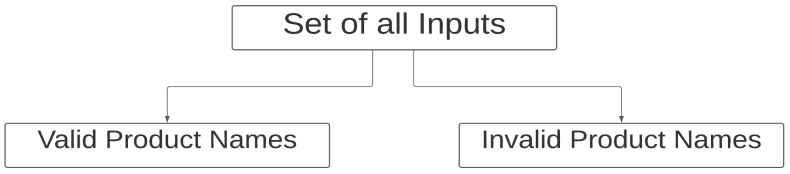
Test suit 1:-
Input: ek@gmail.com, 1234567910@e
Expected Output: homePage
System Output: homepage

Test suit 2:-
Input: ek@gmail.com, 12345678901
Expected Output: Login Failed Message
System Output: Login Failed Message

Test suit 3:-
Input: vivek@gmail.com, 01#
Expected Output: Login Failed Message
System Output: Login Failed Message

Test suit 4:-
Input: ek#mail.com, 23!
Expected Output: Login Failed Message
System Output: Login Failed Message

Function 2: searchStockItem
The equivalence classes are the leaf level classes shown in the below figure: -



Equivalence classes: Valid Product Names (Product already created by the user) and Invalid Product Name (Product not created by the user).
Selecting one representative value from each equivalence class, we have the required test suite{(input, output)}:
{(pen, StockData), (chair, Error)}
Note: Consider pen is stored in the database as a product and chair is not.

Boundary Value Analysis:-

The boundary values for this function are not defined as we are not using statements with comparison based on input.

Overall Test Suite For Function:-
{(pen, StockData), (chair, No Record Message)}

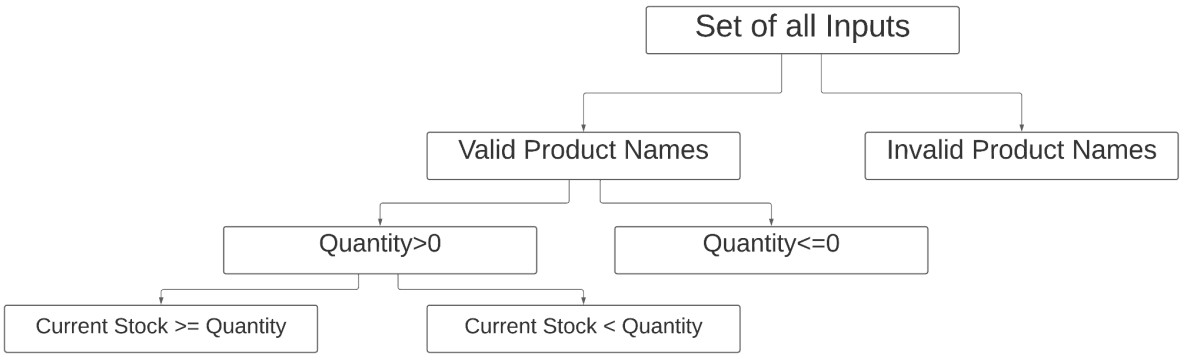
Testing:-
Youtube Link:- https://youtube.com/shorts/hsXIcNrF0i4?feature=share
Test suit 1:-

Input: pen
Expected Output: StockData
System Output: StockData

Test suit 2:-

Input: chair
Expected Output: No Record Message
System Output: No Record Message

Function 3: addSoldItemDetail
The equivalence classes are the leaf level classes shown in the below figure -



Equivalence classes: CurrentStock >= Quantity, CurrentStock < Quantity, Quantity <= 0 and Invalid Product Names.
Selecting one representative value from each equivalence class, we have the required test suite{((input), output)}:-
{((toothpaste, -1), fail message), ((toothpaste, 2), success message), ((toothpaste, 60), fail message), ((toothpaste, 0), fail message), ((table, 2), fail message)}
Note: toothpaste is a valid product name with available quantity 30 and table is not in stock so an invalid product name.

Boundary Value Analysis:-

The boundary values for this function are all the values of quantity of items that are at boundary.

Boundary Test Suit:-

{((toothpaste, 0), fail message), ((toothpaste, 30), success message), ((toothpaste, 31), fail message)}

Overall Test Suite For Function:-

{((toothpaste, -1), fail message), ((toothpaste, 0), fail message), ((toothpaste, 2), success message), ((toothpaste, 30), success message), ((toothpaste, 31), fail message), ((toothpaste, 60), fail message), ((table, 2), fail message)}

Testing:-

Youtube Link:- https://youtu.be/V6YNEjvz2t8

Test suit 1:-

Input: toothpaste, -1
Expected Output: fail message
System Output: success message

Test suit 2:-

Input: toothpaste, 0
Expected Output: fail message
System Output: success message

Test suit 3:-

Input: toothpaste, 2
Expected Output: success message
System Output: success message

Test suit 4:-

Input: toothpaste, 30
Expected Output: success message
System Output: success message

Test suit 5:-

Input: toothpaste, 31
Expected Output: fail message
System Output: success message

Test suit 6:-

Input: toothpaste, 60
Expected Output: fail message

System Output: success message

Test suit 7:-

Input: table, 2

Expected Output: fail message

System Output: fail message

Failed Test Case List:-

Failed Test Suite (Input, Output)	Function Name
1. ((toothpaste, -1), fail message)	addSoldItemDetail
2. ((toothpaste, 0), fail message)	addSoldItemDetail
3. ((toothpaste, 31), fail message)	addSoldItemDetail
4. ((toothpaste, 60), fail message)	addSoldItemDetail

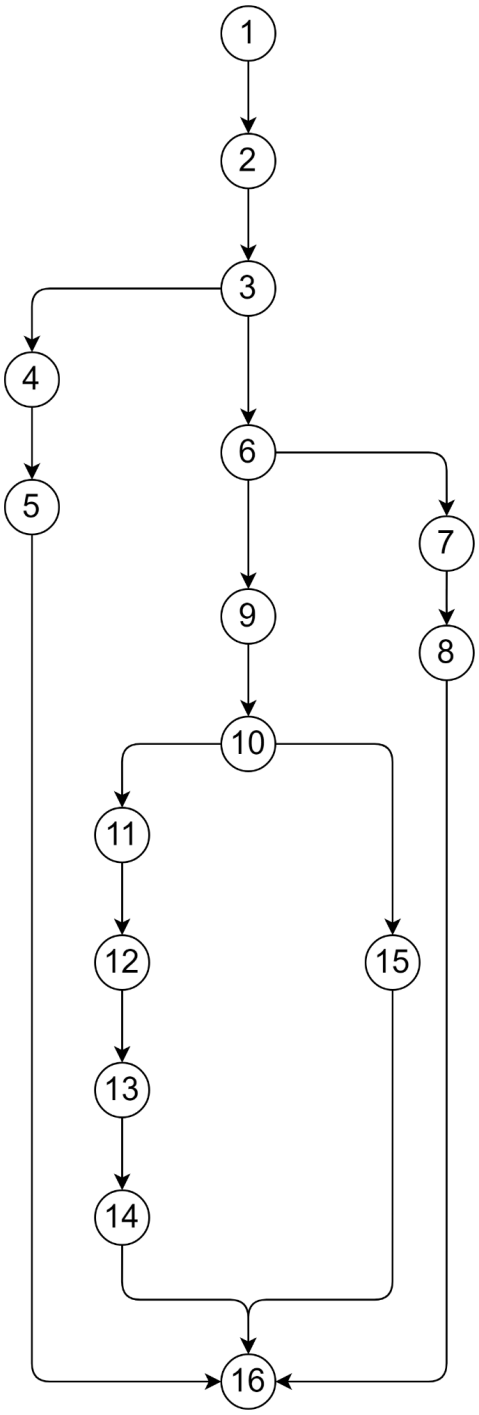
White Box Testing:-

Function 1: loginUserAccount

```
// Take email id and password from user
// and if it matches the already present id,password pair
// redirect the user to home page
  private void loginUserAccount()
  {
    // Take input from user into the edit text fields
    // and store them into string variables
1   String email = emailTextView.getText().toString();
2   String password = passwordTextView.getText().toString();

    // Check whether the email field is non-empty
    // if empty show a message to enter email
3   if (TextUtils.isEmpty(email)) {
4       Toast.makeText(getApplicationContext(),
                "Please enter email!!",
                Toast.LENGTH_LONG)
                .show();
5       return;
    }
```

```
    // Check whether the password field is non-empty
    // if empty show a message to enter email
    if (TextUtils.isEmpty(password)) {
        Toast.makeText(getApplicationContext(),
                "Please enter a password!!",
                Toast.LENGTH_LONG)
                .show();
        return;
    }

    // Check whether the email, password is valid
    // and if valid redirect the user to Home page
    mAuth.signInWithEmailAndPassword(email, password)
          .addOnCompleteListener(
                  task -> {
                      if (task.isSuccessful()) {

                              Toast.makeText(getApplicationContext(),
                                  "Login successful!!",
                                  Toast.LENGTH_LONG)
                                  .show();

                          // if sign-in is successful
                          // redirect the user to home page
                          Intent intent
                                = new Intent(LoginActivity.this,
                                BusinessActivity.class);
                         startActivity(intent);
                          finish();
                      }
                      else {

                          // sign-in failed show login fail message
                          Toast.makeText(getApplicationContext(),
                                  "Login failed!!",
                                  Toast.LENGTH_LONG)
                                  .show();
                      }
                  });
}
```

Control Flow Graph:-



Cyclomatic complexity:-

E = 18, V = 16, C = E - V + 2 = 4

Possible Paths are :-

1. 1→2→3→4→5
2. 1→2→3→6→7→8
3. 1→2→3→6→9→10→11→12→13→14→16
4. 1→2→3→6→9→10→15→16

Test suite for each path (input, output):-

1. ((, 12345678910@e), enter email message)
2. ((ek@gmail.com ,), enter password message)
3. ((ek@gmail.com , 12345678910@e), homepage)
4. ((ek@gmail.com , 1234@e), login failed message)

Testing:-

Youtube Link:- https://youtu.be/w5WpIEctXKE

Test suit 1:-

Input: , 12345678910@e

Expected Output: enter email message

System Output: enter email message

Test suit 2:-

Input: ek@gmail.com ,

Expected Output: enter password message

System Output: enter password message

Test suit 3:-

Input: ek@gmail.com , 12345678910@e

Expected Output: homepage

System Output: homepage

Test suit 4:-

Input: ek@gmail.com , 1234@e

Expected Output: login failed message

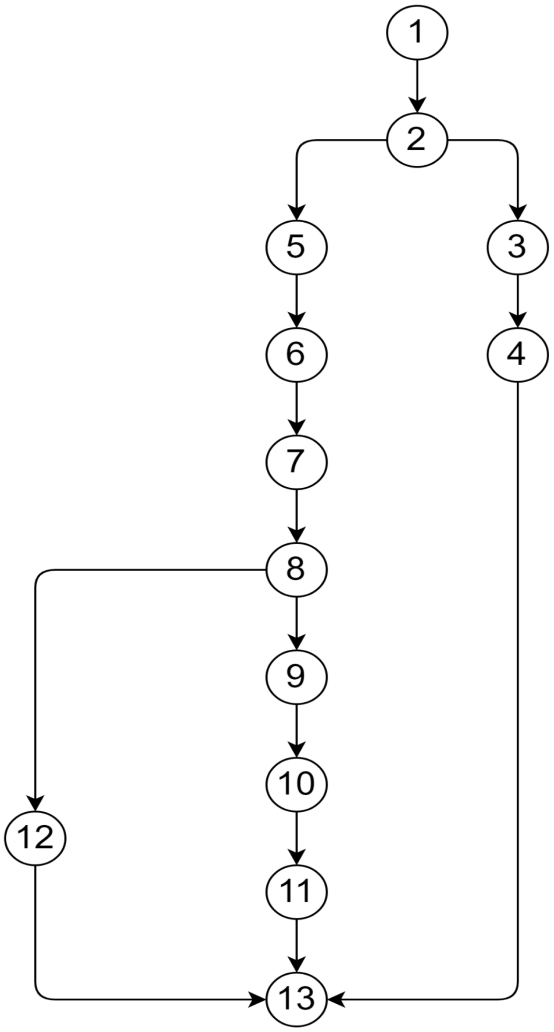
System Output: login failed message

Function 2: searchStockItem

```
  // Function to search whether an item is available in the store
  // and if available how much quantity is present
  private void searchStockItem() {

    // Take the input from user into edit fields and convert into  string
1   String stockName = fieldStockName.getText().toString();

    // Check whether the stock name field is non-empty
    // if empty show a message to enter it.
2   if (TextUtils.isEmpty(stockName)) {
3       Toast.makeText(getApplicationContext(),
                "Please Enter Stock Name!!",
              Toast.LENGTH_LONG)
                .show();
4       return;
    }
    // Find the stock item in the database using the entered name
    // if available show the details
5   DocumentReference docRef = db.collection("stocks").document(stockName);
6    docRef.get().addOnSuccessListener(documentSnapshot -> {

7       Stock stocks = documentSnapshot.toObject(Stock.class);

8       if(stocks != null && stocks.getStockName().equals(stockName)){
            // stock found show the details
```

```
9           textStockName.setText(String.format("Stock Quantity: %s",
stocks.getStockName()));
10          textStockQuantity.setText(String.format("Stock Quantity: %s",
stocks.getStockQuantity()));
11          textStockPrice.setText(String.format("Stock Price: %s",
stocks.getStockPrice()));
        }
        else{
            // stock not found show not found message
12          Toast.makeText(getApplicationContext(),
                "Sorry No Record Found With Name: "+ stockName +".",
                Toast.LENGTH_LONG).show();
        }
    });
13}
```

Control Flow Graph:-



Cyclomatic complexity:-

E = 14, V = 13, C = E - V + 2 = 3

Possible paths are:-

1. 1→2→3→4→13
2. 1→2→5→6→7→8→9→10→11→13
3. 1→2→5→6→7→8→12→13

Test suite for each path (input, output):-

1. (, enter stock name message)
2. (pen, StockData)
3. (chair, No Record Message)

Testing:-

Youtube Link:- https://youtube.com/shorts/Y4y7Ng42Ugc?feature=share

Test suit 1:-

Input:

Expected Output: stock data

System Output: stock data

Test suit 2:-

Input: pen

Expected Output: No Record Message

System Output: No Record Message

Test suit 3:-

Input: chair

Expected Output: No Record Message

System Output: No Record Message

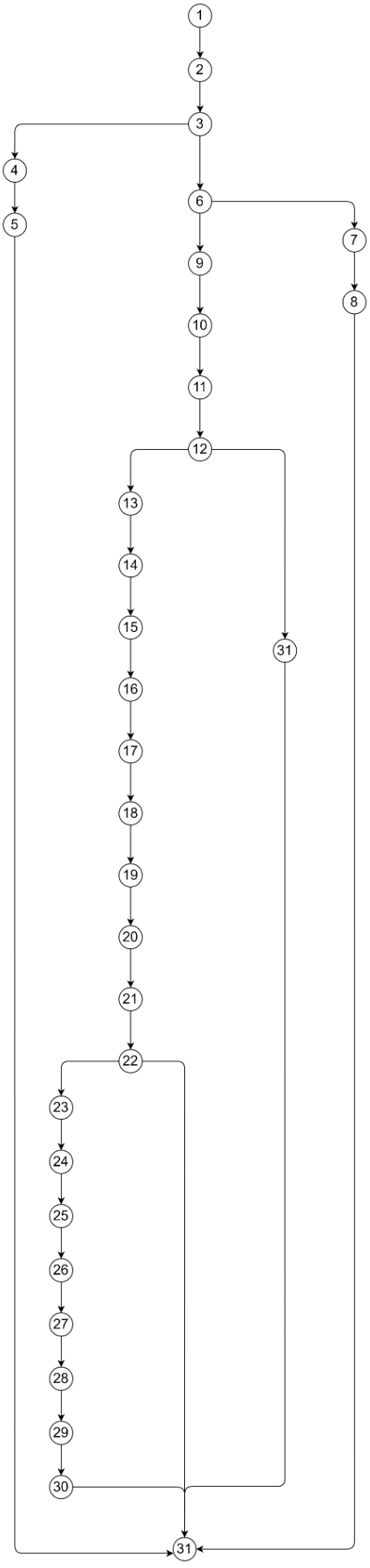
Function 3: addSoldItemDetail

```
  // Function to take details of sold item i.e., stock name and quantity
  // check for the item in database and if available
  // update the quantity of stock and update
  // the account balance accordingly
  private void addSoldItemDetail() {
    // Take the input from user into edit fields and convert into string
1   String stockName = fieldStockName.getText().toString();
2   String stockQuantity = fieldStockQuantity.getText().toString();

    // Check whether the stock name field is non-empty
    // if empty show a message to enter it.
3   if (TextUtils.isEmpty(stockName)) {
4       Toast.makeText(getApplicationContext(),
                "Please Enter Stock Name!!",
                Toast.LENGTH_LONG)
                .show();
5       return;
    }

    // Check whether the stock quantity field is non-empty
    // if empty show a message to enter it.
6   if (TextUtils.isEmpty(stockQuantity)) {
7       Toast.makeText(getApplicationContext(),
```

```
                "Please Enter Stock Quantity!!",
                Toast.LENGTH_LONG)
                .show();
        return;
    }

    // Find the stock item in the database using the entered name
    // if available update the quantity of stock and update
    // the account balance accordingly
    DocumentReference docRef = db.collection("stocks").document(stockName);
    docRef.get().addOnSuccessListener(documentSnapshot -> {
        Stock stocks = documentSnapshot.toObject(Stock.class);

        if(stocks != null && stocks.getStockName().equals(stockName)){
            // stock name found update its quantity and account balance
            float priceStock = Float.parseFloat(stocks.getStockPrice());
            float quantityStocks =
Float.parseFloat(stocks.getStockQuantity());
            float updateQuantity = Float.parseFloat(stockQuantity);
            String newQuantity = Float.toString(quantityStocks -
updateQuantity);

            stocks.setStockQuantity(newQuantity);
            db.collection("stocks").document(stockName).set(stocks);

            DocumentReference docRef1 =
db.collection("account").document("balance");
            docRef1.get().addOnSuccessListener(documentSnapshot1 -> {
                Balance balance =documentSnapshot1.toObject(Balance.class);

                assert balance != null;
                // balance record found update it
                float curBalance = Float.parseFloat(balance.getAmount());
                curBalance += priceStock*updateQuantity;
                String newBalance = Float.toString(curBalance);

                // update the balance record present in database
                balance.setAmount(newBalance);
                db.collection("account").document("balance").set(balance);

                Toast.makeText(getApplicationContext(),
                        "Sold Record Updated Successfully.",
                        Toast.LENGTH_LONG)
```

```
                        .show();

                // return back to previous page
29              startActivity(new Intent(AddSoldRecord.this,
AccountManagement.class));
30              finish();
            });
        }
        else{
            // stock name not found in database show error message
31          Toast.makeText(getApplicationContext(),
                    "Sorry Stock: "+ stockName +" Not Found. Add stock
record first.",
                    Toast.LENGTH_LONG)
                     .show();
        }
    });
32}
```

Control Flow Graph:-



Cyclomatic complexity:-

E = 34, V = 31, C = E - V + 2 = 5

Possible Paths are:-

1. 1→2→3→4→5→32
2. 1→2→3→6→7→8→32
3. 1→2→3→6→9→10→11→12→13→14→15→16→17→18→19→20→21→22→23→24→25→26→27→28→29→30→32
4. 1→2→3→6→9→10→11→12→13→14→15→16→17→18→19→20→21→22→32
5. 1→2→3→6→9→10→11→12→31→32

Test suite for each path (input, output):-

1. ((, 1), enter stock name message)
2. ((toothpaste,), enter stock quantity message)
3. ((toothpaste, 2), success message)
4. ((toothpaste, 0), fail message)
5. ((table, 2), fail message)

Testing:-

Youtube Link:- https://youtu.be/ciGAQlVSBuk

Test suit 1:-

Input: , 1
Expected Output: enter stock name message
System Output: enter stock name message

Test suit 2:-

Input: toothpaste,
Expected Output: enter stock quantity message
System Output: enter stock quantity message

Test suit 3:-

Input: toothpaste, 2
Expected Output: success message
System Output: success message

Test suit 4:-

Input: toothpaste, 0
Expected Output: fail message

System Output: success message

Test suit 5:-

Input: table, 2
Expected Output: fail message
System Output: fail message

Failed Test Case List:-

Failed Test Suite (Input, Output)	Function Name
1. ((toothpaste, -1), fail message)	addSoldItemDetail

----------xxxxxxxxxxxxxxxxxxx----------