# Mobile Computing

Index

| Exp | Title | Dates |
|---|---|---|
| 1 | Study of 5G Architecture | 16/01/2024 |
| 2 | Orthogonal Codes | 23/01/2024 |
| 3 | CDMA | 30/01/2024 |
| 4 | Frequency Reuse | 06/02/2024 |
| 5 | DSS | 13/02/2024 |
| 6 | GSM Security | 20/02/2024 |
| 7 | Study on Android Studio and Flutter (Widgets , Layouts , Events) | 27/02/2024 |
| 8 | Hello World App Using Android Studio | 05/03/2024 |
| 9 | Use of Widgets with Events (Font Change Color Change) | 12/03/2024 |
| 10 | Display Graphical Primitives | 19/03/2024 |
| 11 | Calculator in Android Studio | 26/03/2024 |
| 12 | Database in Android Studio | 02/04/2024 |
|  | Course : Mobile Computing and Cloud | 16/04/2024 |

| | |
|---|---|
| Semester | T.E. Semester VI – Computer Engineering |
| Subject | Mobile computing |
| Subject Professor In-charge | Prof. Sneha Annappanavar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M310A |

| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 21102A0014 |
| TE Division | A |

**Title: Study of Various generation of the Internet**

---

**Explanation:**

1G, 2G, 3G, 4G and 5G are the five generations of mobile networks where G stands for Generation, and the number denotes the generation number. 5G is the latest generation, whereas 1G networks are now obsolete. The cellular technologies GSM, UMTS, LTE and NR enable 2G, 3G, 4G and 5G, respectively.

| Term | Stands for | Launch Year |
|---|---|---|
| 1G | First Generation | 1979 (Obsolete) |
| 2G | Second Generation | 1991 |
| 3G | Third Generation | 2001 |
| 4G | Fourth Generation | 2009 |
| 5G | Fifth Generation | 2019 |

| Generation | Technology standard | Radio access technology |
|---|---|---|
| 1G – First Generation | AMPS, NMT, TACS, J-TACS, C-Netz | FDMA |
| 2G – Second Generation | GSM, D-AMPS, IS-95 | Combination of TDMA & FDMA, and Narrowband CDMA |
| 3G – Third Generation | UMTS (WCDMA) and CDMA2000 | Wideband CDMA and Narrowband CDMA |
| 4G – Fourth Generation | LTE (Long Term Evolution) | OFDMA and SC-FDMA |
| 5G – Fifth Generation | NR (New Radio) | OFDMA |

1G – First Generation

1G stands for the first generation of mobile networks that were designed to provide basic voice calling services. 1G networks started in the early 1980s and were introduced in different parts of the world through various FDMA-based analogue technologies, including AMPS, NMT, TACS, J-TACS and C-Netz.

2G – Second Generation

2G stands for the second generation of mobile networks that initially offered voice calls, text messages and limited mobile internet. 2G networks started in the early 1990s and were introduced in different parts of the world through various digital technologies, including GSM, D-AMPS and IS-95.

3G – Third Generation

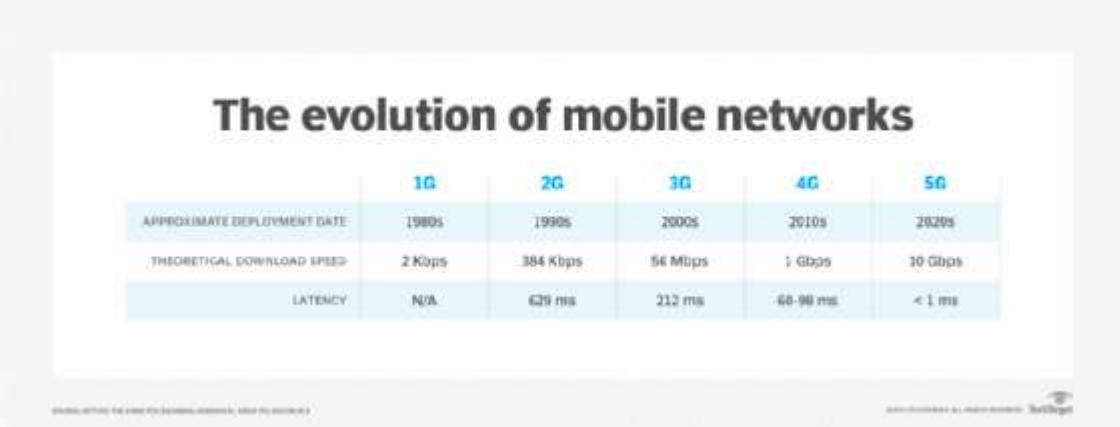
3G stands for the third generation of mobile networks that offer voice, text and data services. The technologies that enable 3G are UMTS and CDMA2000 which are based on the CDMA technology. UMTS is the 3G technology for GSM, and CDMA2000 is the 3G technology for IS-95.

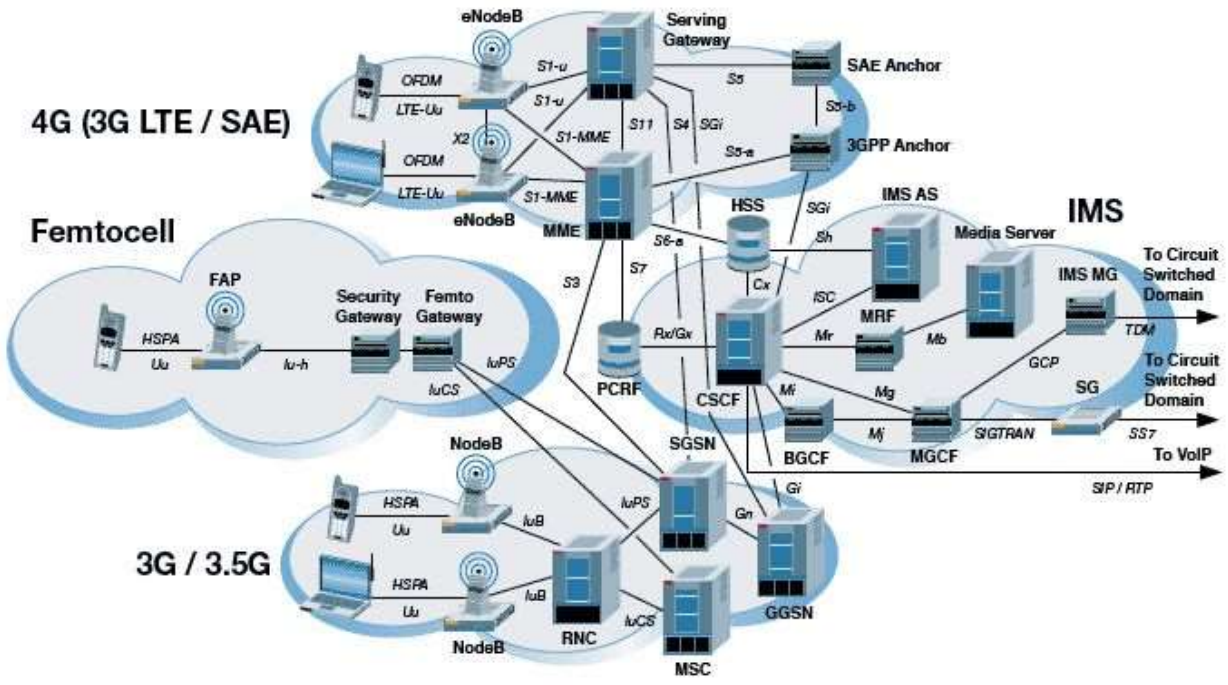
4G – Fourth Generation

4G stands for the fourth generation of mobile networks that are data-only networks enabled by the LTE technology. 4G networks use packet-switching to offer IP-based voice calls and text messages in addition to high-speed mobile data. LTE is the 4G technology for both UMTS and CDMA2000.

5G – Fifth Generation

5G stands for the fifth generation of mobile networks that are data-only and offer average download speeds of around 150 to 200 Mbps. It is the latest generation of mobile networks enabled by the New Radio technology (NR). 5G networks can offer latencies as low as one millisecond.






Sure, I can explain the architecture you sent me, which is a diagram of a typical 3G and 4G cellular network.

The network is made up of a number of different components, which can be broadly categorized into three main parts:

The User Equipment (UE): This is the mobile device that you use to connect to the network, such as your phone or tablet.

The Radio Access Network (RAN): This is the part of the network that is responsible for providing coverage and capacity for mobile devices. It consists of a number of base stations, which are connected to each other via a backhaul network.

The Core Network (CN): This is the part of the network that is responsible for routing calls and data between mobile devices and other networks. It consists of a number of different elements, such as the Mobile Switching Center (MSC), the Gateway GPRS Support Node (GGSN), and the Serving Gateway (S-GW).

The UE communicates with the RAN over the air interface, using a radio frequency (RF) signal. The RAN then forwards the signal to the CN, where it is routed to the appropriate destination.

The specific details of the architecture can vary depending on the type of cellular network (3G or 4G) and the specific vendor of the network equipment. However, the overall principles are the same.

Here are some of the key things to note about the architecture in the image:

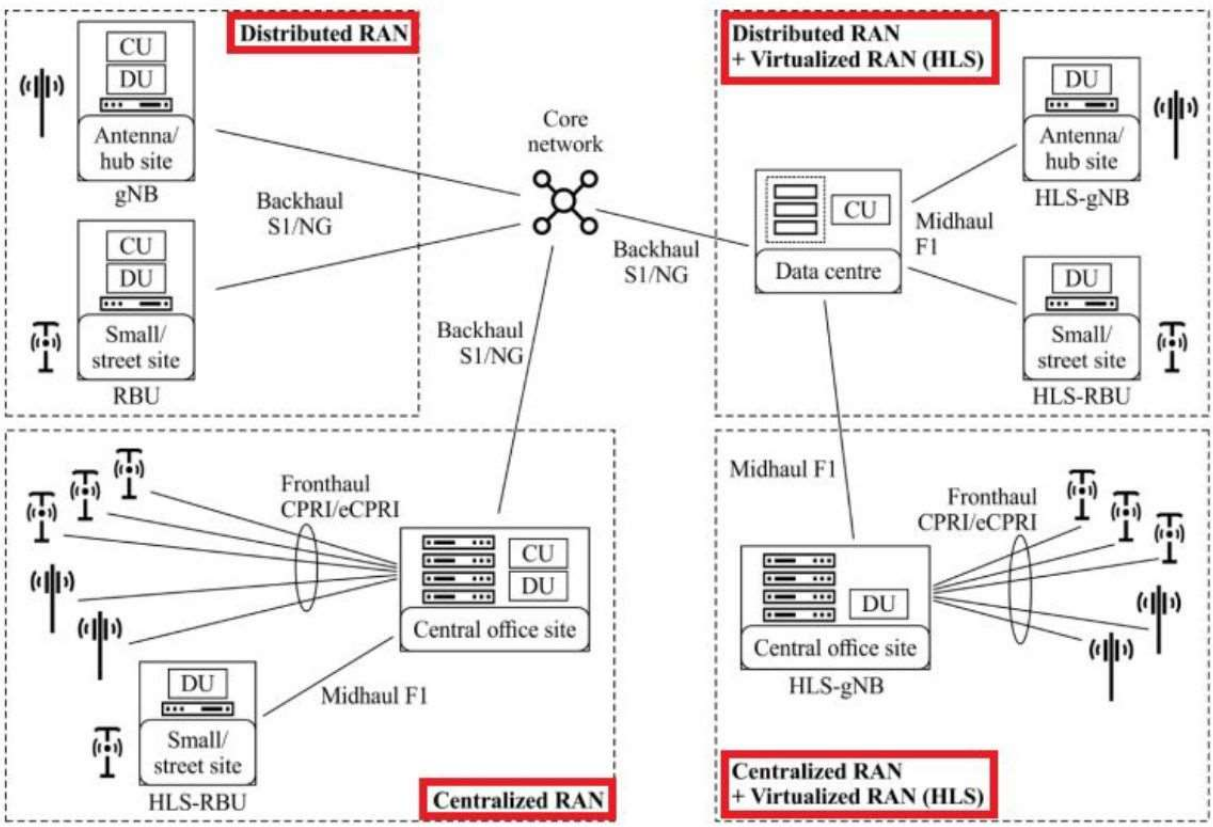
The eNodeB is the base station for 4G networks. It is responsible for providing coverage and capacity for 4G devices.

The NodeB is the base station for 3G networks. It is responsible for providing coverage and capacity for 3G devices.

The MME is the Mobility Management Entity. It is responsible for managing the connections between UEs and the network.

The S-GW is the Serving Gateway. It is responsible for routing data traffic between UEs and the internet.

The HSS is the Home Subscriber Server. It stores information about subscribers, such as their phone numbers and billing information.




The architecture you sent me is a diagram of a distributed and virtualized Radio Access Network (RAN), which is a new approach to designing mobile networks. Traditional RANs are centralized, with all of the processing power and functionality located in a single cell site. This can make them expensive and inflexible to deploy, as well as difficult to scale.

Distributed RANs, on the other hand, are designed to be more flexible and scalable. They distribute the processing power and functionality of the RAN across multiple locations, such as cell sites, central offices, and even data centers. This can make them more cost-effective to deploy, as well as easier to scale to meet the demands of a growing network.

Virtualized RANs take this concept a step further by running the RAN functions on virtual machines (VMs) rather than dedicated hardware. This makes them even more flexible and scalable, as well as easier to manage and maintain.

The specific details of the architecture in the image can vary depending on the vendor and the specific use case. However, the overall principles are the same.

Here are some of the key things to note about the architecture in the image:

The Centralized Unit (CU) is the brains of the RAN. It is responsible for all of the control and signaling functions.

The Distributed Unit (DU) is responsible for the radio processing functions. It is typically located closer to the cell site than the CU.

The HLS-gNB and HLS-RBU are virtualized versions of the gNB and RBU, respectively. They can be run on any standard server hardware.

The fronthaul is the connection between the DU and the CU. It is typically a high-bandwidth, low-latency link.

The midhaul is the connection between the CU and the core network. It can be a variety of different types of links, depending on the specific deployment.

Reference:
https://www.viavisolutions.com/en-uk/what-5g-architecture

https://www.interviewbit.com/blog/lte-architecture/

https://commsbrief.com/what-do-the-terms-1g-2g-3g-4g-and-5g-really-mean/#:~:text=The%20first%2Dgeneration%20mobile%20networks,the%20fifth%20generation%20(5G).

https://www.techtarget.com/searchnetworking/feature/A-deep-dive-into-the-differences-between-4G-and-5G-networks#:~:text=The%20biggest%20difference%20between%204G,such%20as%20faster%20download%20speeds.

---

**Conclusion:**

In conclusion, the evolution of mobile networks from 1G to 5G represents a remarkable journey in the realm of telecommunications. Each generation brought about significant advancements, from the basic voice calling services of 1G to the data-centric, high-speed capabilities of 5G. The transition from analogue to digital technologies, such as GSM, UMTS, LTE, and NR, has played a pivotal role in shaping the mobile communication landscape.

The progression through generations not only addressed the growing demand for enhanced services but also focused on improving security, efficiency, and coverage. The introduction of technologies like TDMA, CDMA, and OFDMA, along with the development of packet-switching and IP-based services, marked crucial milestones in the evolution of mobile networks.

The advent of 5G, with its New Radio (NR) technology and flexibility to operate in various frequency bands, brings forth a new era with unparalleled data rates, low latency, and the ability to cater to diverse use cases. From enhanced mobile broadband to massive Machine Type Communication and ultra-reliable low latency communications, 5G opens doors to innovative applications across industries.

While 4G LTE networks have matured and offer reliable mobile broadband services, 5G presents a substantial leap in performance, with average download speeds of 150 to 200 Mbps and the potential for much higher peak speeds. However, it's essential to acknowledge that 5G is still in its early stages, with most deployments being non-standalone, leveraging a combination of 4G and 5G networks.

As technology continues to advance, it's exciting to anticipate the further enhancements and widespread adoption of 5G, ushering in an era of connectivity that not only transforms our mobile experiences but also enables a plethora of innovative applications and services for both individuals and the Internet of Things (IoT).

# DEPARTMENT OF COMPUTER ENGINEERING

| | |
|---|---|
| Semester | T.E. Semester VI – Computer Engineering |
| Subject | Mobile Computing |
| Subject Professor In-charge | Prof. Sneha Annappanavar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M310A |

| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 21102A0014 |
| TE Division | A |

**Title:**

**Orthogonal Codes**

---

**Explanation:**

In coding theory, orthogonal codes play a crucial role, particularly in communication systems. When two binary codes, A and B, are orthogonal, it means that the inner product of their corresponding sequences is zero.

Mathematically, for binary sequences A=(a1,a2,...,an) and B=(b1,b2,...,bn), the inner product is computed as:

$A \cdot B = \sum_{i=1}^{n} a_i \cdot b_i$

If $A \cdot B = 0$, then the codes A and B are orthogonal.

In our implementation, we utilize a bipolar representation of binary sequences, where '1' is represented as 1 and '0' is represented as -1. This bipolar representation is commonly used in communication systems.

Here's a breakdown of our code implementation:

1. Bipolar Function (Bipolar):
    - Converts a binary string to a bipolar sequence.
    - Each '1' is represented as 1, and each '0' is represented as -1.
2. Multiply Function (Multiply):
    - Performs element-wise multiplication of two bipolar sequences.

3. Main Function:
    - Takes two binary strings as input.
    - Converts them into bipolar sequences.
    - Performs element-wise multiplication of the sequences.
    - Checks if the sum of the multiplied sequence is zero to determine orthogonality.

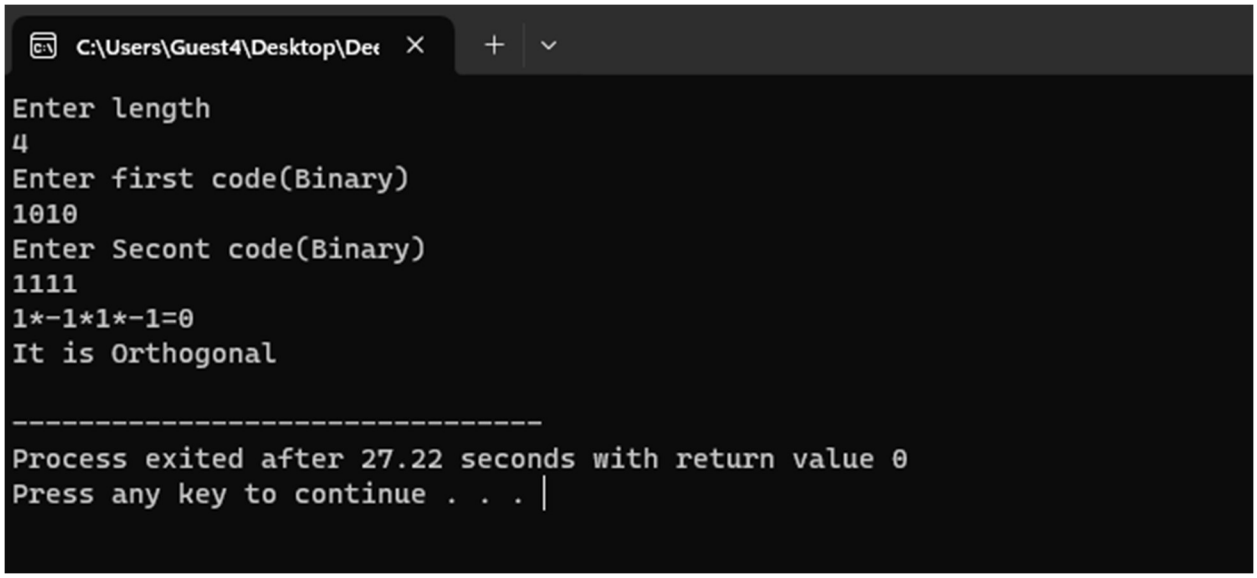
---

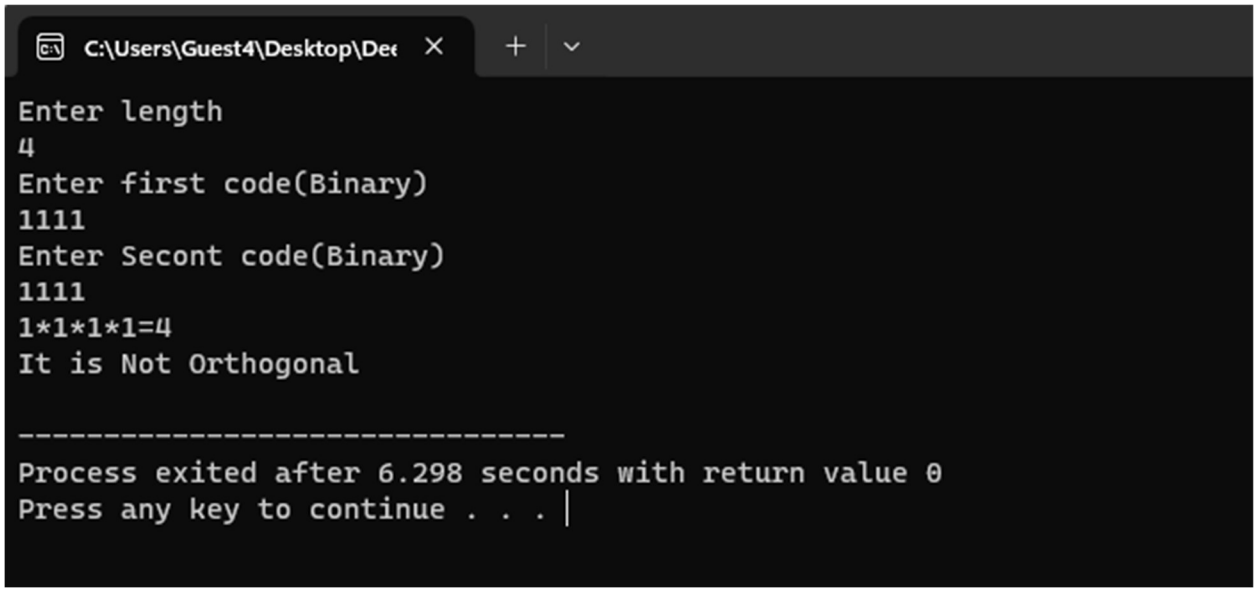
**Implementation:**

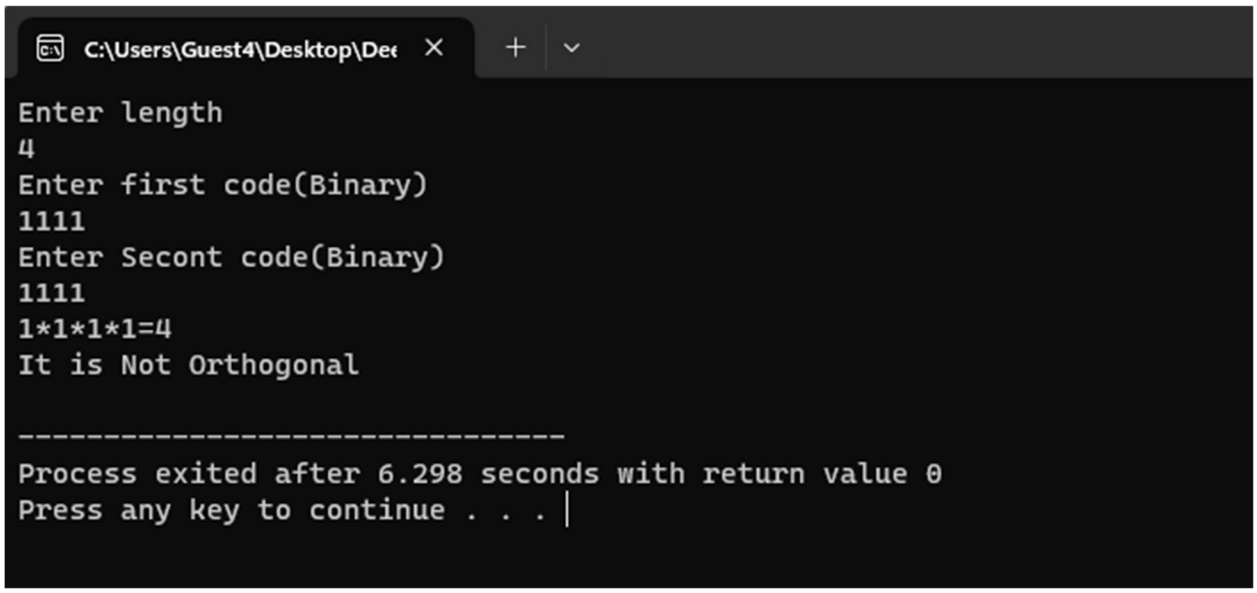
```
#include<iostream>
#include<vector>
using namespace std;
void Bipolar(vector<int>&C,string s){
    int size=s.length();
    for(int i=0;i<size;i++){
        if(s[i]=='1'){
            C.push_back(1);
        }else{
            C.push_back(-1);
        }
    }
}

void    Multiply(vector<int>A,vector<int>B,vector<int>&M){
    int size=A.size();
    for(int i=0;i<size;i++){
        M.push_back(A[i]*B[i]);
    }
}

void TakeCode(string &s,int size){

    int originalsize=size;

    cin>>s;
    if(s.length()!=size){
            cout<<"Invalid size "<<endl;
            return;
    }
    for(int i=0;i<size;i++){
        if(s[i]=='1'|| s[i]=='0'){
```

```
            continue;
     }else{
        cout<<"Not a Binary"<<endl;
        return;
     }

    }

}

int main(){
    int len=0;
    cout<<"Enter length"<<endl;
    cin>>len;
    string A,B;
    cout<<"Enter first code(Binary)"<<endl;
    TakeCode(A,len);
    cout<<"Enter Secont code(Binary)"<<endl;
    TakeCode(B,len);
    vector<int>AC,BC,M;
    Bipolar(AC,A);
    Bipolar(BC,B);
    Multiply(AC,BC,M);
    int temp=0;
    int size=A.size();
    for(int i=0;i<size;i++){
    temp=temp+M[i];
    }

    for(int i=0;i<len;i++){
        cout<<M[i];
        if(i!=len-1) cout<<"*";
    }
    cout<<"="<<temp<<endl;

    if(temp==0){
        cout<<"It is Orthogonal"<<endl;

    }else
    {
            cout<<"It is Not Orthogonal"<<endl;
    }
    return 0;
}
```

**End Result:**


C:\Users\Guest4\Desktop\Dee
Enter length
4
Enter first code(Binary)
1010
Enter Secont code(Binary)
1111
1*-1*1*-1=0
It is Orthogonal
--------------------------------
Process exited after 27.22 seconds with return value 0
Press any key to continue . . .


C:\Users\Guest4\Desktop\Dee
Enter length
4
Enter first code(Binary)
1111
Enter Secont code(Binary)
1111
1*1*1*1=4
It is Not Orthogonal
--------------------------------
Process exited after 6.298 seconds with return value 0
Press any key to continue . . .




---

**Conclusion:**

The code essentially checks whether the two input binary codes are orthogonal by converting them to bipolar sequences and performing element-wise multiplication. If the sum of the multiplied sequence is zero, the codes are orthogonal.

# DEPARTMENT OF COMPUTER ENGINEERING

| | |
|---|---|
| Semester | T.E. Semester VI – Computer Engineering |
| Subject | Mobile Computing |
| Subject Professor In-charge | Prof. Sneha Annappanavar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M310A |

| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 21102A0014 |
| TE Division | A |

**Title:**

**DSSS**

---

**Explanation:**

Introduction: Direct Sequence Spread Spectrum (DSSS) is a modulation technique used in wireless communication systems to spread the signal over a wider bandwidth than the minimum required. It involves multiplying the data signal by a chipping code sequence.

Chipping Code: The chipping code, also known as the spreading code, is a sequence of chips (binary digits) generated by a pseudorandom noise (PN) sequence generator. This code determines how the data signal is spread across the bandwidth.

Encryption Process:

1. Input Data: The original data is taken as input, usually in the form of binary digits.
2. Chipping Code Generation: Another binary sequence, called the chipping code, is generated. This sequence is typically longer than the original data sequence.
3. DSSS Encryption: Each bit of the original data is XORed with the corresponding bit of the chipping code sequence. This process spreads the data across a wider bandwidth.
4. Transmission: The encrypted signal, which now occupies a larger bandwidth due to the spreading effect, is transmitted over the channel.

Decryption Process:

1. Received Signal: The transmitted signal, possibly corrupted by noise and interference, is received.

2. Synchronization: Synchronization is achieved between the received signal and the chipping code sequence.
3. DSSS Decryption: The received signal is XORed with the synchronized chipping code sequence to recover the original data.
4. Output Data: The decrypted data is obtained, which ideally matches the original input data.

---

**Implementation:**

```
#include<iostream>
#include<vector>
using namespace std;

bool Takeinput(vector<int>&input,int &size){

    cout<<"Pls enter the size"<<endl;
    cin>>size;

    string tempinput;

    cout<<"pls Enter value"<<endl;
    cin>>tempinput;

    if(tempinput.length()!=size){
        cout<<"Invalid size"<<endl;
        return false;
    }

    for(int i=0;i<size;i++){

        char temp=tempinput[i];

        if(temp=='1')
        input.push_back(1);
        else
        input.push_back(0);
    }

    return true;
```

```
}

void DSSS_encrypt(vector<int>data,int data_size,vector<int>chipping_code,int
chipping_size,vector<vector<int> >&division){

    int parts=data_size;
    int size_of_part=chipping_size/parts;

    int index=0;
    for(int i=0;i<chipping_size;i++){

        int parts=size_of_part;
        vector<int>temp;
        while(parts--){
            int xored=chipping_code[i]^data[index];
            temp.push_back(xored);
            i++;
        }
        i--;
        index++;
        division.push_back(temp);


    }

    cout<<"Encryted value"<<endl;
    for(int i=0;i<division.size();i++){
        for(int j=0;j<division[0].size();j++){
            cout<<division[i][j]<<" ";
        }
    }
    cout<<endl;


}

vector<int> DSSS_decode(vector<vector<int> >encoded,vector<int>chipping){

    int index=0;
    vector<int>ans;

    for(int i=0;i<encoded.size();i++){
        for(int j=0;j<encoded[0].size();j++){
            int temp=chipping[index]^encoded[i][j];
```

```
            ans.push_back(temp);
            index++;
        }

    }

    return ans;
}

void Print(vector<int>ans){
    cout<<" Decrypted value"<<endl;
    for(int i=0;i<ans.size();i++){
        cout<<ans[i]<<" ";
    }
}


int main(){
    vector<int>data;
    int data_size;
    vector<int>chippingCode;
    int chipping_code_size;
    vector<vector<int> >encoded;
    bool check=true;

    cout<<"For Data"<<endl;
    check=Takeinput(data,data_size);
    if(!check) return 0;

    cout<<"For ChippingCode"<<endl;
    check=Takeinput(chippingCode,chipping_code_size);
    if(!check) return 0;

    DSSS_encrypt(data,data_size,chippingCode,chipping_code_size,encoded);
    vector<int>ans=DSSS_decode(encoded,chippingCode);
    Print(ans);
    return 0;
}
```

---

**End Result:**

```
PS E:\GIt> cd "e:\GIt\SEM-6\MC\" ; if ($?) { g++ DSSS.cpp -o DSSS } ; if ($?) { .\DSSS }
For Data
Pls enter the size
2
pls Enter value
01
For ChippingCode
Pls enter the size
14
pls Enter value
01101010110101
Encryted value
0 1 1 0 1 0 1 1 0 0 1 0 1 0
 Decrypted value
0 0 0 0 0 0 0 1 1 1 1 1 1 1
PS E:\GIt\SEM-6\MC>
```

---

**Conclusion:**

In this report, we implemented a basic simulation of the DSSS encryption and decryption process using C++. The program takes input data and a chipping code, encrypts the data using DSSS, and then decrypts the encrypted signal to recover the original data.

DSSS offers several advantages in wireless communication systems, including increased resistance to interference and jamming, enhanced security through encryption, and the ability to coexist with other communication systems operating in the same frequency band. However, it also comes with some drawbacks, such as increased bandwidth requirements and complexity in implementation.

By understanding the principles of DSSS and implementing them in this lab exercise, we gain insights into the fundamental concepts of spread spectrum communication techniques and their practical applications in mobile computing and wireless networks. Further experimentation and study in this area can lead to a deeper understanding of advanced modulation and coding schemes used in modern wireless communication systems.

# DEPARTMENT OF COMPUTER ENGINEERING

| | |
|---|---|
| Semester | T.E. Semester VI – Computer Engineering |
| Subject | Mobile Computing |
| Subject Professor In-charge | Prof. Sneha Annappanavar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M310A |

| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 21102A0014 |
| TE Division | A |

**Title:**

**CDMA**

---

**Explanation:**

1. **CDMA (Code Division Multiple Access)**: CDMA is a multiple access technique used in wireless communication systems. Unlike other multiple access techniques such as TDMA and FDMA, where users share the same frequency channels by dividing them into time slots or frequency bands, CDMA allows multiple users to transmit simultaneously over the same frequency band using unique spreading codes.

2. **Orthogonal Codes**: In CDMA, orthogonal codes are used to distinguish between different users' signals. Orthogonal codes are sequences of binary digits (1s and -1s) that are mutually orthogonal to each other, meaning their dot product is zero. This property allows the receiver to separate the desired signal from interference caused by other users' signals.

3. **Bipolar Encoding**: Bipolar encoding is a method of representing binary data using bipolar signals, where the binary 1s are represented by positive voltages and the binary 0s are represented by negative voltages. In the context of CDMA, bipolar encoding is used to convert binary data into bipolar signals before transmission.

4. **Signal Multiplication**: In CDMA, the transmitted signal is the result of multiplying the encoded data signal with the spreading code. This process spreads the signal across a wider frequency band, making it more resistant to interference and allowing multiple users to transmit simultaneously.

Implementation:

```
#include<iostream>
#include<vector>
using namespace std;

void Bipolar(vector<int>&C,string s){
    int size=s.length();
    for(int i=0;i<size;i++){
        if(s[i]=='1'){
            C.push_back(1);
        }else{
            C.push_back(-1);
        }
    }
}

void    Multiply(vector<int>A,vector<int>B,vector<int>&M){
    int size=A.size();
    for(int i=0;i<size;i++){
        M.push_back(A[i]*B[i]);
    }
}

void TakeCode(string &s,int size){

    int originalsize=size;

    cin>>s;
    if(s.length()!=size){
            cout<<"Invalid size "<<endl;
            return;
    }
    for(int i=0;i<size;i++){
        if(s[i]=='1'|| s[i]=='0'){
            continue;
     }else{
        cout<<"Not a Binary"<<endl;
        return;
     }

    }

}
```

```
void printvector(vector<int> vec){

    int n=vec.size();
    for(int i=0;i<n;i++){
        cout<<vec[i]<<" ";
    }
    cout<<endl;

}

int  getorthogonals(vector<int>&AC,vector<int>&BC){
    int len=0;
    cout<<"Enter length"<<endl;
    cin>>len;
    string A,B;
    cout<<"Enter first code(Binary)"<<endl;
    TakeCode(A,len);
    cout<<"Enter Secont code(Binary)"<<endl;
    TakeCode(B,len);
    vector<int>M;
    Bipolar(AC,A);
    Bipolar(BC,B);
    Multiply(AC,BC,M);
    int temp=0;
    int size=A.size();
    for(int i=0;i<size;i++){
    temp=temp+M[i];
    }

    for(int i=0;i<len;i++){
        cout<<M[i];
        if(i!=len-1) cout<<"*";
    }
    cout<<"="<<temp<<endl;

    if(temp==0){
        cout<<"It is Orthogonal"<<endl;


    }else
    {
            cout<<"It is Not Orthogonal"<<endl;
    }
```

```
    return 0;
}

void TakeData(vector<int>&A,vector<int>&B){

    string Bd,Ad;

    cout<<"Enter data A"<<endl;

    cin>>Ad;
    int la=Ad.length();
    for(int i=0;i<la;i++){
        if(Ad[i]=='1')
        A.push_back(1);
        else{
            A.push_back(-1);
        }
    }

    cout<<"Enter data B"<<endl;

    cin>>Bd;
    int lb=Bd.length();
    for(int i=0;i<lb;i++){
        if(Bd[i]=='1')
        B.push_back(1);
        else{
            B.push_back(-1);
        }
    }


}

void cal_data_signal(vector<int>&data,vector<int>&code,vector<int>&signal){

    int n=code.size();
    for(int i=0;i<n;i++){
        signal.push_back(data[0]*code[i]);
    }


}
```

```
void cal_carier_signal(vector<int>Asignal,vector<int>Bsignal,vector<int>&Csig-
nal){
    int n=Asignal.size();
    for(int i=0;i<n;i++){

        Csignal.push_back(Asignal[i]+Bsignal[i]);
    }
}

void receiver(vector<int>Csignal,vector<int>AK,vector<int>BK){

    vector<int> Adata,Bdata;
    int nak=AK.size();
    int nbk=BK.size();

    //filling Adata;
    for(int i=0;i<nak;i++){
        int temp=0;
        temp=Csignal[i]*AK[i];
        Adata.push_back(temp);
    }

    //filling Bdata;
    for(int i=0;i<nbk;i++){
        int temp=0;
        temp=Csignal[i]*BK[i];
        Bdata.push_back(temp);
    }

    int sum_of_Adata=0;
    int sum_of_Bdata=0;

    for(int i=0;i<nak;i++){
        sum_of_Adata=sum_of_Adata+Adata[i];
    }

    for(int i=0;i<nbk;i++){
        sum_of_Bdata=sum_of_Bdata+Bdata[i];
    }

    if(sum_of_Adata>0){
        cout<<"Adata=1"<<endl;
    }else{
        cout<<"Adata=0"<<endl;
```

```
    }

    if(sum_of_Bdata>0){
        cout<<"Bdata=1"<<endl;
    }else{
        cout<<"Bdata=0"<<endl;
    }

}

int main(){
    //currently this code works only for the values of data is 1bit and the code
is 4 bit
    vector<int > AK,BK;
    getorthogonals(AK,BK);
    vector<int>Adata,Bdata;
    TakeData(Adata,Bdata);
    vector<int>Asignal,Bsignal,Csignal;
    cal_data_signal(Adata,AK,Asignal);
    cal_data_signal(Bdata,BK,Bsignal);
    printvector(Asignal);
    printvector(Bsignal);
    cal_carier_signal(Asignal,Bsignal,Csignal);
    cout<<"Carrier Signal"<<endl;
    printvector(Csignal);
    cout<<"Receiver end"<<endl;
    receiver(Csignal,AK,BK);
    return 0;
}
```

---

**End Result:**

```
PS E:\GIt\SEM-6\MC> cd "e:\GIt\SEM-6\MC\" ; if ($?) { g++ CDMA.cpp -o CDMA } ; if ($?) { .\CDMA }
Enter length
4
Enter first code(Binary)
0101
Enter Secont code(Binary)
0110
1*1*-1*-1=0
It is Orthogonal
Enter data A
1
Enter data B
0
-1 1 -1 1
1 -1 -1 1
Carrier Signal
0 0 -2 2
Receiver end
Adata=1
Bdata=0
PS E:\GIt\SEM-6\MC>
```

---

**Conclusion:**

In this lab experiment, we explored the principles of CDMA communication. We implemented a simple CDMA transmitter and receiver system using C++ programming. The transmitter encodes binary data using bipolar encoding and spreads the signal using orthogonal codes. The resulting signal is then transmitted over a shared communication channel. At the receiver end, the received signal is despread using the same orthogonal codes used at the transmitter. The despread signal is then decoded to recover the original binary data. Through this experiment, we gained practical insights into the operation of CDMA systems and learned about the importance of orthogonal codes in enabling multiple access in wireless communication networks. Additionally, we observed the advantages of CDMA, such as increased capacity and improved resistance to interference, compared to other multiple access techniques.

# DEPARTMENT OF COMPUTER ENGINEERING

| | |
|---|---|
| Semester | T.E. Semester VI – Computer Engineering |
| Subject | Mobile Computing |
| Subject Professor In-charge | Prof. Sneha Annappanavar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M310A |

| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 21102A0014 |
| TE Division | A |

**Title:**

**Frequency Reuse**

---

**Explanation:**

1. **Importing Libraries**:
    - The code imports the **turtle**, **math**, and **random** libraries, which are used for graphics drawing and generating random colors.
2. **Global Variables**:
    - **size**: Represents the size of each hexagon.
    - **label**: Represents the label number for each hexagon.
    - **cluster**: Represents the cluster of hexagons.
3. **Helper Functions**:
    - **round_position(position)**: Rounds the position coordinates to two decimal places.
    - **draw_hexagon(drawer, cords)**: Draws a hexagon with the specified coordinates.
    - **color_hexagon(drawer, cords, color)**: Draws and fills a hexagon with the specified color.
    - **calculate_adjacent_centers(centers, visited, cords)**: Calculates the coordinates of adjacent hexagons.

- **check_input(cords, visited, input_clr)**: Checks if adjacent hexagons have the same color.

4. **Drawing the Cluster** (**draw_cluster(n)**):
    - Draws a cluster of hexagons based on the given input **n**.
    - Initializes a turtle drawer and sets its speed to maximum.
    - Iteratively draws hexagons and calculates adjacent centers until **n** hexagons are drawn.
    - Asks the user to input colors for each hexagon and checks if adjacent hexagons have the same color.
    - If the colors are invalid, prints a message.
5. **Input Colors** (**input_colors(n)**):
    - Asks the user to input colors for each hexagon.
    - Returns a list of input colors.
6. **Main Function**:
    - Prompts the user to enter values for **i** and **j**.
    - Calculates the number of hexagons (**n**) based on the input values.
    - Draws the cluster of hexagons and waits for user interaction.
7. **Random Color Function** (**random_color()**):
    - Generates a random RGB color.

Overall, this code uses the **turtle** library to draw a cluster of hexagons on the screen. It prompts the user to input colors for each hexagon and ensures that adjacent hexagons have different colors. This code can be used for visualizing and experimenting with hexagonal grids.

---

**Implementation:**

```
import turtle
import math
import random


size = 20
label = 1
cluster = []
```

```
def round_position(position):
    return round(position[0], 2), round(position[1], 2)


def draw_hexagon(drawer, cords):
    global size, label
    x, y = cords

    vertices = [
        (-size, 0),
        (-size / 2, -(3**0.5) / 2 * size),
        (size / 2, -(3**0.5) / 2 * size),
        (size, 0),
        (size / 2, (3**0.5) / 2 * size),
        (-size / 2, (3**0.5) / 2 * size),
    ]

    drawer.penup()
    drawer.goto(x, y - 8)
    drawer.write(label, align="center")

    drawer.goto(x + vertices[0][0], y + vertices[0][1])
    drawer.pendown()

    for vertex in vertices[1:]:
        drawer.goto(x + vertex[0], y + vertex[1])

    drawer.goto(x + vertices[0][0], y + vertices[0][1])


def color_hexagon(drawer, cords, color):
    global size, label
    x, y = cords

    vertices = [
        (-size, 0),
        (-size / 2, -(3**0.5) / 2 * size),
        (size / 2, -(3**0.5) / 2 * size),
        (size, 0),
        (size / 2, (3**0.5) / 2 * size),
        (-size / 2, (3**0.5) / 2 * size),
    ]
```

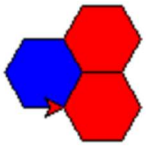
```
    drawer.penup()
    drawer.goto(x, y - 8)
    drawer.write(label, align="center")

    drawer.fillcolor(color)
    drawer.begin_fill()
    drawer.goto(x + vertices[0][0], y + vertices[0][1])
    drawer.pendown()

    for vertex in vertices[1:]:
        drawer.goto(x + vertex[0], y + vertex[1])

    drawer.goto(x + vertices[0][0], y + vertices[0][1])
    drawer.end_fill()


def calculate_adjacent_centers(centers, visited, cords):
    global size
    x, y = cords

    adjacent = [
        (-3 / 2 * size, -(3**0.5) / 2 * size),
        (0, -(3**0.5) * size),
        (3 / 2 * size, -(3**0.5) / 2 * size),
        (3 / 2 * size, (3**0.5) / 2 * size),
        (0, (3**0.5) * size),
        (-3 / 2 * size, (3**0.5) / 2 * size),
    ]

    adjacent_centers = [
        round_position((center[0] + x, center[1] + y)) for center in adjacent
    ]

    for center in adjacent_centers:
        if center not in visited:
            distance = round((center[0] ** 2) + (center[1] ** 2), 4)
            centers.append(center)


def check_input(cords, visited, input_clr):
    x, y = cords

    adjacent = [
        (-3 / 2 * size, -(3**0.5) / 2 * size),
        (0, -(3**0.5) * size),
```

```
        (3 / 2 * size, -(3**0.5) / 2 * size),
        (3 / 2 * size, (3**0.5) / 2 * size),
        (0, (3**0.5) * size),
        (-3 / 2 * size, (3**0.5) / 2 * size),
    ]

    adjacent_centers = [
        round_position((center[0] + x, center[1] + y)) for center in adjacent
    ]

    center_ind = visited.index(cords)

    for center in adjacent_centers:
        ind = visited.index(center)

        if input_clr[center_ind] == input_clr[ind]:
            return False

    return True



def draw_cluster(n):
    global cluster, label

    if n == 0:
        return

    temp = n
    drawer = turtle.Turtle()
    drawer.speed(0)

    centers = []
    centers.append((0, 0))
    visited = []
    cluster = [(label, (0, 0))]

    while temp != 0:
        cords = centers.pop(0)

        if cords not in visited:
            visited.append(cords)
            draw_hexagon(drawer, cords)
            label += 1
```

```
            cluster.append((label, (0, 0)))
            temp -= 1
            calculate_adjacent_centers(centers, visited, cords)

    input_clr = input_colors(n)

    for i in range(n):
        color_hexagon(drawer, visited[i], input_clr[i])

    for i in range(n):
        if not check_input(visited[i], visited, input_clr):
            print("Invalid color selection!")
            break



def input_colors(n):
    input_clr = []

    for i in range(n):
        input_clr.append(input("Enter color for {} cell: ".format(i)))

    return input_clr


def random_color():
    r = random.random()
    g = random.random()
    b = random.random()
    return (r, g, b)


def main():
    i, j = map(int, input("Enter values of i, j: ").split())
    n = i ** 2 + i * j + j ** 2

    screen = turtle.Screen()

    draw_cluster(n)

    screen.exitonclick()


if __name__ == "__main__":
    main()
```

**End Result:**

```
PROBLEMS  OUTPUT  TERMINAL  A

PS E:\GIt> python -u "e:\GIt\SE
Enter values of i, j: 1 1
Enter color for 0 cell: red
Enter color for 1 cell: blue
Enter color for 2 cell: red
Invalid color selection!
```



```
PROBLEMS  OUTPUT  TERMINAL  AZU

PS E:\GIt> python -u "e:\GIt\SEM-
Enter values of i, j: 1 1
Enter color for 0 cell: red
Enter color for 1 cell: blue
Enter color for 2 cell: green
```

**Conclusion:**

Each cellular base station is allocated a group of radio channels to be used within a small geographic

area called a cell. Base stations in adjacent cells are assigned channel groups which contain completely different channels than neighboring cells. Base station antennas are designed to achieve the desired coverage within a particular cell. By limiting the coverage area within the boundaries of a cell, the same group of channels may be used to cover different cells that are separated from one another by geographic distances large enough to keep interference levels within tolerable limits. The design process of selecting and allocating channel groups for all cellular base stations within a system is called frequency reuse or frequency planning

# DEPARTMENT OF COMPUTER ENGINEERING

| | |
|---|---|
| Semester | T.E. Semester VI – Computer Engineering |
| Subject | Mobile Computing |
| Subject Professor In-charge | Prof. Sneha Annappanavar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M310A |

| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 21102A0014 |
| TE Division | A |

**Title:**

**GSM security**

---

**Explanation:**

In GSM (Global System for Mobile Communications) security, A3, A8, and A5 are algorithms used to provide authentication, key generation, and encryption, respectively. Here's some theoretical background on each:

1. **A3 Algorithm (Authentication Algorithm)**:

    - The A3 algorithm is responsible for authenticating the mobile subscriber to the network. It generates a digital signature based on a secret key (Ki) stored on the SIM card and the subscriber's authentication challenge (RAND) received from the network.
    - When a mobile device attempts to connect to the network, the network sends a random challenge (RAND) to the device.
    - The device uses the A3 algorithm along with its secret key (Ki) to generate a response (SRES), which it sends back to the network.
    - The network also calculates the expected response using the same algorithm and compares it with the received response to authenticate the device.

2. **A8 Algorithm (Key Generating Algorithm)**:

    - The A8 algorithm is used to generate session keys (Kc) for encryption purposes. These session keys are unique to each call or data session and are used to encrypt the communication between the mobile device and the base station.

- A8 takes as input the secret key (Ki) and the random challenge (RAND) generated during the authentication process.
- It produces the session key (Kc) as its output, which is then used by both the mobile device and the network to encrypt and decrypt communication.

3. **A5 Algorithm (Encryption Algorithm)**:
   - The A5 algorithm is responsible for encrypting voice and data traffic between the mobile device and the base station. It ensures the confidentiality of the communication by scrambling the data using a stream cipher.
   - A5 generates a pseudo-random keystream based on a session key (Kc) and an initialization vector (IV). This keystream is XORed with the plaintext data to produce encrypted ciphertext.
   - Both the mobile device and the network use the same session key (Kc) and initialization vector (IV) to synchronize the encryption and decryption process.

---

**Implementation:**

```
#include <string>
#include <iostream>
#include <cstdlib> // For rand() and srand()
#include <ctime>   // For time()
using namespace std;

string Random128(){

   // Seed the random number generator
    srand(time(nullptr));

    // This function creates a random 128-bit string comprising of 0 and 1
    std::string s;
    for (int i = 0; i < 128; ++i) {
        s.push_back('0' + rand() % 2); // Convert 0 and 1 to characters '0' and
'1'
    }
    return s;

}

void printstring(string s){
```

```
    for(int i=0;i<s.length();i++){
        cout<<s[i];
    }
    cout<<endl;
}

string A3(string key, string random){
    //Logic for A3
    //Divide key and random in two 64 bits strings
    //say key_64_L, key_64_R, random_64_L, random_64_R
    //XOR key_64_L and random_64_R and store in A3_64_1
    //XOR key_64_R and random_64_L and store in A3_64_2
    //Divide A3_64_1 and A3_64_2 in two 32 bits strings
    //name them as A3_64_1_L, A3_64_1_R, A3_64_2_L, A3_64_2_R
    //XOR A3_64_1_L and A3_64_2_R and store in A3_32_1
    //XOR A3_64_1_R and A3_64_2_L and store in A3_32_2
    //Xor A3_32_1 and A3_32_2 and store in A3_32
    //return A3_32

    string key_64_L=key.substr(0,64);
    string key_64_R=key.substr(64,64);
    string random_64_L=random.substr(0,64);
    string random_64_R=random.substr(64,64);

    string A3_64_1;
    string A3_64_2;
    for(int i=0;i<64;i++){
        A3_64_1.push_back(key_64_L[i]^random_64_R[i]);
        A3_64_2.push_back(key_64_R[i]^random_64_L[i]);
    }

    string A3_64_1_L=A3_64_1.substr(0,32);
    string A3_64_1_R=A3_64_1.substr(32,32);
    string A3_64_2_L=A3_64_2.substr(0,32);
    string A3_64_2_R=A3_64_2.substr(32,32);

    string A3_32_1;
    string A3_32_2;
    for(int i=0;i<32;i++){
        A3_32_1.push_back(A3_64_1_L[i]^A3_64_2_R[i]);
        A3_32_2.push_back(A3_64_1_R[i]^A3_64_2_L[i]);
    }

    string A3_32;
```

```
    for(int i=0;i<32;i++){
        A3_32.push_back(A3_32_1[i]^A3_32_2[i]);
    }



    return A3_32;


}

string changebit(string s){
    int i=rand()%128;
    if(s[i]=='0'){
        s[i]='1';
    }
    else{
        s[i]='0';
    }

    return s;
}

string A8(string key ,string random){
  //logic for A8
  //Divide key and random in two 64 bits strings
  //say key_64_L, key_64_R, random_64_L, random_64_R
  //XOR key_64_L and random_64_R and store in A8_64_1
  //XOR key_64_R and random_64_L and store in A8_64_2
  //Xor A8_64_1 and A8_64_2 and store in A8_64
  //return A8_64

    string key_64_L=key.substr(0,64);
    string key_64_R=key.substr(64,64);

    string random_64_L=random.substr(0,64);
    string random_64_R=random.substr(64,64);

    string A8_64_1;
    string A8_64_2;

    for(int i=0;i<64;i++){
        A8_64_1.push_back(key_64_L[i]^random_64_R[i]);
        A8_64_2.push_back(key_64_R[i]^random_64_L[i]);
```

```
    }

    string A8_64;
    for(int i=0;i<64;i++){
        A8_64.push_back(A8_64_1[i]^A8_64_2[i]);
    }

    return A8_64;
}

int main(){
    string Key_AUC = Random128();
    string Key_SIM=Key_AUC;
    string Random_Number = Random128();

    string Random_Number_2=changebit(Random_Number);


    cout<<"Key_AUC: "<<endl;
    printstring(Key_AUC);
    cout<<"Key_SIM: "<<endl;
     printstring(Key_AUC);
    cout<<"Random_Number: "<<endl;
    printstring(Random_Number);

    string A3_AUC=A3(Key_AUC,Random_Number);
    string A3_SIM=A3(Key_SIM,Random_Number);




    if(A3_AUC==A3_SIM){
        cout<<"Valid SIM Card"<<endl;
    }
    else{
        cout<<"Invalid SIM Card"<<endl;
    }

    string ecrykey_AUC=A8(Key_AUC,Random_Number);
    string ecrykey_SIM=A8(Key_SIM,Random_Number);



    if(ecrykey_AUC==ecrykey_SIM){
```

```
        cout<<"Proper ecryption is possible"<<endl;
    }
    else{
        cout<<"Proper encryption is not possible"<<endl;
    }


    return 0;
}
```

---

**End Result:**

```
PS E:\GIt\SEM-6\MC> cd "e:\GIt\SEM-6\MC\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tem
pCodeRunnerFile }
Key_AUC:
01101101111110001000010111101000111101101000000011001011101110010011000011101110000100010011100110111011111000001101000000
001011
Key_SIM:
01101101111110001000010111101000111101101000000011001011101110010011000011101110000100010011100110111011111000001101000000
001011
Random_Number:
01101101111110001000010111101000111101101000000011001011101110010011000011101110000100010011100110111011111000001101000000
001011
Valid SIM Card
Proper ecryption is possible
PS E:\GIt\SEM-6\MC>
```

```
PS E:\GIt\SEM-6\MC> cd "e:\GIt\SEM-6\MC\" ; if ($?) { g++ GSMAuth.cpp -o GSMAuth } ; if ($?) { .\GSMAuth }
Key_AUC:
11011000101101110100111010000011111010011110001000011110010111101010111101010001001000101011111100110111100110000100100010
110011
Key_SIM:
11011000101101110100111010000011111010011110001000011110010111101010111101010001001000101011111100110111100110000100100010
110011
Random_Number:
11011000101101110100111010000011111010011110001000011110010111101010111101010001001000101011111100110111100110000100100010
110011
Invalid SIM Card
Proper encryption is not possible
```

---

**Conclusion:**

These algorithms collectively form the security framework of GSM networks, providing authentication, key management, and encryption to protect the privacy and integrity of mobile communication. However, it's worth noting that the A5 encryption algorithm has been subject to various security vulnerabilities over the years, leading to the development of more secure encryption algorithms in later generations of mobile networks

| | |
|---|---|
| Semester | T.E. Semester V – Computer Engineering |
| Subject | Software Engineering |
| Subject Professor In-charge | Prof. Sneha Annappanavar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M313B |

| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 21102A0014 |
| TE Division | A |

**Title: Introduction to Flutter**

---

**Explanation:**

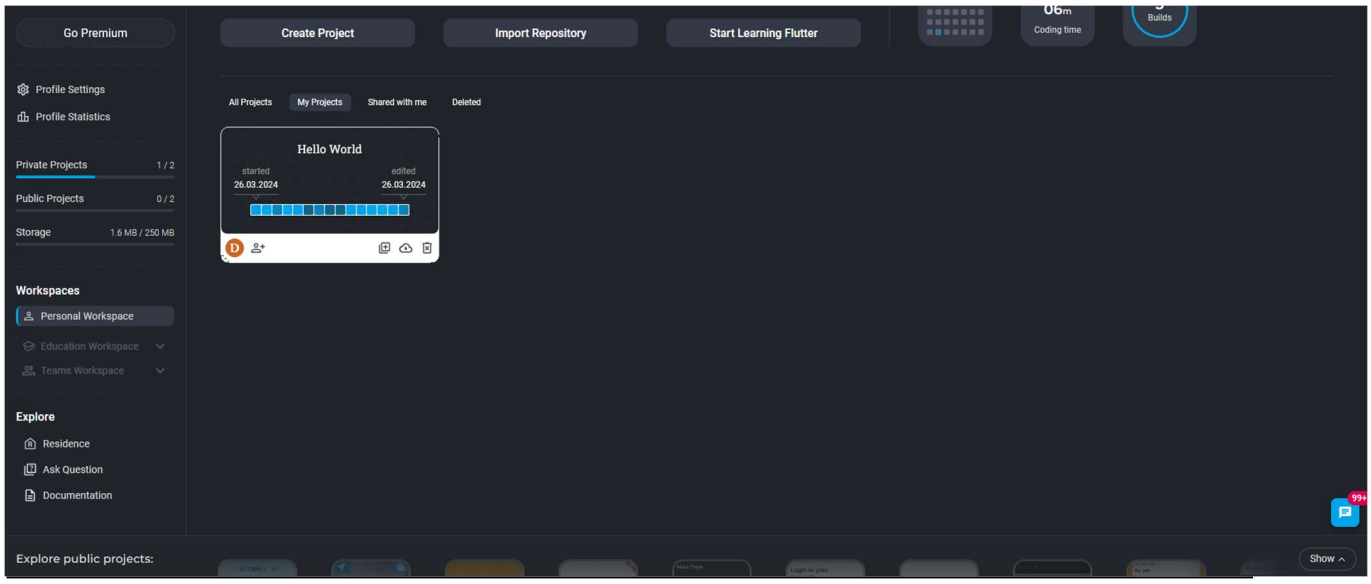
Flutter is an open-source UI framework created by Google that allows you to build beautiful, high-performance apps for mobile (Android and iOS), web, and desktop (Windows, macOS, Linux) using a single codebase.

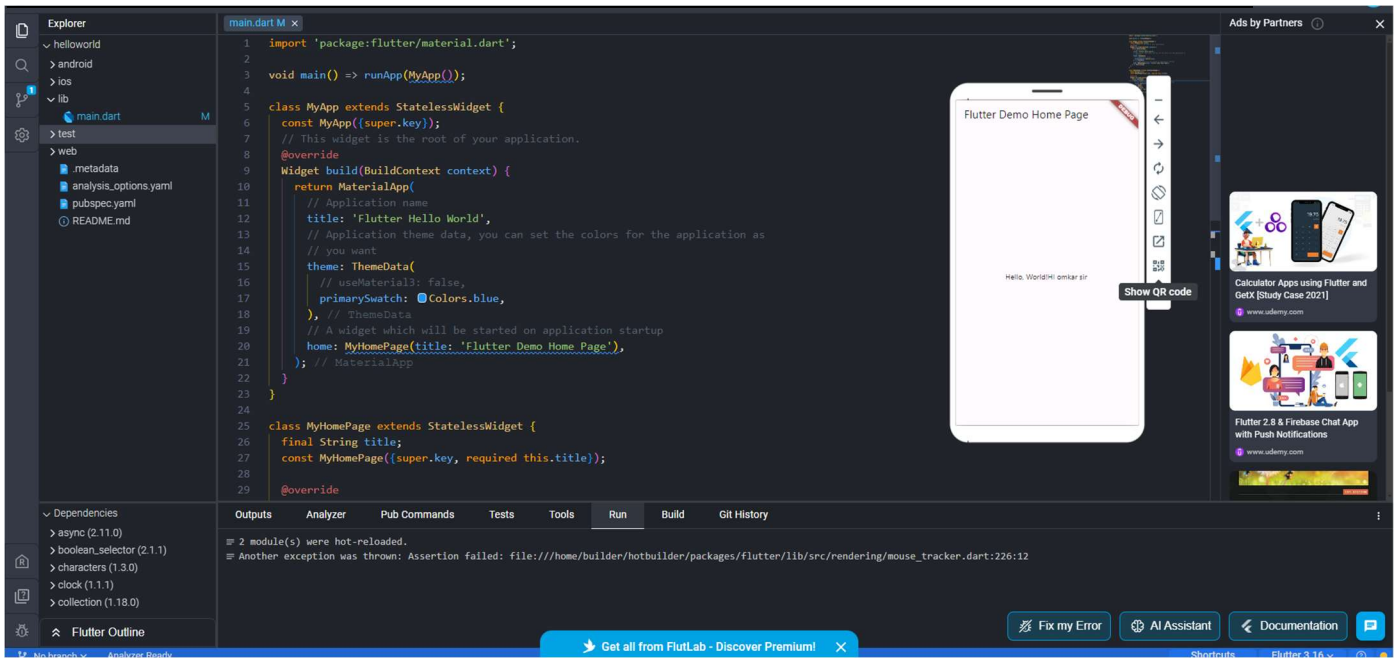
Here's a breakdown of some key points about Flutter:

- **Built with Widgets:** Everything in Flutter is built using widgets, which are reusable building blocks that represent a part of the UI. You can think of them like Legos for your app. By composing these widgets together, you can create complex and interactive UIs.
- **Dart Language:** Unlike many other mobile frameworks that rely on JavaScript, Flutter uses Dart, a modern object-oriented language known for its ease of learning and development speed.
- **Single Codebase:** One of the biggest advantages of Flutter is its ability to write code once and deploy it across multiple platforms. This saves developers a significant amount of time and effort compared to developing separate native apps for each platform.
- **Hot Reload:** Flutter features a super helpful function called hot reload, which allows you to see the changes you make to your code reflected in the app in real-time without having to restart the app. This significantly improves development efficiency.
- **Performance:** Flutter apps are known for their smooth performance and fast rendering. This is because Flutter uses its own rendering engine, which bypasses the need for platform-specific UI components.

---

**Implementation:**




**Conclusion:**

In conclusion, Flutter offers a compelling development experience for creating visually stunning, high-performing applications across various platforms. With its intuitive

**Title:** **Introduction to Flutter** **Roll No:** 21102A0014

widget system, modern Dart language, single codebase advantage, hot reload functionality, and smooth performance, Flutter empowers developers to build robust apps efficiently. If you're looking for a framework to streamline your mobile, web, or desktop app development process, Flutter is definitely worth considering.

| | |
|---|---|
| Semester | T.E. Semester VI – Computer Engineering |
| Subject | Mobile Computing |
| Subject Professor In-charge | Prof. Sneha Annappanavar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M310A |

| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 21102A0014 |
| TE Division | A |

**Title:**

**Introduction to android development**

---

**Explanation:**

**Android development using Java is a popular approach for creating mobile applications for devices running the Android operating system. Java is the primary language used for Android app development, offering a robust and well-supported environment for building powerful, feature-rich applications. Below is a comprehensive introduction to Android development using Java:**

1. **Understanding Android: Android is an open-source mobile operating system developed by Google. It powers billions of devices worldwide, including smartphones, tablets, smartwatches, and even smart TVs. Android provides developers with a rich set of tools and APIs to create innovative and engaging applications.**

2. **Setting Up Development Environment: Before you start developing Android applications, you need to set up your development environment. This includes installing Android Studio, the official integrated development environment (IDE) for Android development. Android Studio provides all the necessary tools for designing, coding, testing, and debugging Android apps.**

3. **Understanding the Android Architecture: Android applications are built using a layered architecture. At the core is the Linux kernel, which provides low-level hardware abstraction and basic system services. On top of the kernel, there are various layers such as the native libraries, the Android Runtime (ART), the application framework, and finally, the applications themselves.**

4. **Learning Java Programming: Since Java is the primary language for Android development, it's essential to have a good understanding of Java programming concepts. This includes object-oriented programming principles, data types, control structures, arrays, collections, and exception handling.**

5. **Understanding Android Components: Android applications are composed of various components that interact with each other to provide the desired functionality. The main components include Activities, Services, Broadcast Receivers, and Content Providers. Understanding how these components work together is crucial for building efficient and scalable Android applications.**

6. Creating User Interfaces with XML and Layouts: Android user interfaces (UIs) are created using XML layout files, which define the structure and appearance of the UI elements. Android provides a wide range of layout managers and UI widgets to design responsive and visually appealing user interfaces.

7. Handling User Input and Events: Android applications interact with users through various input methods such as touch gestures, keyboard input, and voice commands. Handling user input and events is an essential aspect of Android development, and it involves implementing event listeners and callbacks to respond to user actions.

8. Working with Resources: Android applications use various types of resources such as images, strings, colors, and dimensions. These resources are stored in the res directory of the project and can be accessed programmatically using resource identifiers.

9. Understanding Activities and Fragments: Activities are the building blocks of Android applications, representing individual screens or UI components. Fragments are reusable UI components that can be combined within activities to create flexible and modular UI designs.

10. Testing and Debugging: Testing is a crucial part of the Android development process. Android Studio provides tools for testing your applications on virtual devices or physical devices connected to your development machine. You can also use debugging tools to identify and fix errors in your code.

---

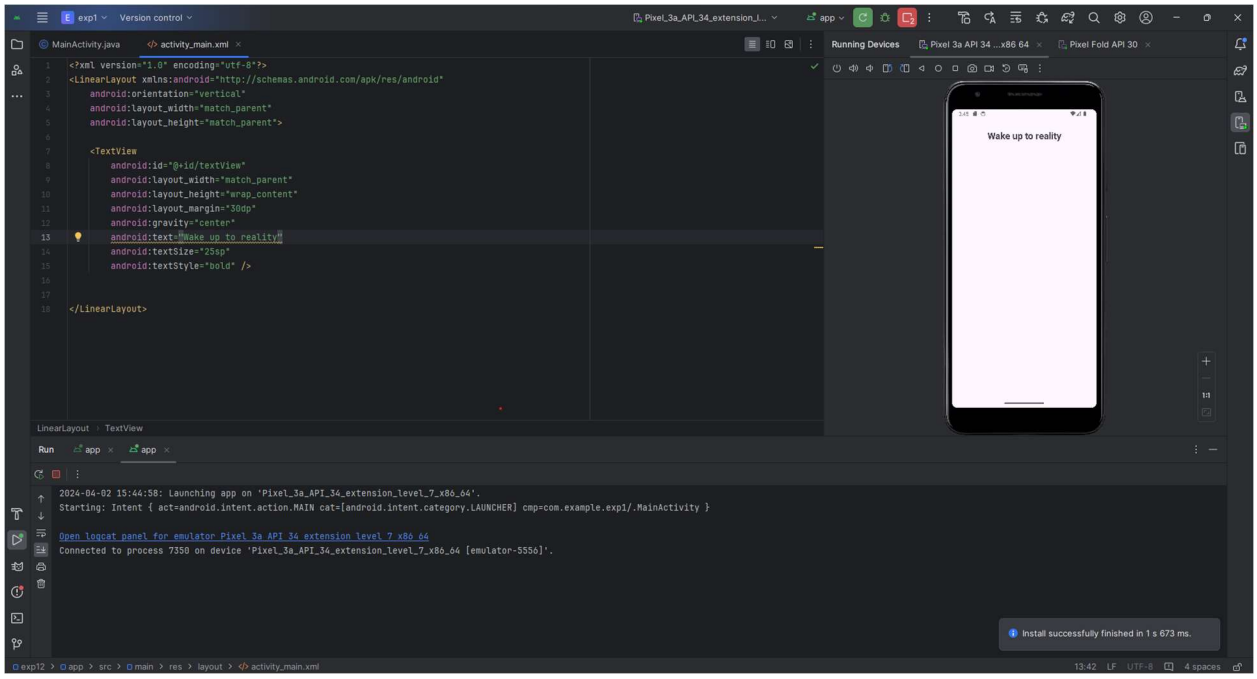
Implementation: -

```
package com.example.exp1;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.graphics.Color;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity
{
    int ch=1;
    float font=30;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final TextView t= (TextView) findViewById(R.id.textView);
        });
                });
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:gravity="center"
        android:text="Wake up to reality”
        android:textSize="25sp"
        android:textStyle="bold" />

    </LinearLayout>
```

---

**End Result:**




---

**Conclusion:**

By mastering these fundamentals of Android development using Java, you'll be well-equipped to create a wide range of Android applications, from simple utility apps to complex, feature-rich applications. As you gain experience and proficiency, you can explore advanced topics such as performance optimization, security, and integration with other technologies and services.

| | |
|---|---|
| Semester | T.E. Semester VI – Computer Engineering |
| Subject | Mobile Computing |
| Subject Professor In-charge | Prof. Sneha Annappanavar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M310A |

| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 21102A0014 |
| TE Division | A |

**Title:**

**Introduction to android development 2**

---

**Explanation:**

**Android development using Java is a popular approach for creating mobile applications for devices running the Android operating system. Java is the primary language used for Android app development, offering a robust and well-supported environment for building powerful, feature-rich applications. Below is a comprehensive introduction to Android development using Java:**

1. **Understanding Android: Android is an open-source mobile operating system developed by Google. It powers billions of devices worldwide, including smartphones, tablets, smartwatches, and even smart TVs. Android provides developers with a rich set of tools and APIs to create innovative and engaging applications.**

2. **Setting Up Development Environment: Before you start developing Android applications, you need to set up your development environment. This includes installing Android Studio, the official integrated development environment (IDE) for Android development. Android Studio provides all the necessary tools for designing, coding, testing, and debugging Android apps.**

3. **Understanding the Android Architecture: Android applications are built using a layered architecture. At the core is the Linux kernel, which provides low-level hardware abstraction and basic system services. On top of the kernel, there are various layers such as the native libraries, the Android Runtime (ART), the application framework, and finally, the applications themselves.**

4. **Learning Java Programming: Since Java is the primary language for Android development, it's essential to have a good understanding of Java programming concepts. This includes object-oriented programming principles, data types, control structures, arrays, collections, and exception handling.**

5. **Understanding Android Components: Android applications are composed of various components that interact with each other to provide the desired functionality. The main components include Activities, Services, Broadcast Receivers, and Content Providers. Understanding how these components work together is crucial for building efficient and scalable Android applications.**

6. Creating User Interfaces with XML and Layouts: Android user interfaces (UIs) are created using XML layout files, which define the structure and appearance of the UI elements. Android provides a wide range of layout managers and UI widgets to design responsive and visually appealing user interfaces.

7. Handling User Input and Events: Android applications interact with users through various input methods such as touch gestures, keyboard input, and voice commands. Handling user input and events is an essential aspect of Android development, and it involves implementing event listeners and callbacks to respond to user actions.

8. Working with Resources: Android applications use various types of resources such as images, strings, colors, and dimensions. These resources are stored in the res directory of the project and can be accessed programmatically using resource identifiers.

9. Understanding Activities and Fragments: Activities are the building blocks of Android applications, representing individual screens or UI components. Fragments are reusable UI components that can be combined within activities to create flexible and modular UI designs.

10. Testing and Debugging: Testing is a crucial part of the Android development process. Android Studio provides tools for testing your applications on virtual devices or physical devices connected to your development machine. You can also use debugging tools to identify and fix errors in your code.

---

Implementation: -

```
package com.example.exp1;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.graphics.Color;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity
{
    int ch=1;
    float font=30;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final TextView t= (TextView) findViewById(R.id.textView);
        Button b1= (Button) findViewById(R.id.button1);
        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                t.setTextSize(font);
```

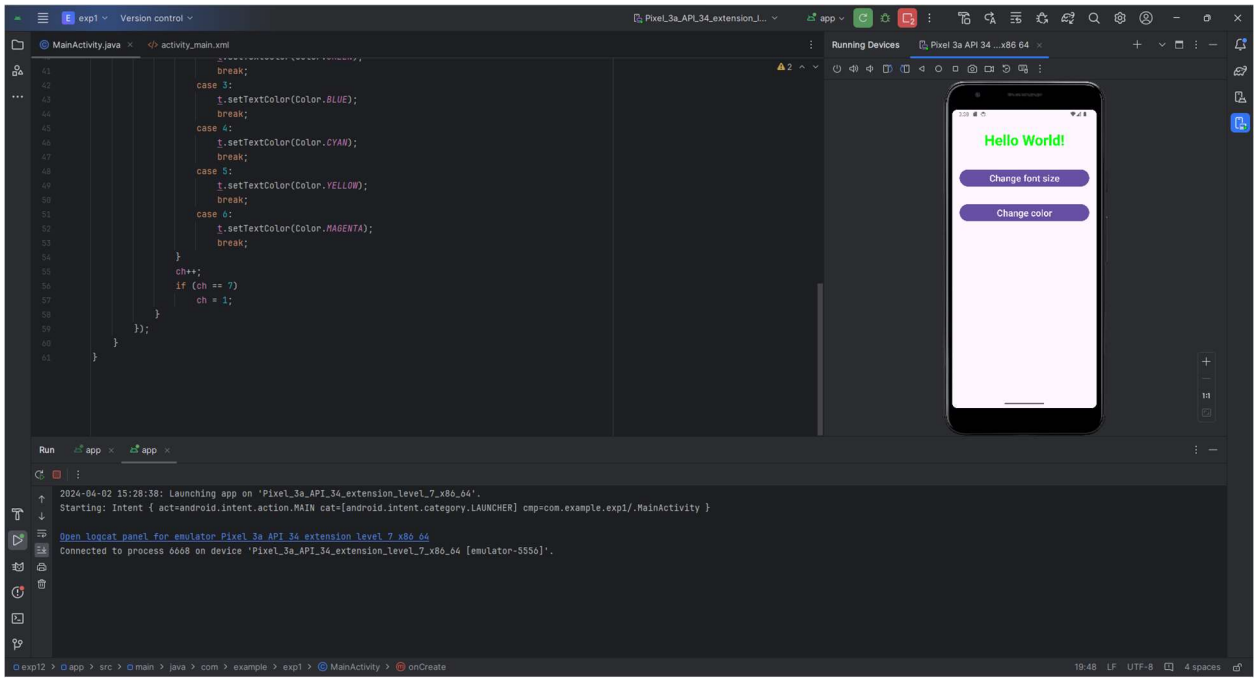
```
                font = font + 5;
                if (font == 50)
                    font = 30;
            }
        });
        Button b2= (Button) findViewById(R.id.button2);
        b2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                switch (ch) {
                    case 1:
                        t.setTextColor(Color.RED);
                        break;
                    case 2:
                        t.setTextColor(Color.GREEN);
                        break;
                    case 3:
                        t.setTextColor(Color.BLUE);
                        break;
                    case 4:
                        t.setTextColor(Color.CYAN);
                        break;
                    case 5:
                        t.setTextColor(Color.YELLOW);
                        break;
                    case 6:
                        t.setTextColor(Color.MAGENTA);
                        break;
                }
                ch++;
                if (ch == 7)
                    ch = 1;
            }
        });
    }
}
```

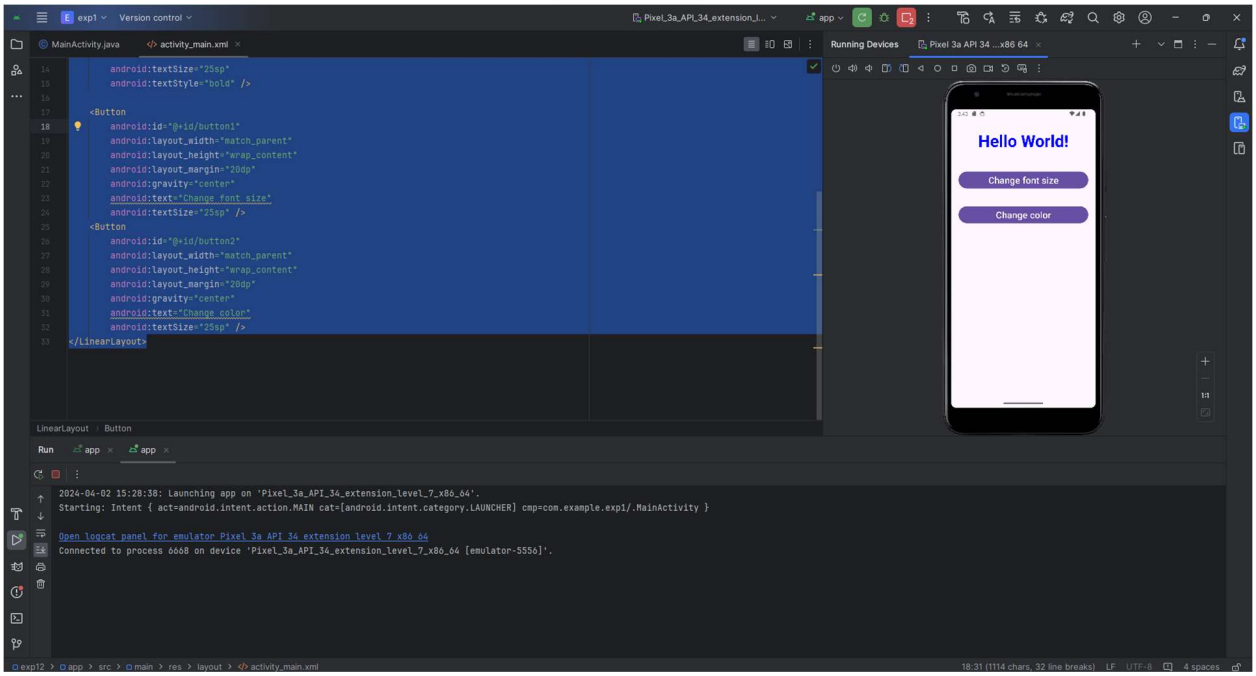
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:gravity="center"
        android:text="Hello World!"
        android:textSize="25sp"
        android:textStyle="bold" />
```

```
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:gravity="center"
        android:text="Change font size"
        android:textSize="25sp" />
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:gravity="center"
        android:text="Change color"
        android:textSize="25sp" />
</LinearLayout>
```

---

**End Result:**






**Conclusion:**

By mastering these fundamentals of Android development using Java, you'll be well-equipped to create a wide range of Android applications, from simple utility apps to complex, feature-rich applications. As you gain experience and proficiency, you can explore advanced topics such as performance optimization, security, and integration with other technologies and services.

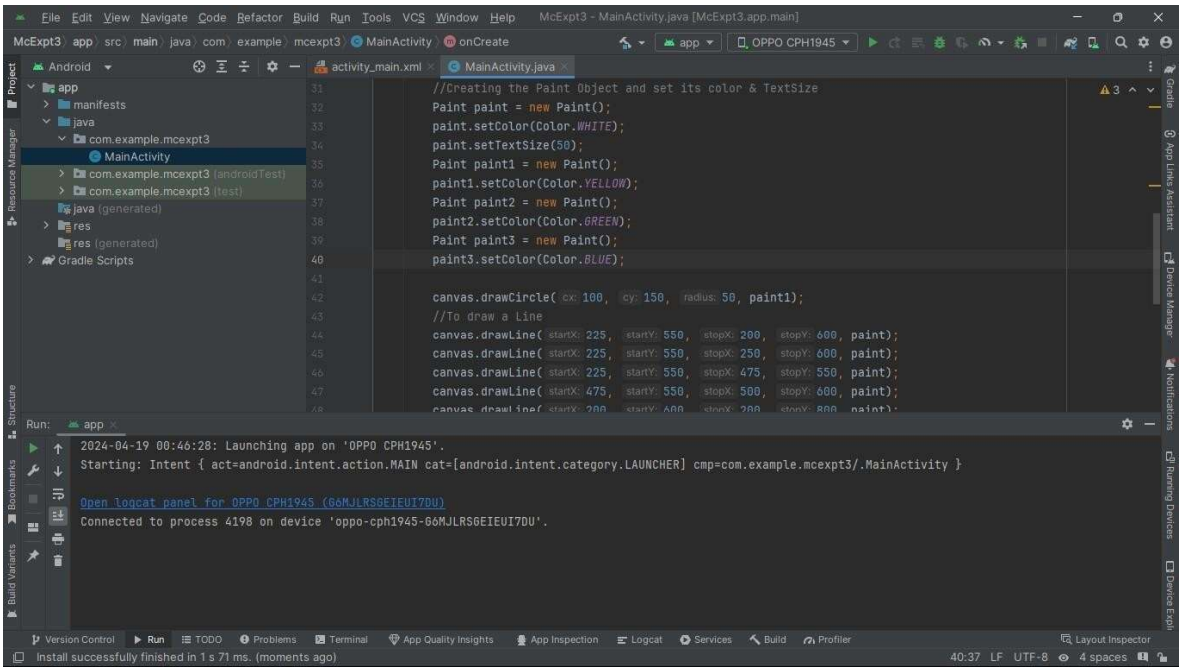
| | |
|---|---|
| Semester | T.E. Semester V – Computer Engineering |
| Subject | Software Engineering |
| Subject Professor In-charge | Prof. Sneha Annappanavar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M313B |

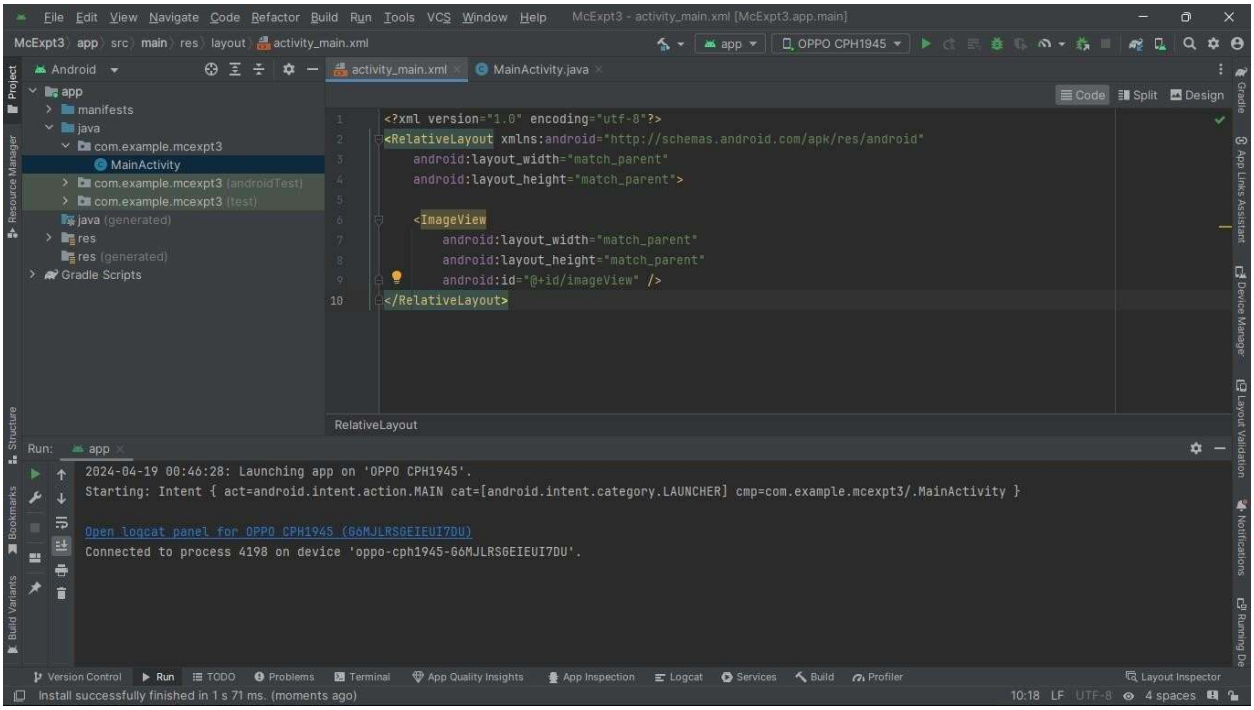
| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 21102A0014 |
| TE Division | A |

**Title: Display Graphical Primitives**

---

**Implementation:**

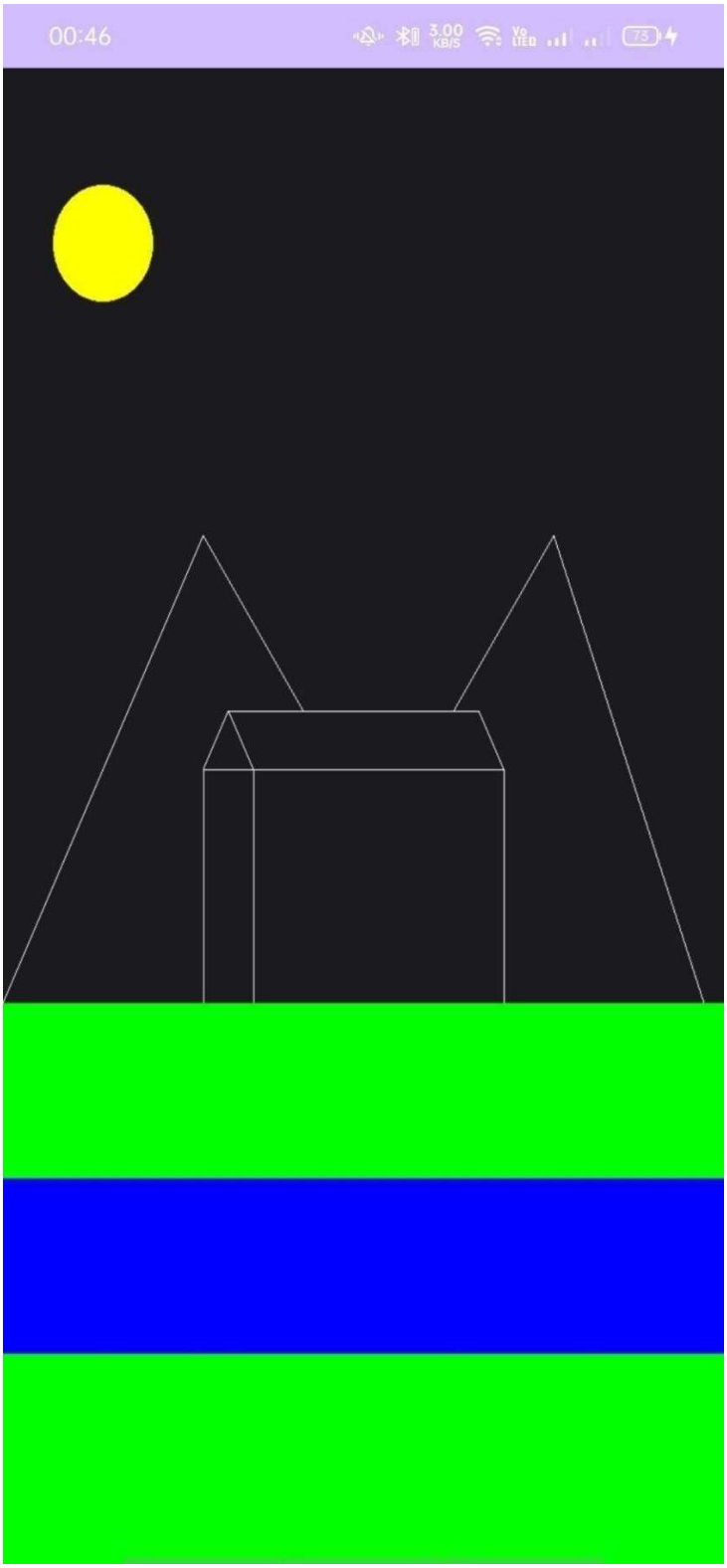





---

**Output:**

**Title:** **Display Graphical Primitives** **Roll No:** 21102A0014




**Title:** **Display Graphical Primitives** **Roll No:** 21102A0014

| | |
|---|---|
| Semester | T.E. Semester VI – Computer Engineering |
| Subject | Mobile Computing |
| Subject Professor In-charge | Prof. Sneha Annappanavar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M310A |

| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 22102A0014 |
| TE Division | A |

**Title: Case Study(Calaculator)**

---

**Explanation:**

1. Design the database schema for storing student information.
2. Set up an SQLite database to manage student data locally.
3. Implement CRUD operations (Create, Read, Update, Delete) for interacting with the database.
4. Design the user interface to input, view, and modify student records.
5. Integrate database operations with UI components for user interaction.
6. Test the app thoroughly to ensure functionality and data integrity.
7. Enhance user experience with features like input validation and data management options.
8. Deploy the app via the Google Play Store or other distribution platforms for users to access and use.

---

**Implementation:**

**activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_x="0dp"
    android:layout_y="0dp"
    android:background="#2A2A2A">
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="64dp"
    android:layout_y="124dp"
    android:fontFamily="@font/amaranth"
    android:text="Bank Acc Details"
    android:textColor="#FFA200"
    android:textSize="40sp"
    android:textStyle="bold" />

<EditText
    android:id="@+id/txtPin"
    android:layout_width="139dp"
    android:layout_height="wrap_content"
    android:layout_x="210dp"
    android:layout_y="236dp"
    android:backgroundTint="#0D23C8"
    android:fontFamily="@font/bree_serif"
    android:gravity="center"
    android:hint=""
    android:inputType="number"
    android:textSize="20sp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="52dp"
    android:layout_y="256dp"
    android:fontFamily="@font/bree_serif"
    android:text="Enter PIN:"
    android:textColor="#B53625"
    android:textSize="20sp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="52dp"
    android:layout_y="306dp"
    android:fontFamily="@font/bree_serif"
    android:text="Enter Acc Type:"
    android:textColor="#B53625"
    android:textSize="20sp" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="52dp"
    android:layout_y="356dp"
    android:fontFamily="@font/bree_serif"
    android:text="Enter Amount:"
    android:textColor="#B53625"
    android:textSize="20sp" />

<EditText
    android:id="@+id/txtActype"
    android:layout_width="139dp"
    android:layout_height="wrap_content"
    android:layout_x="210dp"
    android:layout_y="286dp"
    android:backgroundTint="#0D23C8"
    android:fontFamily="@font/bree_serif"
    android:gravity="center"
    android:hint=""
    android:inputType="text"
    android:textSize="20sp" />

<EditText
    android:id="@+id/txtAmount"
    android:layout_width="139dp"
    android:layout_height="wrap_content"
    android:layout_x="210dp"
    android:layout_y="336dp"
    android:backgroundTint="#0D23C8"
    android:fontFamily="@font/bree_serif"
    android:gravity="center"
    android:hint=""
    android:inputType="number"
    android:textSize="20sp" />


<Button
    android:id="@+id/btnDeb"
    android:layout_width="150dp"
    android:layout_height="62dp"
    android:layout_gravity="center"
    android:layout_x="199dp"
```

```
        android:layout_y="429dp"
        android:backgroundTint="#283593"
        android:fontFamily="@font/fascinate"
        android:text="DEBIT"
        android:textColor="#A0F600"
        android:textSize="22sp" />

    <Button
        android:id="@+id/btnCred"
        android:layout_width="140dp"
        android:layout_height="61dp"
        android:layout_x="50dp"
        android:layout_y="429dp"
        android:backgroundTint="#283593"
        android:fontFamily="@font/fascinate"
        android:text="CREDIT"
        android:textColor="#A0F600"
        android:textSize="22sp" />

    <Button
        android:id="@+id/btnCbal"
        android:layout_width="143dp"
        android:layout_height="80dp"
        android:layout_x="49dp"
        android:layout_y="494dp"
        android:backgroundTint="#283593"
        android:fontFamily="@font/fascinate"
        android:text="CHECK BALANCE"
        android:textColor="#A0F600"
        android:textSize="22sp" />

    <Button
        android:id="@+id/btnAdd"
        android:layout_width="150dp"
        android:layout_height="80dp"
        android:layout_x="202dp"
        android:layout_y="493dp"
        android:backgroundTint="#283593"
        android:fontFamily="@font/fascinate"
        android:text="ADD ACCOUNT"
        android:textColor="#A0F600"
        android:textSize="22sp" />

    <Button
```

```
        android:id="@+id/btnViewAll"
        android:layout_width="296dp"
        android:layout_height="wrap_content"
        android:layout_x="52dp"
        android:layout_y="582dp"
        android:backgroundTint="#283593"
        android:fontFamily="@font/fascinate"
        android:text="View All"
        android:textColor="#A0F600"
        android:textSize="22sp" />


</AbsoluteLayout>
```

**MainActivity.java:**

```
package com.example.exp11b;

import android.app.Activity;
import android.app.AlertDialog.Builder;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends Activity implements OnClickListener
{
    EditText pin, actype, amount;
    Button cred, deb, cbal, viewall, add;
    SQLiteDatabase db;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

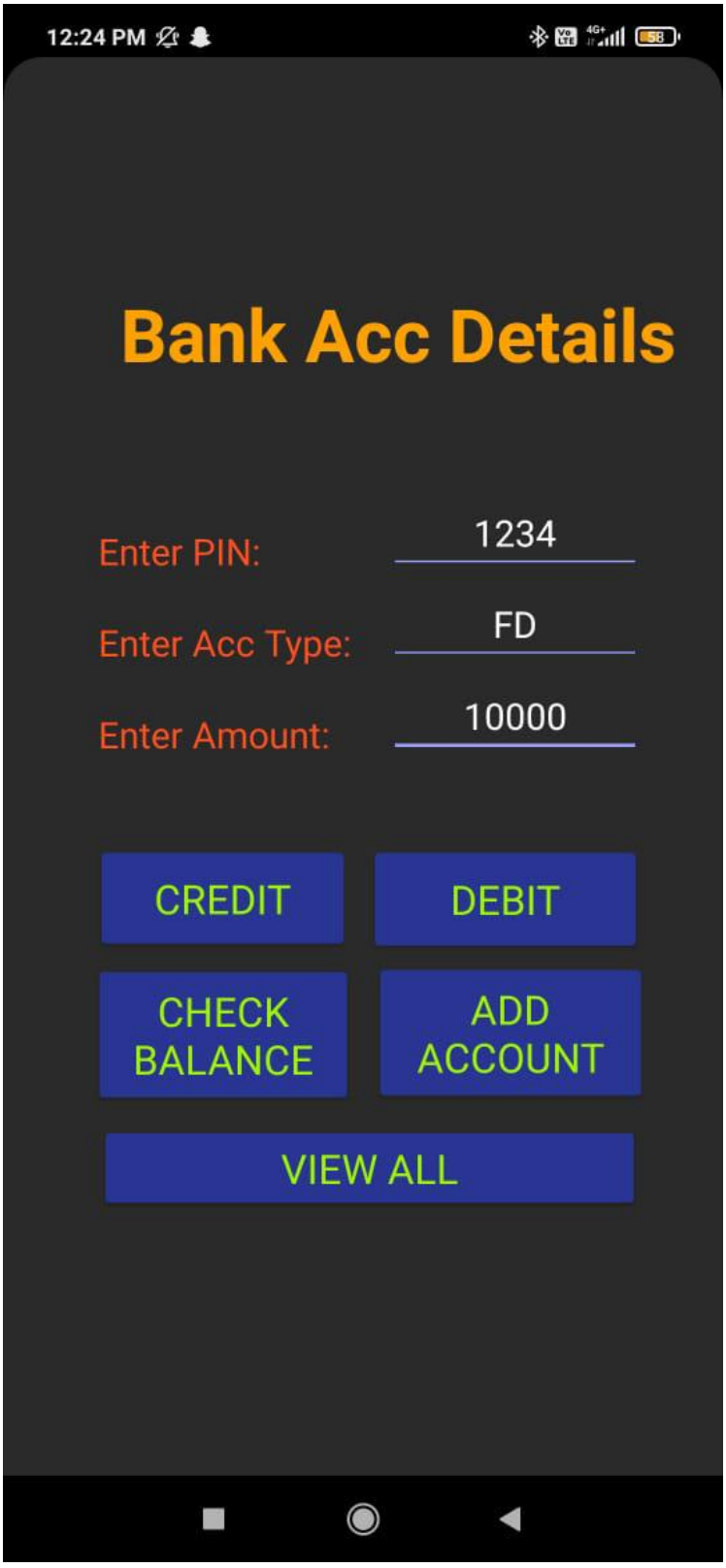
```
        pin = (EditText)findViewById(R.id.txtPin);
        actype = (EditText)findViewById(R.id.txtActype);
        amount = (EditText)findViewById(R.id.txtAmount);

        add = (Button)findViewById(R.id.btnAdd);
        cred = (Button)findViewById(R.id.btnCred);
        deb = (Button)findViewById(R.id.btnDeb);
        cbal = (Button)findViewById(R.id.btnCbal);
        viewall = (Button)findViewById(R.id.btnViewAll);

        add.setOnClickListener(this);
        cred.setOnClickListener(this);
        deb.setOnClickListener(this);
        cbal.setOnClickListener(this);
        viewall.setOnClickListener(this);

        // Creating database and table
        db=openOrCreateDatabase("BankDB", Context.MODE_PRIVATE, null);
        db.execSQL("CREATE TABLE IF NOT EXISTS bank(pin VARCHAR,actype VARCHAR,
balance VARCHAR);");
    }
    public void onClick(View view)
    {
        // Inserting a record
        if(view==add)
        {
            // Checking for empty fields
            if(pin.getText().toString().trim().length()==0||
                    actype.getText().toString().trim().length()==0||
                    amount.getText().toString().trim().length()==0)
            {
                showMessage("Error", "Please enter all values");
                return;
            }
            db.execSQL("INSERT INTO bank VALUES('" + pin.getText() + "','" +
actype.getText() +
                    "','" + amount.getText() + "');");
            showMessage("Success", "Account Created");
            clearText();
        }

        if(view==cred)
        {
```

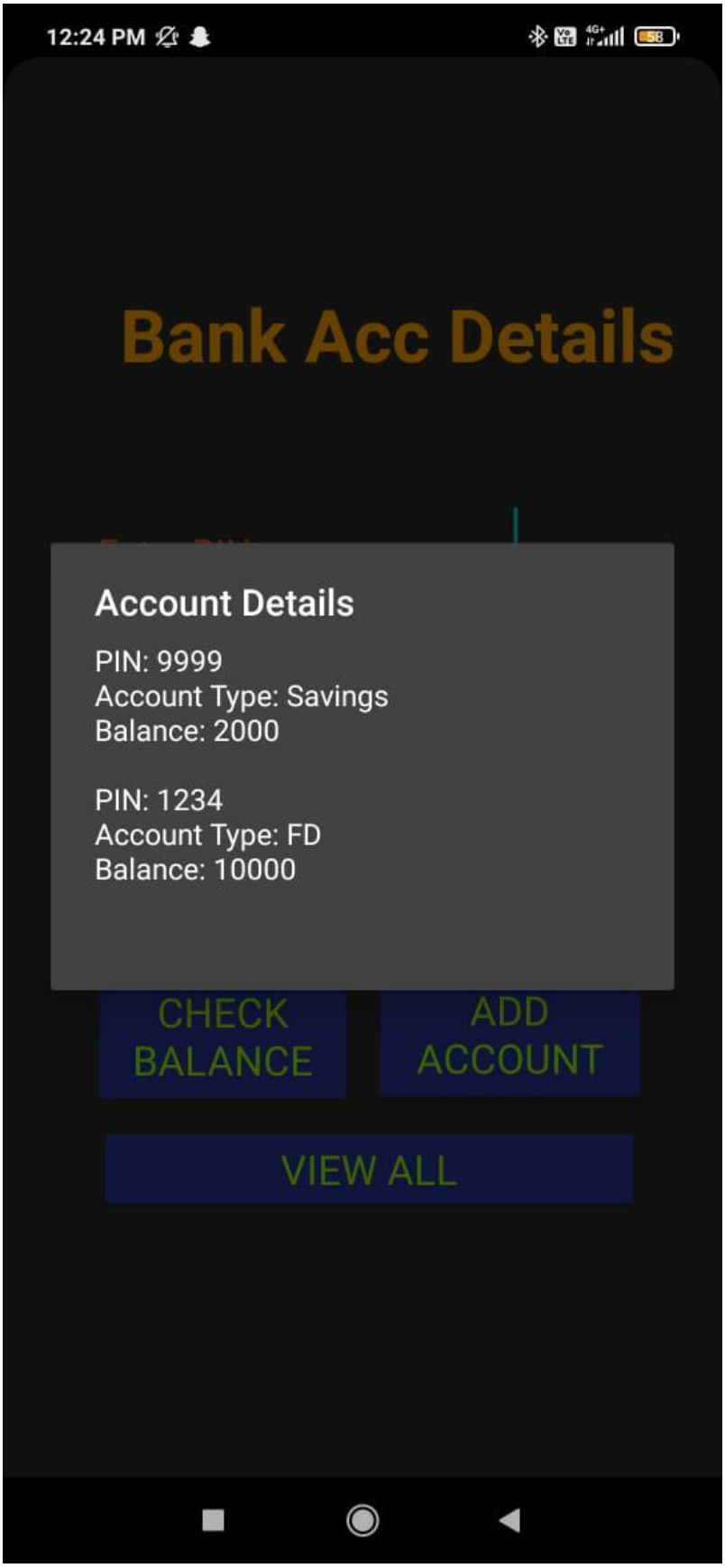
```
            // Checking for empty fields
            if(pin.getText().toString().trim().length()==0||
                    actype.getText().toString().trim().length()==0||
                    amount.getText().toString().trim().length()==0)
            {
                showMessage("Error", "Please enter all values!");
                return;
            }
            Cursor c=db.rawQuery("SELECT * FROM bank WHERE
pin='"+pin.getText()+"'", null);
            if(c.moveToFirst())
            {
                db.execSQL("UPDATE bank SET balance = balance +
'"+amount.getText()+"' WHERE pin='"+pin.getText()+"'");
                showMessage("Success", "Amount Credited");
            }
            else
            {
                showMessage("Error", "Invalid PIN! If not registered the try
creating a new account!!");
            }

            clearText();
        }
        // Deleting a record from the Student table
        if(view==deb)
        {
            // Checking for empty roll number
            if(pin.getText().toString().trim().length()==0)
            {
                showMessage("Error", "Please enter PIN");
                return;
            }
            Cursor c=db.rawQuery("SELECT * FROM bank WHERE
pin='"+pin.getText()+"'", null);
            if(c.moveToFirst())
            {
                db.execSQL("UPDATE bank SET balance = balance -
'"+amount.getText()+"' WHERE pin='"+pin.getText()+"'");
                showMessage("Success", "Amount Debited");
            }
            else
            {
                showMessage("Error", "Invalid PIN!");
```

```
            }
            clearText();
        }

        // Display a record from the Student table
        if(view==cbal)
        {
            // Checking for empty roll number
            if(pin.getText().toString().trim().length()==0)
            {
                showMessage("Error", "Please enter PIN");
                return;
            }
            Cursor c=db.rawQuery("SELECT * FROM bank WHERE
pin='"+pin.getText()+"'", null);

            StringBuffer buffer=new StringBuffer();
            if(c.moveToFirst())
            {
                buffer.append("PIN: "+c.getString(0)+"\n");
                buffer.append("Account Type: "+c.getString(1)+"\n");
                buffer.append("Balance: "+c.getString(2)+"\n\n");
                showMessage("Account Details", buffer.toString());
            }
            else
            {
                showMessage("Error", "Invalid Rollno");
                clearText();
            }
        }
        // Displaying all the records
        if(view==viewall)
        {
            Cursor c=db.rawQuery("SELECT * FROM bank", null);
            if(c.getCount()==0)
            {
                showMessage("Error", "No accounts found!");
                return;
            }
            StringBuffer buffer=new StringBuffer();
            while(c.moveToNext())
            {
                buffer.append("PIN: "+c.getString(0)+"\n");
                buffer.append("Account Type: "+c.getString(1)+"\n");
```

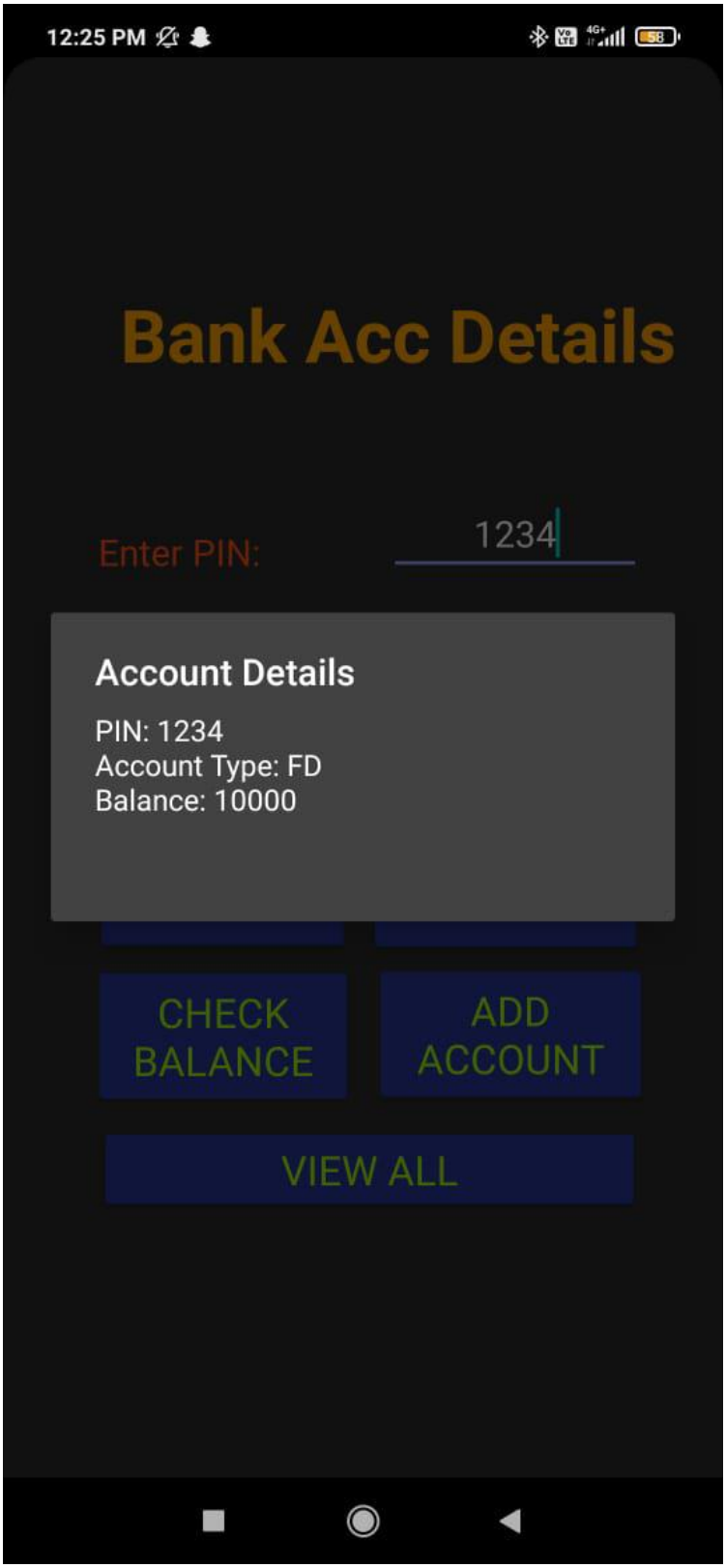
```
                buffer.append("Balance: "+c.getString(2)+"\n\n");
            }
            showMessage("Account Details", buffer.toString());
        }
    }
    public void showMessage(String title,String message)
    {
        Builder builder=new Builder(this);
        builder.setCancelable(true);
        builder.setTitle(title);
        builder.setMessage(message);
        builder.show();
    }
    public void clearText()
    {
        pin.setText("");
        actype.setText("");
        amount.setText("");
        pin.requestFocus();
    }
}
```

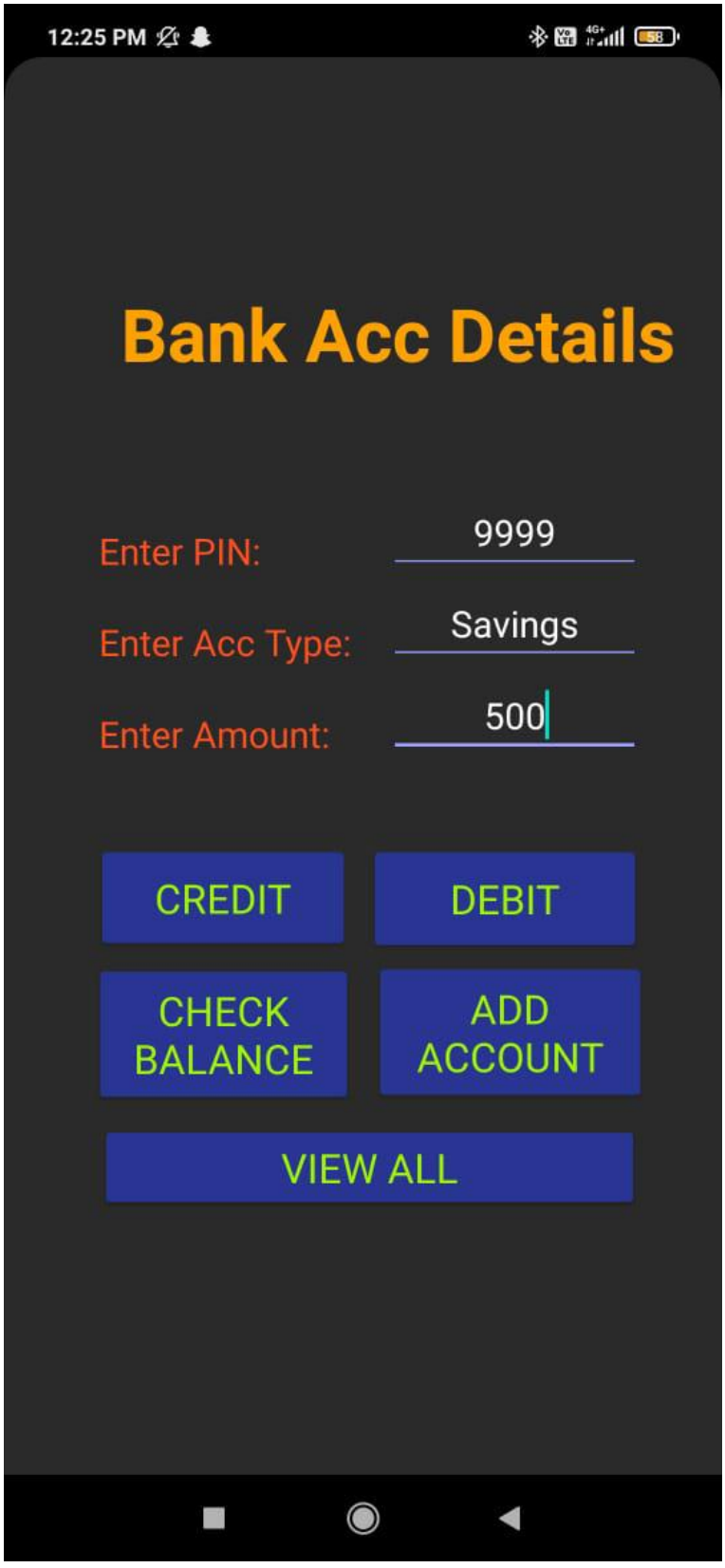
**Output:**

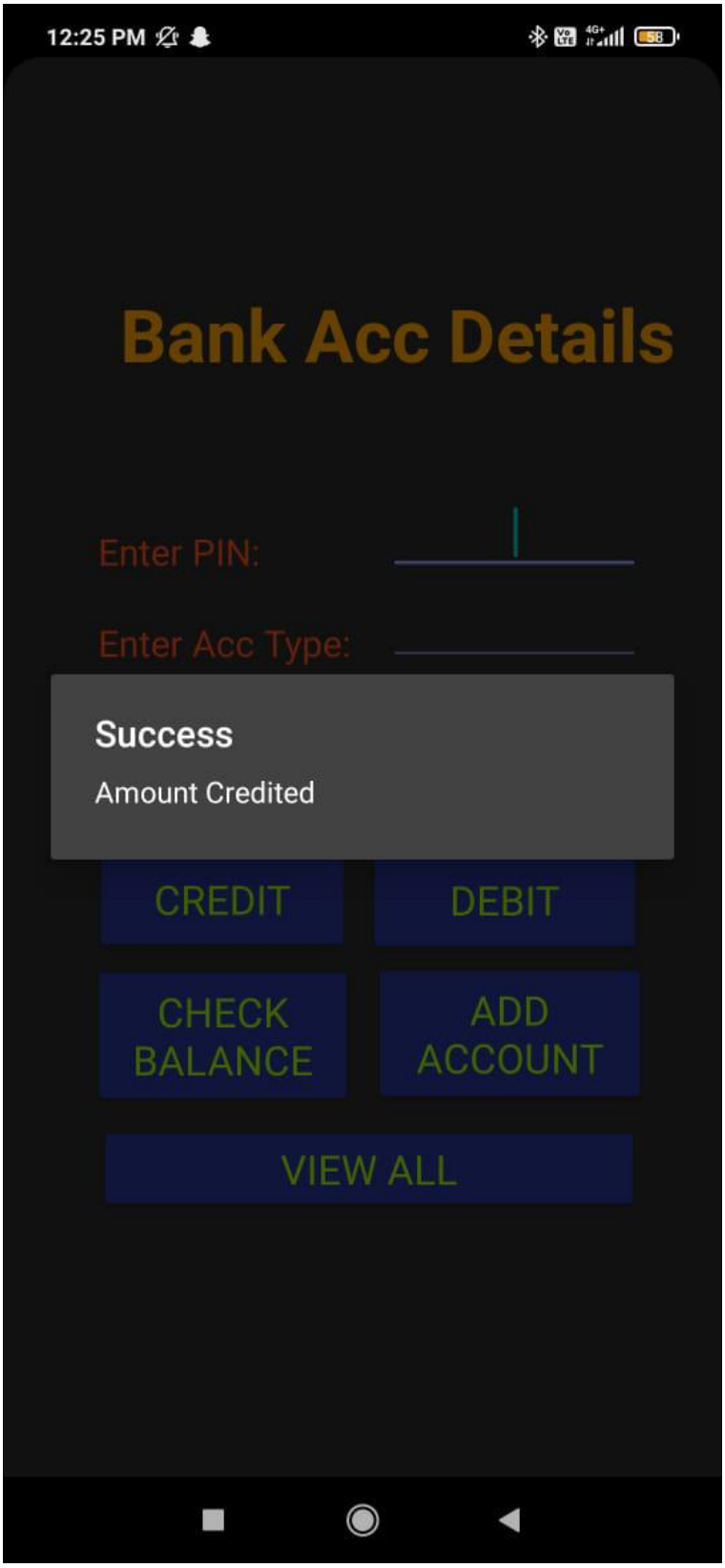

12:24 PM
Bank Acc Details
Enter PIN:
1234
Enter Acc Type:
FD
Enter Amount:
10000
CREDIT
DEBIT
CHECK BALANCE
ADD ACCOUNT
VIEW ALL

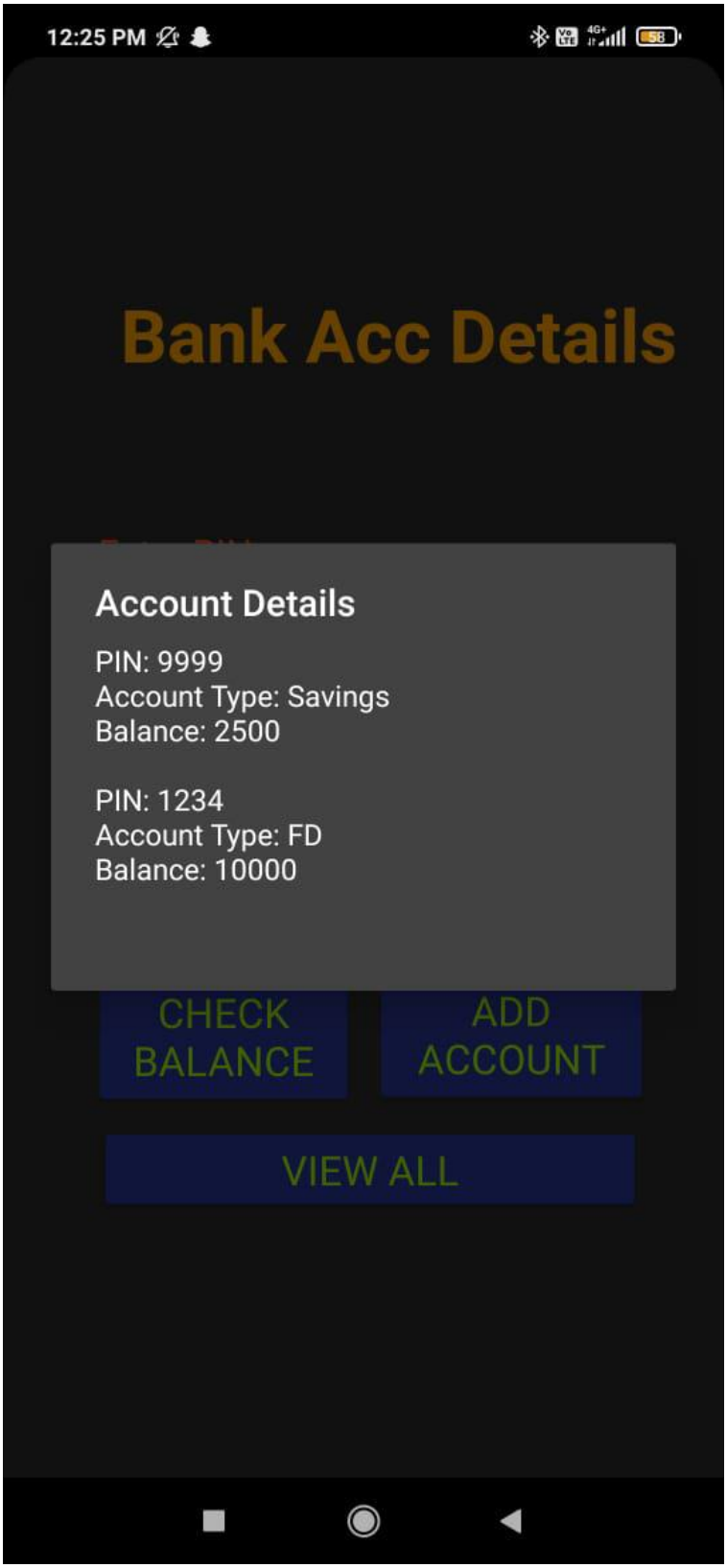


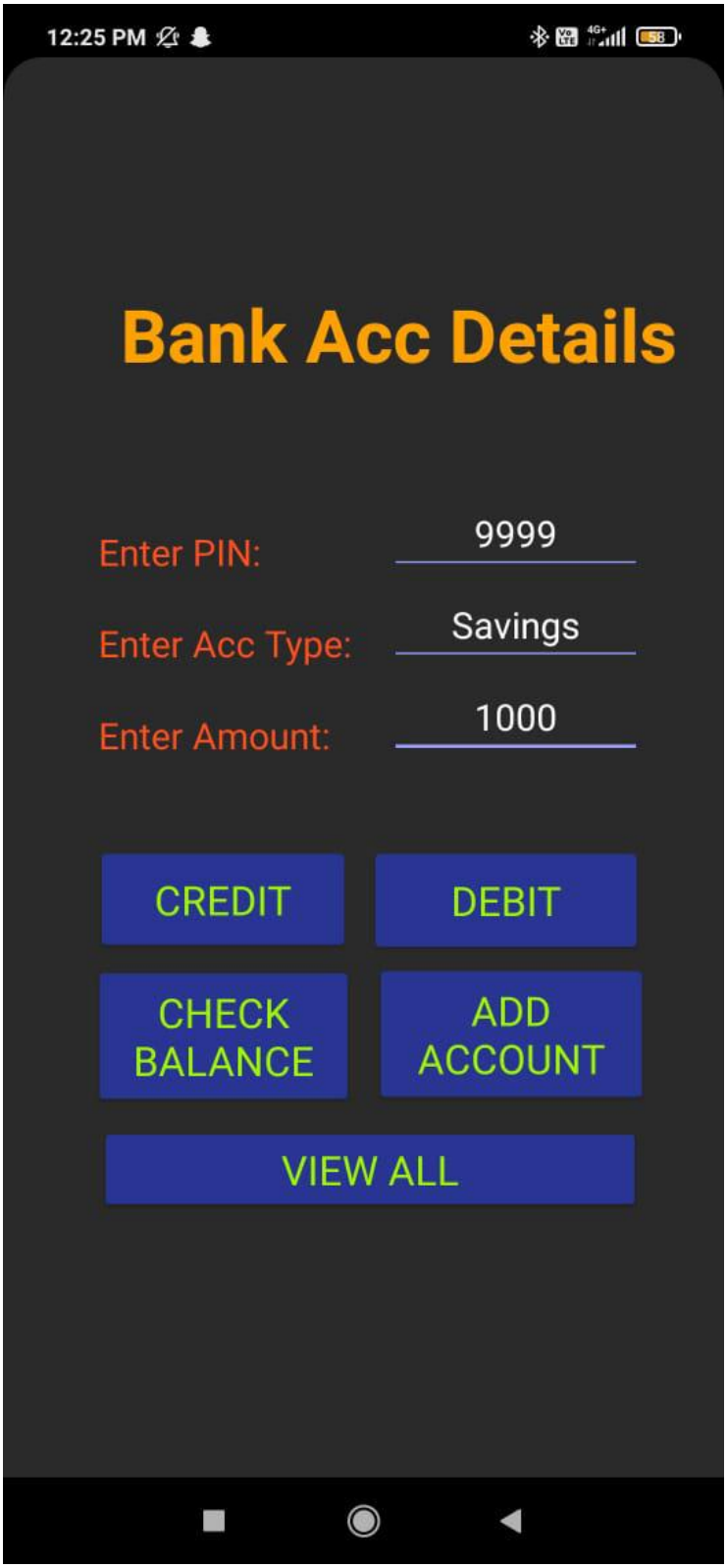

**Roll No:** 22102A0014

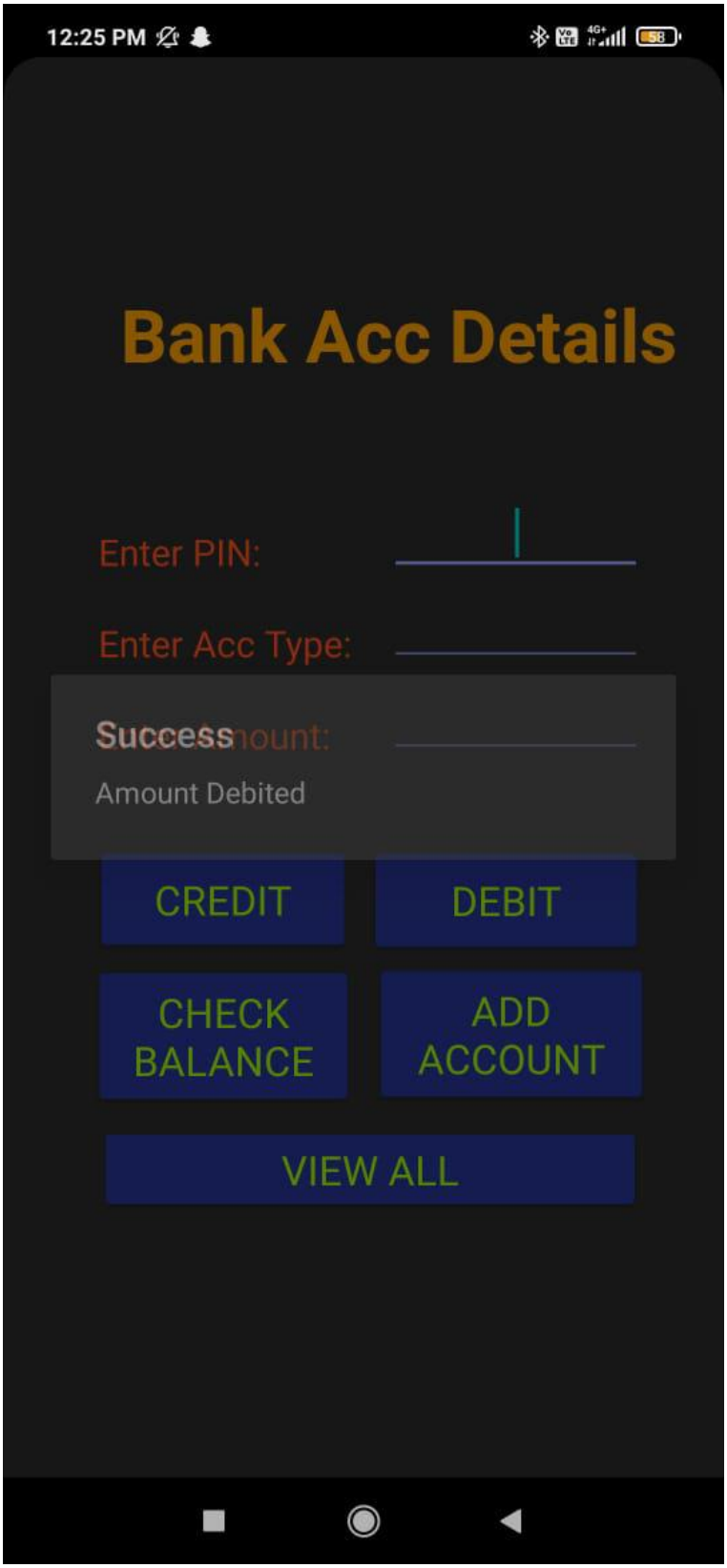

12:25 PM
Bank Acc Details
Enter PIN:
1234
Account Details
PIN: 1234
Account Type: FD
Balance: 10000
CHECK BALANCE
ADD ACCOUNT
VIEW ALL

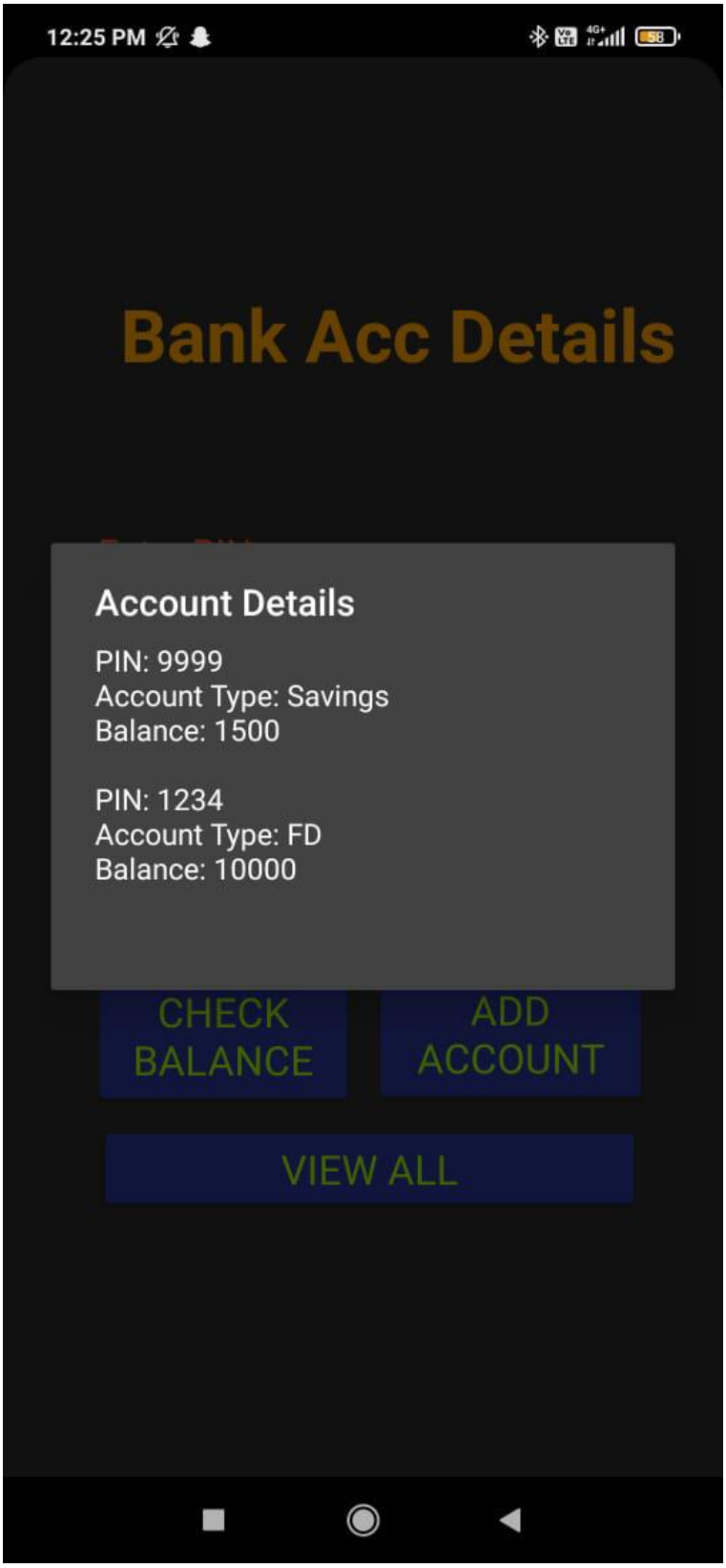

12:25 PM
Bank Acc Details
Enter PIN:
9999
Enter Acc Type:
Savings
Enter Amount:
500
CREDIT
DEBIT
CHECK BALANCE
ADD ACCOUNT
VIEW ALL


12:25 PM
Bank Acc Details
Enter PIN:
Enter Acc Type:
Success
Amount Credited
CREDIT
DEBIT
CHECK BALANCE
ADD ACCOUNT
VIEW ALL


12:25 PM
Bank Acc Details
Account Details
PIN: 9999
Account Type: Savings
Balance: 2500
PIN: 1234
Account Type: FD
Balance: 10000
CHECK BALANCE
ADD ACCOUNT
VIEW ALL


12:25 PM
Bank Acc Details
Enter PIN:
9999
Enter Acc Type:
Savings
Enter Amount:
1000
CREDIT
DEBIT
CHECK BALANCE
ADD ACCOUNT
VIEW ALL


12:25 PM
Bank Acc Details
Enter PIN:
Enter Acc Type:
Success
Amount Debited
CREDIT
DEBIT
CHECK BALANCE
ADD ACCOUNT
VIEW ALL


12:25 PM
Bank Acc Details
Account Details
PIN: 9999
Account Type: Savings
Balance: 1500
PIN: 1234
Account Type: FD
Balance: 10000
CHECK BALANCE
ADD ACCOUNT
VIEW ALL

**Conclusion:**

developing a student database app in Android Studio involves designing a robust database schema, implementing CRUD operations, designing a user-friendly interface, and ensuring smooth interaction between the UI and the database. Through proper testing and validation, the app can provide a seamless experience for managing student information. By following best practices and considering user feedback, developers can create a valuable tool for organizing and accessing student data efficiently.

| | |
|---|---|
| Semester | T.E. Semester VI – Computer Engineering |
| Subject | Mobile Computing |
| Subject Professor In-charge | Prof. Sneha Annappanavar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M310A |

| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 22102A0014 |
| TE Division | A |

**Title: Case Study(Calaculator)**

---

**Explanation:**

1. Design the UI with EditText for input and Buttons for operators.
2. Implement onClickListeners for buttons to capture input and perform calculations.
3. Write logic to handle arithmetic operations in Java or Kotlin.
4. Test the app thoroughly and debug any issues.
5. Enhance user experience with feedback and efficiency optimizations.
6. Build and distribute the app via the Google Play Store or other channels for users to download and use.

---

**Implementation:**

**activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#2A2A2A">


    <TextView
```

```
        android:id="@+id/t1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:fontFamily="@font/chela_one"
        android:gravity="center"
        android:paddingTop="50dp"
        android:text="GRADE CALCULATOR"
        android:textColor="#FF9800"
        android:textSize="42dp"
        android:textStyle="bold"  />

    <GridLayout
        android:id="@+id/gridLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:alignmentMode="alignMargins"
        android:columnCount="3"
        android:rowCount="5"
        android:columnOrderPreserved="false"
        android:rowOrderPreserved="false"
        android:layout_marginTop="30dp"
        android:layout_marginLeft="35dp"
        android:layout_marginRight="35dp">

        <TextView
            android:id="@+id/test"
            android:layout_width="136dp"
            android:layout_height="wrap_content"
            android:fontFamily="@font/bree_serif"
            android:gravity="center"
            android:text="TEST"
            android:textColor="#CDDC39"
            android:textSize="21sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/marks"
            android:layout_width="89dp"
            android:layout_height="wrap_content"
            android:fontFamily="@font/bree_serif"
            android:gravity="center"
            android:text="MARKS"
            android:textColor="#CDDC39"
```

```
            android:textSize="21sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/txtView"
            android:layout_width="101dp"
            android:layout_height="wrap_content"
            android:fontFamily="@font/bree_serif"
            android:gravity="center"
            android:text="OUT OFF"
            android:textColor="#CDDC39"
            android:textSize="21sp"
            android:textStyle="bold" />


        <TextView
            android:id="@+id/assgn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:fontFamily="@font/bree_serif"
            android:textColor="#FFFFFF"
            android:paddingTop="30dp"
            android:text="Assignment"
            android:textSize="18dp"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/txtAss"
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginStart="10dp"
            android:autofillHints=""
            android:backgroundTint="#0D23C8"
            android:fontFamily="@font/bree_serif"
            android:gravity="center"
            android:hint=""
            android:inputType="numberDecimal"
            android:minHeight="48dp"
            android:singleLine="true"
            android:textColor="#003296" />

        <EditText
            android:id="@+id/outAss"
```

```
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginStart="15dp"
            android:layout_marginLeft="16dp"
            android:autofillHints=""
            android:backgroundTint="#0D23C8"
            android:fontFamily="@font/bree_serif"
            android:gravity="center"
            android:hint=""
            android:inputType="numberDecimal"
            android:minHeight="48dp"
            android:textColor="#003296" />


        <TextView
            android:id="@+id/pro"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:fontFamily="@font/bree_serif"
            android:paddingTop="10dp"
            android:text="Project"
            android:textColor="#FFFFFF"
            android:textSize="18dp"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/txtPro"
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginStart="10dp"
            android:autofillHints=""
            android:backgroundTint="#0D23C8"
            android:fontFamily="@font/bree_serif"
            android:gravity="center"
            android:hint=""
            android:inputType="numberDecimal"
            android:minHeight="48dp"
            android:textColor="#003296" />

        <EditText
            android:id="@+id/outPro"
            android:layout_width="70dp"
```

```
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginStart="15dp"
            android:autofillHints=""
            android:backgroundTint="#0D23C8"
            android:fontFamily="@font/bree_serif"
            android:gravity="center"
            android:hint=""
            android:inputType="numberDecimal"
            android:minHeight="48dp"
            android:textColor="#003296" />

        <TextView
            android:id="@+id/mterm"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:fontFamily="@font/bree_serif"
            android:paddingTop="10dp"
            android:text="Mid Term Exam"
            android:textColor="#FFFFFF"
            android:textSize="18dp"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/txtMidt"
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginStart="10dp"
            android:autofillHints=""
            android:backgroundTint="#0D23C8"
            android:fontFamily="@font/bree_serif"
            android:gravity="center"
            android:hint=""
            android:inputType="numberDecimal"
            android:minHeight="48dp"
            android:textColor="#003296" />

        <EditText
            android:id="@+id/outMidt"
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginStart="15dp"
```

```
            android:autofillHints=""
            android:backgroundTint="#0D23C8"
            android:fontFamily="@font/bree_serif"
            android:gravity="center"
            android:hint=""
            android:inputType="numberDecimal"
            android:minHeight="48dp"
            android:textColor="#003296" />

        <TextView
            android:id="@+id/fterm"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:fontFamily="@font/bree_serif"
            android:paddingTop="10dp"
            android:text="Final Term Exam"
            android:textColor="#FFFFFF"
            android:textSize="18dp"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/txtFint"
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginStart="10dp"
            android:layout_marginLeft="16dp"
            android:autofillHints=""
            android:backgroundTint="#0D23C8"
            android:fontFamily="@font/bree_serif"
            android:gravity="center"
            android:hint=""
            android:inputType="numberDecimal"
            android:minHeight="48dp"
            android:textColor="#003296" />

        <EditText
            android:id="@+id/outFint"
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginStart="15dp"
            android:layout_marginLeft="16dp"
            android:autofillHints=""
```

```
            android:backgroundTint="#0D23C8"
            android:fontFamily="@font/bree_serif"
            android:gravity="center"
            android:hint=""
            android:inputType="numberDecimal"
            android:minHeight="48dp"
            android:textColor="#003296" />

    </GridLayout>

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center">

        <Button
            android:id="@+id/percent"
            android:layout_width="120dp"
            android:layout_height="50dp"
            android:layout_margin="3dp"
            android:backgroundTint="#003296"
            android:fontFamily="@font/bevan"
            android:gravity="center"
            android:text="%"
            android:textColor="#F85944"
            android:textSize="18sp" />

        <Button
            android:id="@+id/grade"
            android:layout_width="160dp"
            android:layout_height="50dp"
            android:layout_margin="3dp"
            android:backgroundTint="#003296"
            android:fontFamily="@font/bevan"
            android:gravity="center"
            android:text="Grade"
            android:textColor="#F85944"
            android:textSize="18sp" />

    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearLayout3"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center">

        <Button
            android:id="@+id/pointer"
            android:layout_width="200dp"
            android:layout_height="50dp"
            android:layout_gravity="center"
            android:layout_margin="1dp"
            android:backgroundTint="#003296"
            android:fontFamily="@font/bevan"
            android:gravity="center"
            android:text="Pointer"
            android:textColor="#F85944"
            android:textSize="18sp" />

    </LinearLayout>

    <TextView
        android:id="@+id/res"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp"
        android:fontFamily="@font/amaranth"
        android:gravity="center"
        android:text=""
        android:textColor="#028FFA"
        android:textSize="35sp"
        android:textStyle="bold" />

</LinearLayout>
```

**MainActivity.java:**

```
package com.example.exp11a;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
```

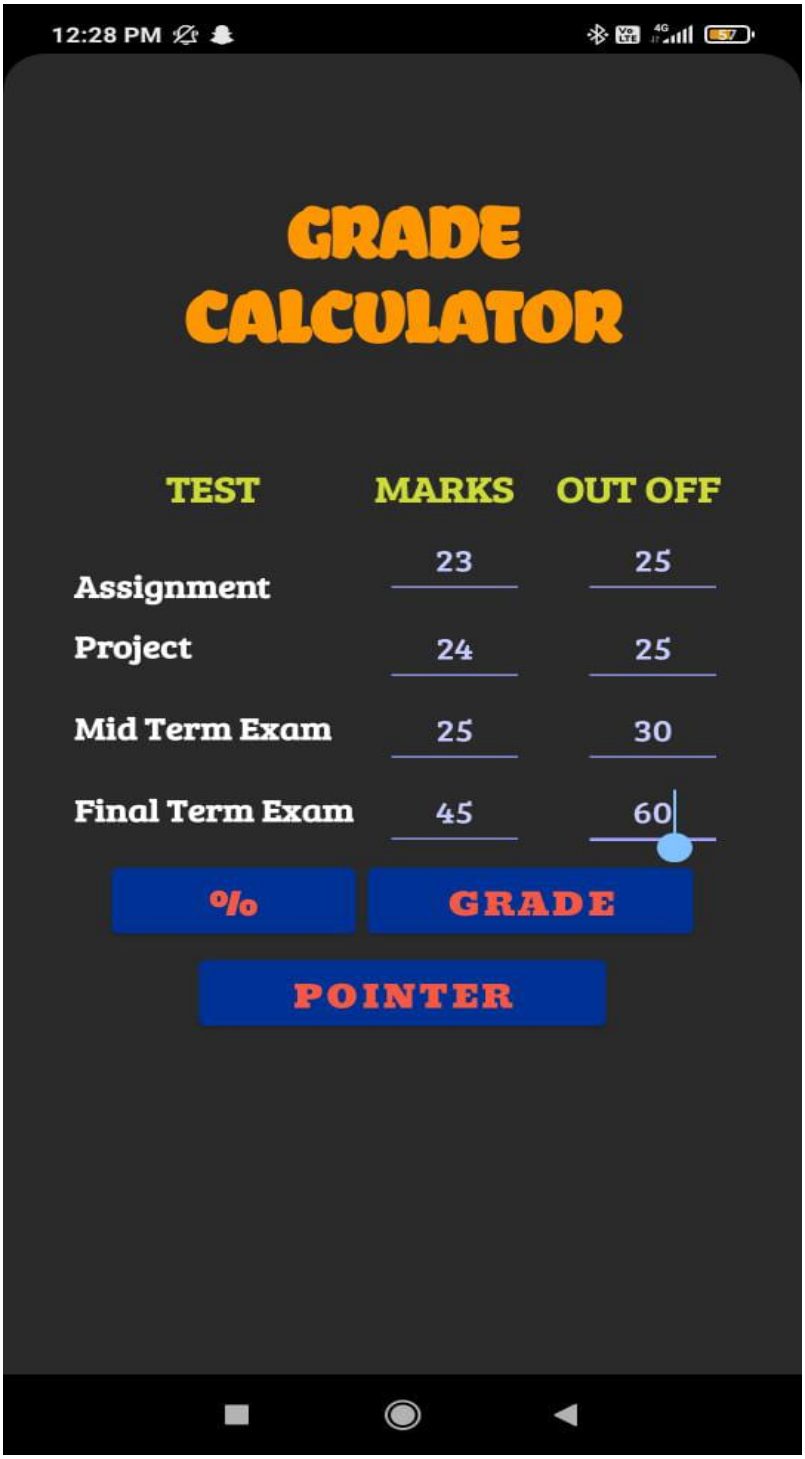
```
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity implements OnClickListener
{
    //Defining the Views
    EditText txtAss;
    EditText txtPro;
    EditText txtMidt;
    EditText txtFint;
    EditText outAss;
    EditText outPro;
    EditText outMidt;
    EditText outFint;
    Button percent;
    Button grade;
    Button pointer;

    TextView Result;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Referring the Views
        txtAss = (EditText) findViewById(R.id.txtAss);
        txtPro = (EditText) findViewById(R.id.txtPro);
        txtMidt = (EditText) findViewById(R.id.txtMidt);
        txtFint = (EditText) findViewById(R.id.txtFint);


        outAss = (EditText) findViewById(R.id.outAss);
        outPro = (EditText) findViewById(R.id.outPro);
        outMidt = (EditText) findViewById(R.id.outMidt);
        outFint = (EditText) findViewById(R.id.outFint);

        percent = (Button) findViewById(R.id.percent);
        grade = (Button) findViewById(R.id.grade);
        pointer = (Button) findViewById(R.id.pointer);

        Result = (TextView) findViewById(R.id.res);

        // set a listener
```

```
        percent.setOnClickListener(this);
        grade.setOnClickListener(this);
        pointer.setOnClickListener(this);
    }

        @Override
    public void onClick (View v)
    {

        int ass = 0;
        int pro = 0;
        int midt = 0;
        int fint = 0;

        int oass = 0;
        int opro = 0;
        int omidt = 0;
        int ofint = 0;

        float percent = 0;
        String grade = "";
        Integer pointer = 0;

        // read EditText and fill variables with numbers
        ass = Integer.parseInt(txtAss.getText().toString());
        pro = Integer.parseInt(txtPro.getText().toString());
        midt = Integer.parseInt(txtMidt.getText().toString());
        fint = Integer.parseInt(txtFint.getText().toString());

        oass = Integer.parseInt(outAss.getText().toString());
        opro = Integer.parseInt(outPro.getText().toString());
        omidt = Integer.parseInt(outMidt.getText().toString());
        ofint = Integer.parseInt(outFint.getText().toString());

        int markTot = ass + pro + midt + fint;
        int outTot = oass + opro + omidt + ofint;

        // defines the button that has been clicked and performs the
corresponding operation
        // write operation into oper, we will use it later for output
        switch (v.getId())
        {
            case R.id.percent:
                percent = percent(markTot, outTot);
```

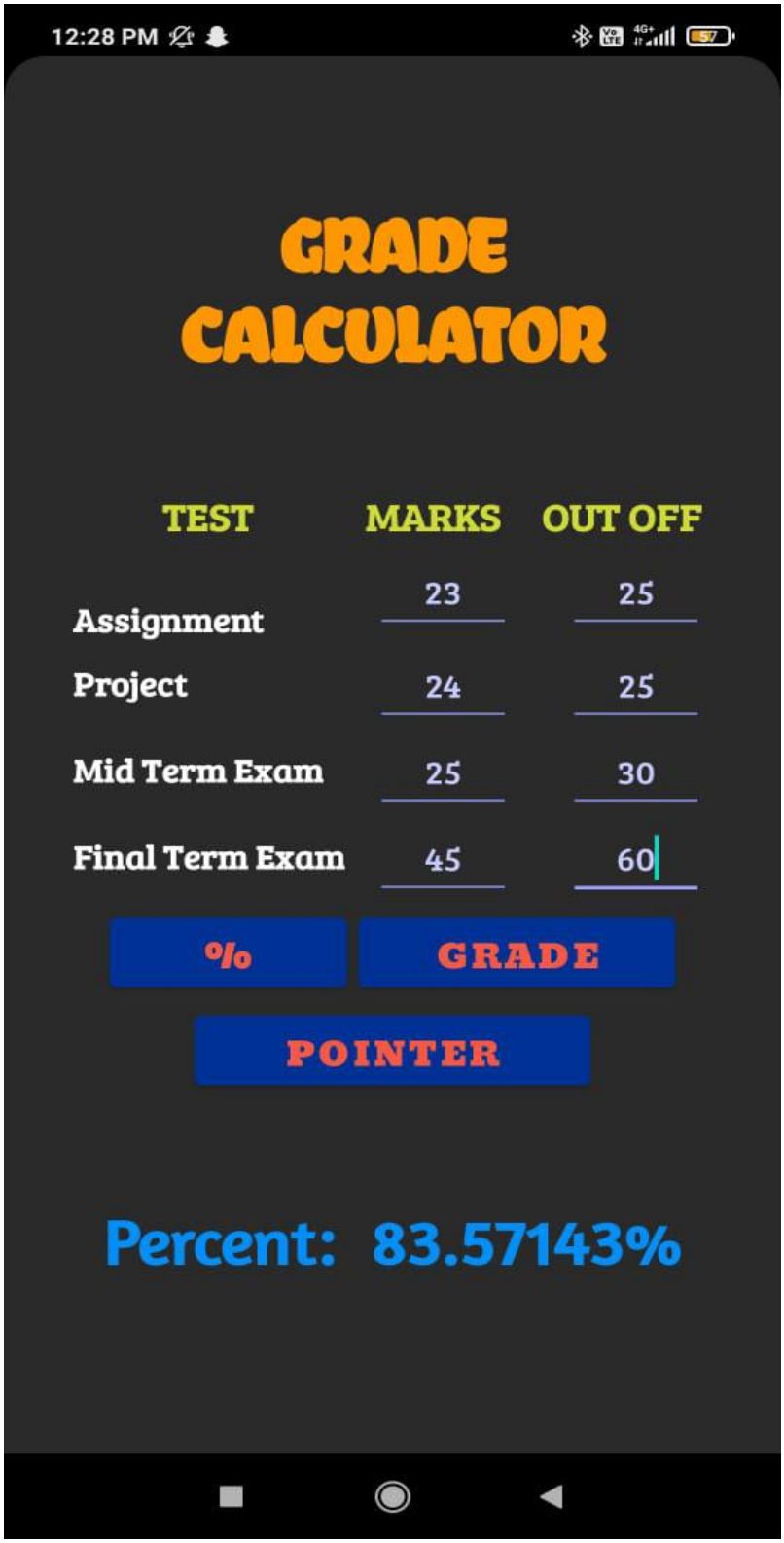
```
                break;
            case R.id.grade:
                percent = percent(markTot, outTot);
                grade = grade(percent);
                Result.setText("Grade:  " + grade);
                break;
            case R.id.pointer:
                percent = percent(markTot, outTot);
                grade = grade(percent);
                pointer = pointer(grade);
                Result.setText("Pointer:  " + pointer);
                break;
            default:
                break;
        }
    }

    private Integer pointer(String grade) {
        Integer pointer = 0;
        if(grade == "O"){
            pointer = 10;
        }
        else if(grade == "A"){
            pointer = 9;
        }
        else if(grade == "B"){
            pointer = 8;
        }
        else if(grade == "C"){
            pointer = 7;
        }
        else if(grade == "D"){
            pointer = 6;
        }
        else if(grade == "E"){
            pointer = 5;
        }
        else{
            pointer = 4;
        }

        return pointer;
    }
```

```
    private String grade(float percent) {
        String grade = "";
        if(percent >= 89.50 && percent <= 100.00){
            grade = "O";
        }
        else if(percent >= 79.50 && percent < 89.50){
            grade = "A";
        }
        else if(percent >= 69.50 && percent < 79.50){
            grade = "B";
        }
        else if(percent >= 59.50 && percent < 69.50){
            grade = "C";
        }
        else if(percent >= 49.50 && percent < 59.50){
            grade = "D";
        }
        else if(percent >= 35.50 && percent < 49.50){
            grade = "E";
        }
        else{
            grade = "F";
        }

        return grade;
    }

    public float percent(int markTot, int outTot) {
            float percent = ((float)markTot / outTot) * 100;
            Result.setText("Percent:  " + percent + "%");
            return percent;
    }

}
```

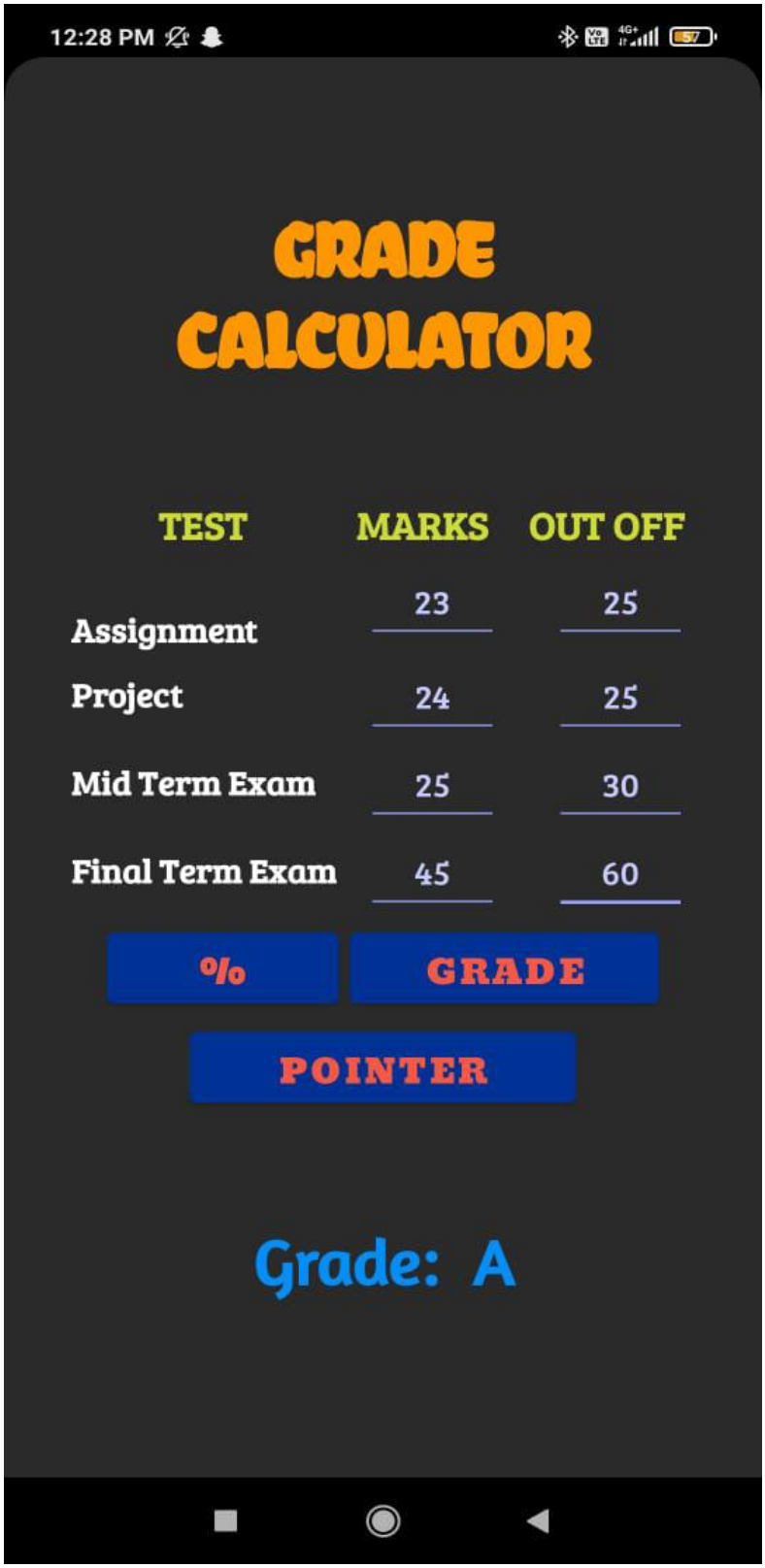
**Output:**



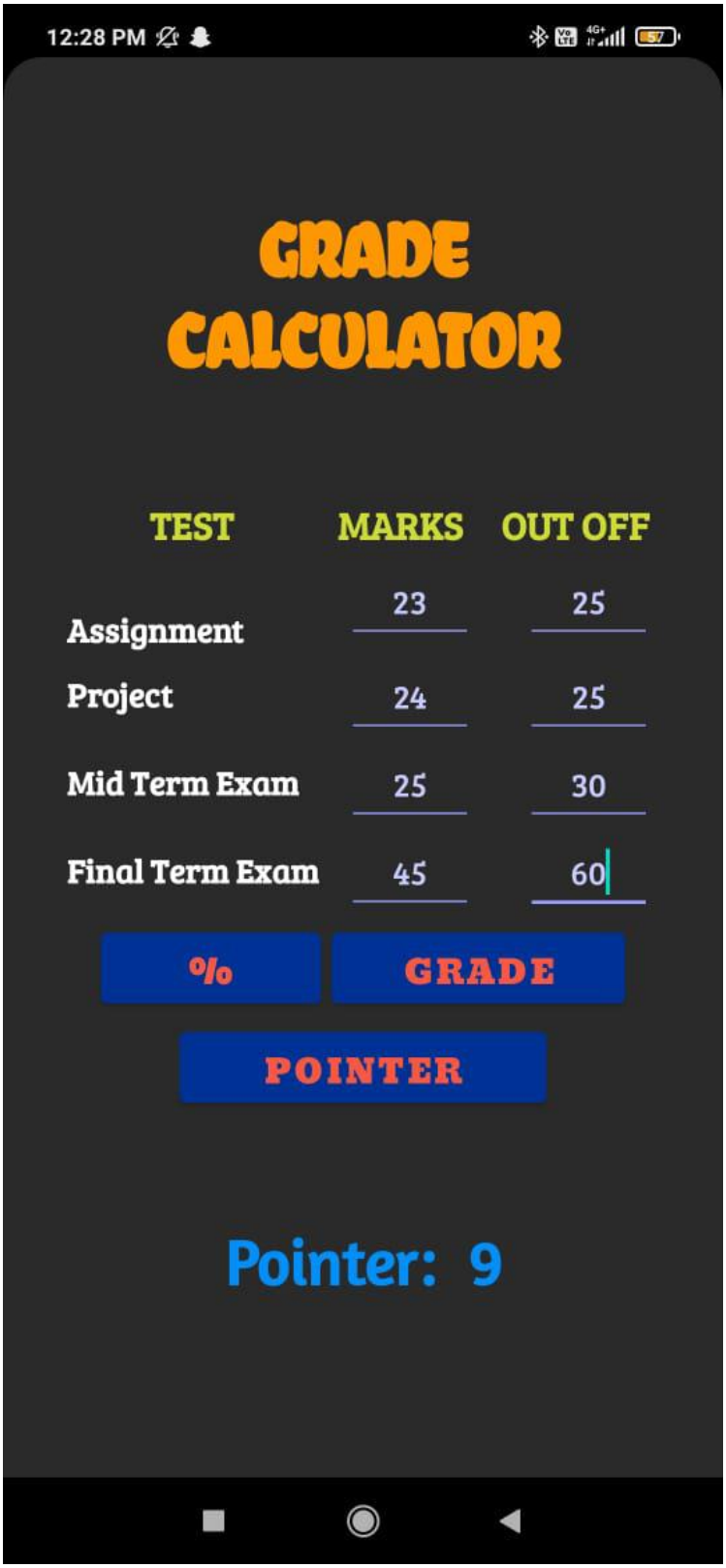

**Title:** calculator **Roll No:** 22102A2014




**Title:** calculator **Roll No:** 22102A2014




**Title:** calculator **Roll No:** 22102A2014




**Title:** calculator **Roll No:** 22102A2014

**Conclusion:**

Android Studio, developers can create a calculator app by integrating EditText for input, Buttons for operations, and TextView for displaying results. By implementing onClickListeners and logic for arithmetic operations, along with layout managers for UI organization, a functional and user-friendly calculator can be built to meet users' needs.

# DEPARTMENT OF COMPUTER ENGINEERING

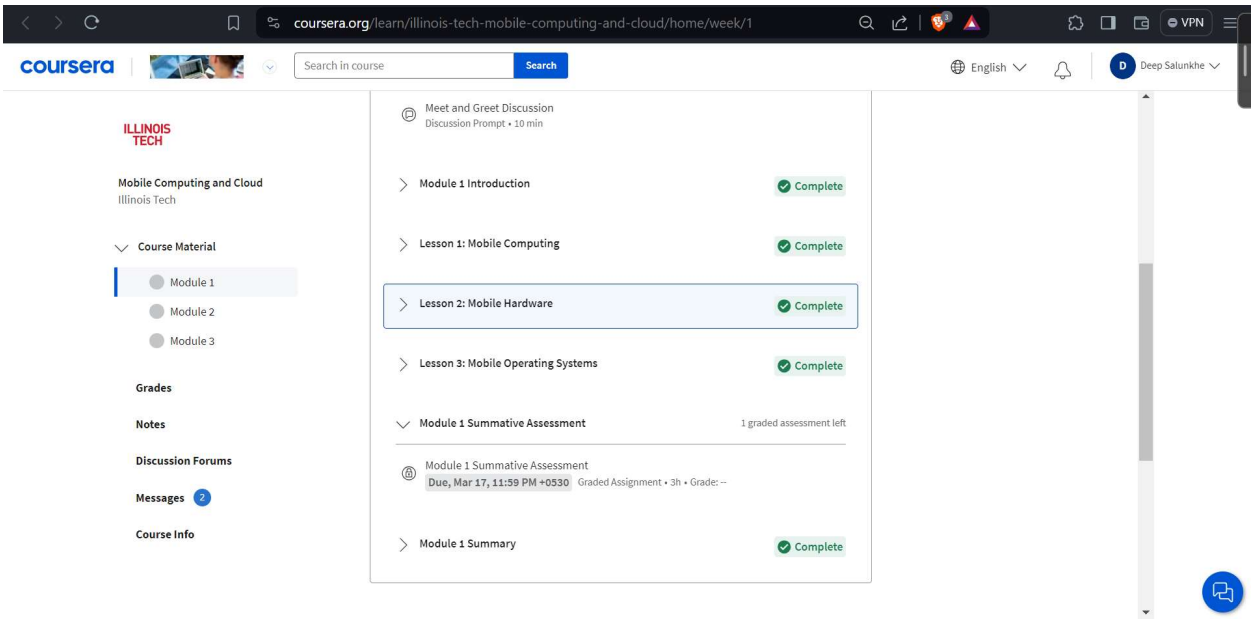
| | |
|---|---|
| Semester | T.E. Semester VI – Computer Engineering |
| Subject | Mobile Computing |
| Subject Professor In-charge | Prof. Sneha Annappanavar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M310A |

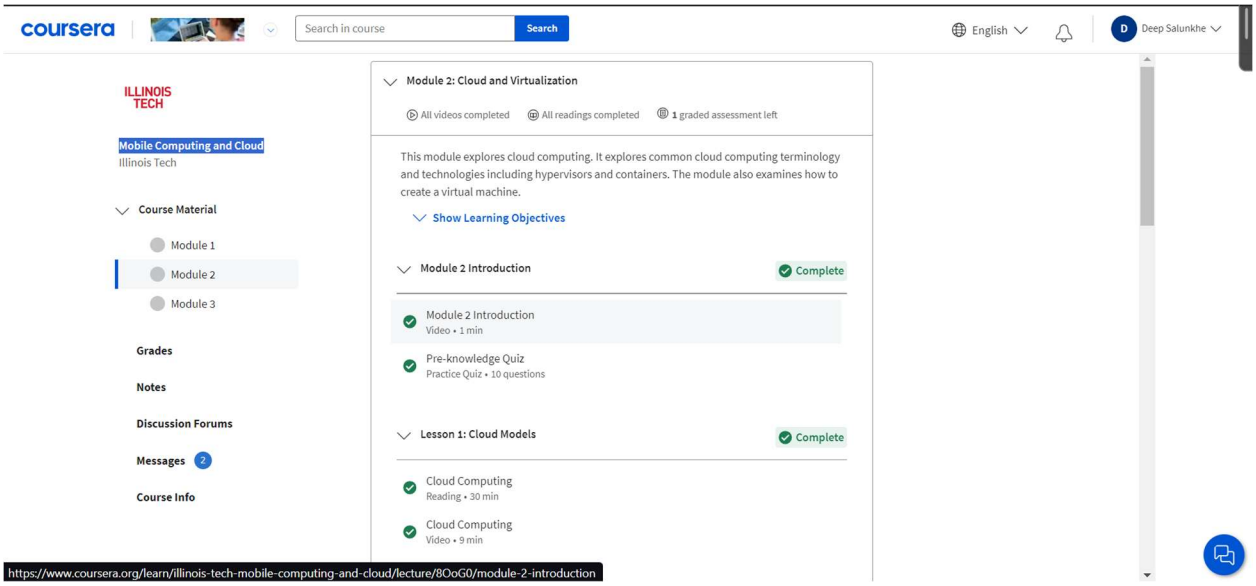
| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 21102A0014 |
| TE Division | A |

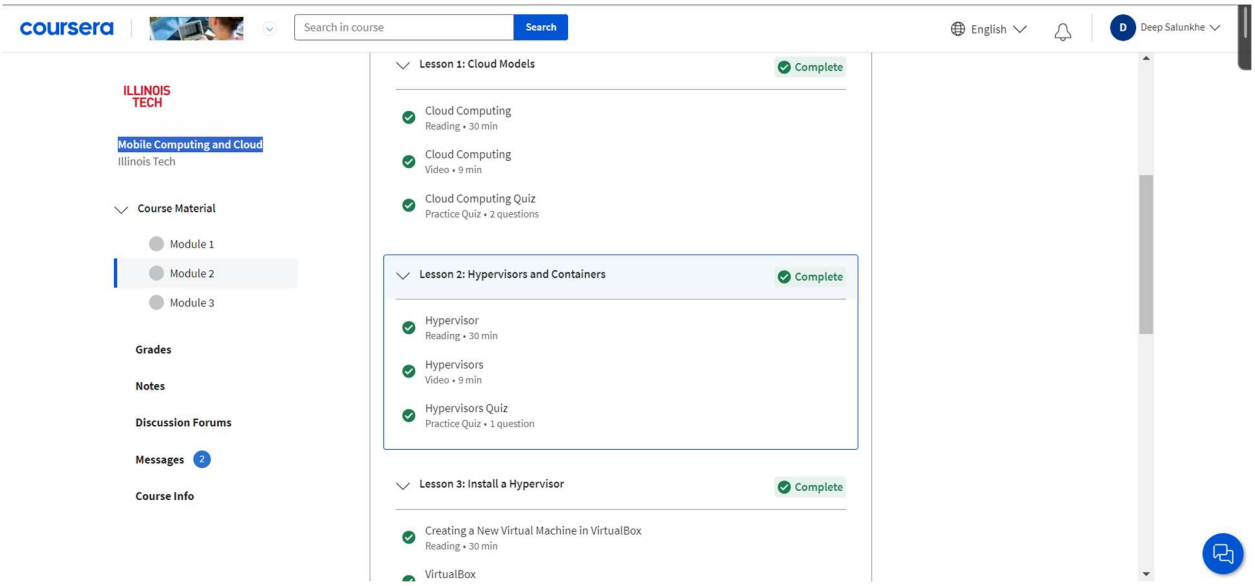
**Title:**

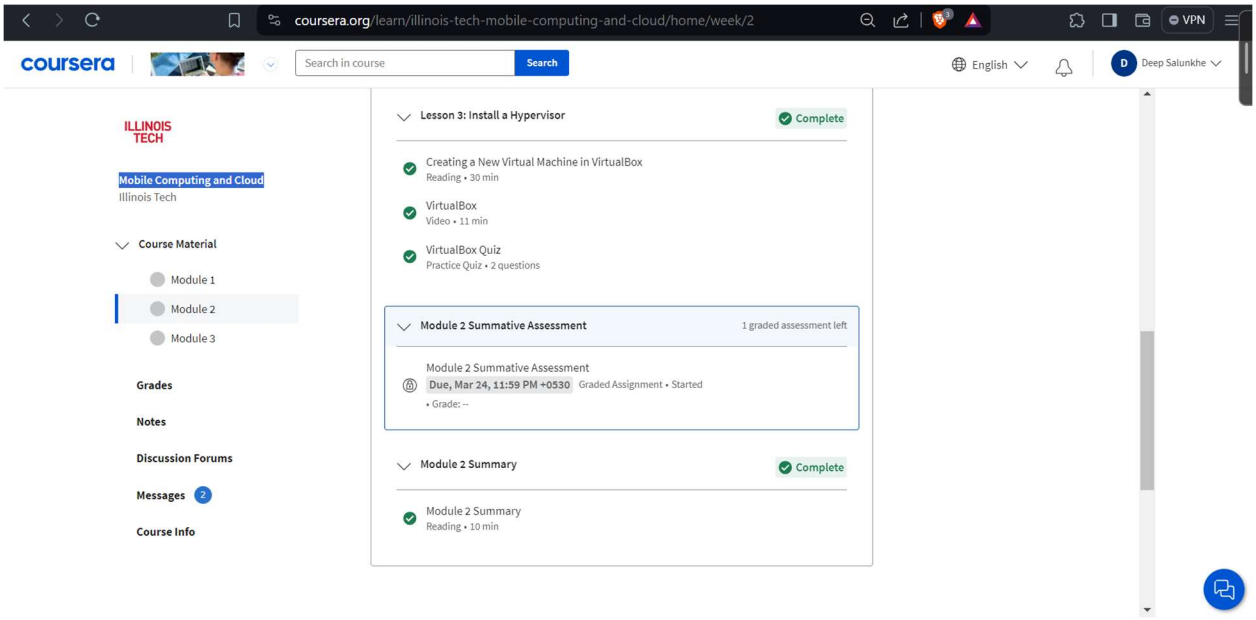
**Mobile Computing and Cloud**

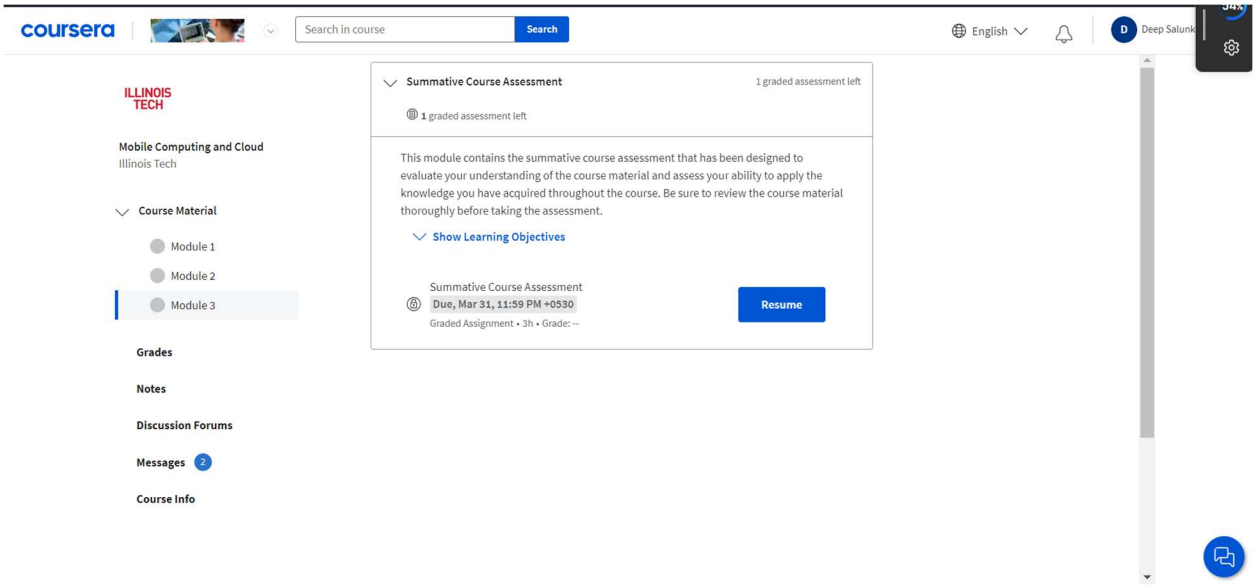
---

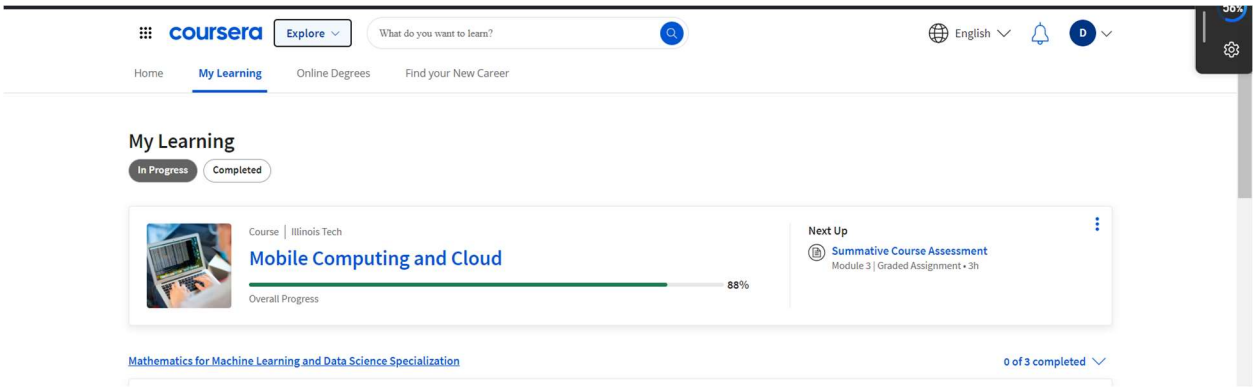



coursera
Search in course
Search
English
Deep Salunkhe
ILLINOIS TECH
Mobile Computing and Cloud
Illinois Tech
Course Material
Module 1
Module 2
Module 3
Grades
Notes
Discussion Forums
Messages 2
Course Info
Module 2: Cloud and Virtualization
All videos completed
All readings completed
1 graded assessment left
This module explores cloud computing. It explores common cloud computing terminology and technologies including hypervisors and containers. The module also examines how to create a virtual machine.
Show Learning Objectives
Module 2 Introduction
Complete
Module 2 Introduction
Video • 1 min
Pre-knowledge Quiz
Practice Quiz • 10 questions
Lesson 1: Cloud Models
Complete
Cloud Computing
Reading • 30 min
Cloud Computing
Video • 9 min
https://www.coursera.org/learn/illinois-tech-mobile-computing-and-cloud/lecture/8OoG0/module-2-introduction


coursera
Search in course
Search
English
Deep Salunkhe
ILLINOIS TECH
Mobile Computing and Cloud
Illinois Tech
Course Material
Module 1
Module 2
Module 3
Grades
Notes
Discussion Forums
Messages 2
Course Info
Lesson 1: Cloud Models
Complete
Cloud Computing
Reading • 30 min
Cloud Computing
Video • 9 min
Cloud Computing Quiz
Practice Quiz • 2 questions
Lesson 2: Hypervisors and Containers
Complete
Hypervisor
Reading • 30 min
Hypervisors
Video • 9 min
Hypervisors Quiz
Practice Quiz • 1 question
Lesson 3: Install a Hypervisor
Complete
Creating a New Virtual Machine in VirtualBox
Reading • 30 min
VirtualBox


coursera.org/learn/illinois-tech-mobile-computing-and-cloud/home/week/2
coursera
Search in course
Search
English
Deep Salunkhe
ILLINOIS TECH
Mobile Computing and Cloud
Illinois Tech
Course Material
Module 1
Module 2
Module 3
Grades
Notes
Discussion Forums
Messages 2
Course Info
Lesson 3: Install a Hypervisor
Complete
Creating a New Virtual Machine in VirtualBox
Reading • 30 min
VirtualBox
Video • 11 min
VirtualBox Quiz
Practice Quiz • 2 questions
Module 2 Summative Assessment
1 graded assessment left
Module 2 Summative Assessment
Due, Mar 24, 11:59 PM +0530 Graded Assignment • Started
• Grade: --
Module 2 Summary
Complete
Module 2 Summary
Reading • 10 min


coursera
Search in course
Search
English
ILLINOIS TECH
Mobile Computing and Cloud
Illinois Tech
Course Material
Module 1
Module 2
Module 3
Grades
Notes
Discussion Forums
Messages 2
Course Info
Summative Course Assessment
1 graded assessment left
1 graded assessment left
This module contains the summative course assessment that has been designed to evaluate your understanding of the course material and assess your ability to apply the knowledge you have acquired throughout the course. Be sure to review the course material thoroughly before taking the assessment.
Show Learning Objectives
Summative Course Assessment
Due, Mar 31, 11:59 PM +0530
Graded Assignment • 3h • Grade: --
Resume




Conclusion:

In this course we got introduced to the world of cloud computing and cloud architecture we got to know how these two technologies work hand in hand, what are various OS for that are used in mobile devices, about security and privacy related concerns related to these technologies