Fr. Conceicao Rodrigues College of Engineering
Department of Computer Engineering
Academic year -2023-24

Class :TE Comp B
Subject: Mobile Computing (CSL 603)

Student Name: Emmanuel Gudinho
Roll No: 9609

Sr no.	Name of the experiment	Date of performance
1	Implementation of Mobile Network using Network Simulator (NS2): Create a Mobile Ad hoc network.	02/02/24
2	To implement a Bluetooth network with application as transfer of a file from one device to another.	02/02/42
3	To understand the cellular frequency reuse concept to find the co-channel cells for a particular cell.	17/02/24
4	Illustration of Hidden Terminal Problem (NS-2).	01/03/24
5	To implement basic functions of CDMA.	04/03/24
6	Develop an application that uses GUI components.	04/03/24
7	To implement GSM security algorithms(A3/A5/A8).	18/03/24
8	To make an application that draws basic graphical primitives on the screen.	25/03/24
9	Develop an application that makes use of database.	29/03/24
10	Implement an application that creates an alert upon receiving a message.	29/03/24
	Assignment1	20/1/24
	Assignment2	23/2/24
	Assignment 3	28/2/24

17.4.24

PRACTICAL-

1

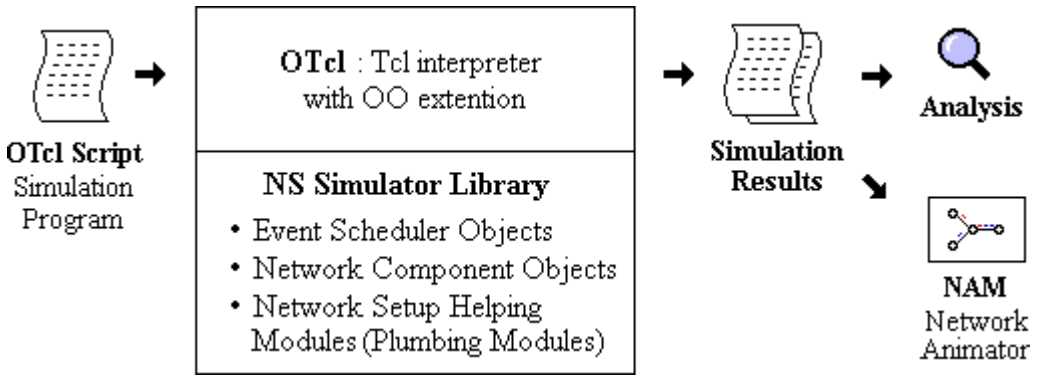
Title: Implementation of Mobile Network using Network Simulator (NS2): Create a Mobile Ad hoc network

Objective: To study Routing in MANET

Pre-Requisite: Basic knowledge of wireless networking

Description:

NS (version 2) is an object-oriented, discrete event driven network simulator developed at UC Berkely written in C++ and OTcl. NS is primarily useful for simulating local and wide area networks. It implements network protocols such as TCP and UPD, traffic source behavior such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms such as Dijkstra, and more. NS also implements multicasting and some of the MAC layer protocols for LANsimulations.



Program description:

Each agent keep track of what messages it has seen and only forwards those which it has seen and only forwards those which it hasn't seen before. Each message is of the form "ID:DATA" where ID is some arbitrary message identifier and DATA is the payload. In order to reduce memory usage , the agent store only the messageID.

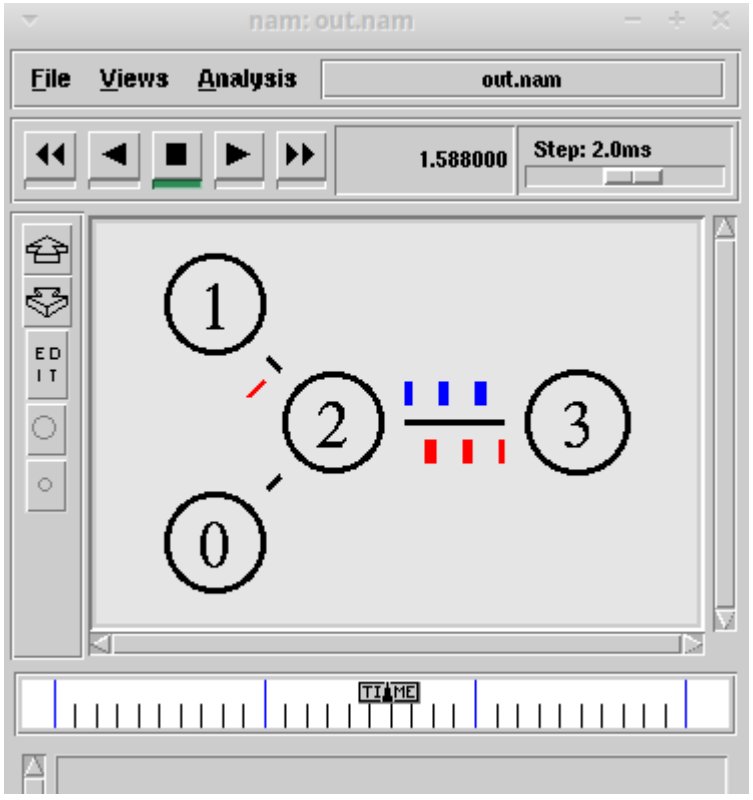
Steps:

1. Set the following configuration for each node'sinterface
   - Type of channel -WirelessChannel
   - Type of propagation –TwoRayGround
   - Physical Layer –Wireless
   - Mac Layer – MAC802.11
   - Type of Queue –DropTail/PriQueue
   - LinkLayer –LL
   - Type of Antenna–OmniAntenna
   - Maximum Packet in Queue -50
2. Open Trace file in write mode
3. Open NAM file in write mode.
4. Create a topology containing 6 groups each having 4 nodes. Use Flat Grid topology

5. Configure each node using the configuration set in step1.
6. Create a simple Message Passing/Flooding agent
7. Create Receive procedure that receives each packet and maintain list of unseen messages
8. Create send procedure that broadcasts message.
9. Create Message Passing/Flooding agent and attach it with every node.
10. Set up some events.
11. Write finish procedure.

Conclusion: Mobile networks using NS2 has been studied and implemented successfully.

```
universe@lenovo18:~/Desktop/      ns simple.tcl
210
0.0037499999999999999
running nam...
universe@lenovo18:~/Desktop/      awk -f exp.awk out.tr
137140  105840  1.00    3.00
universe@lenovo18:~/Desktop/
```



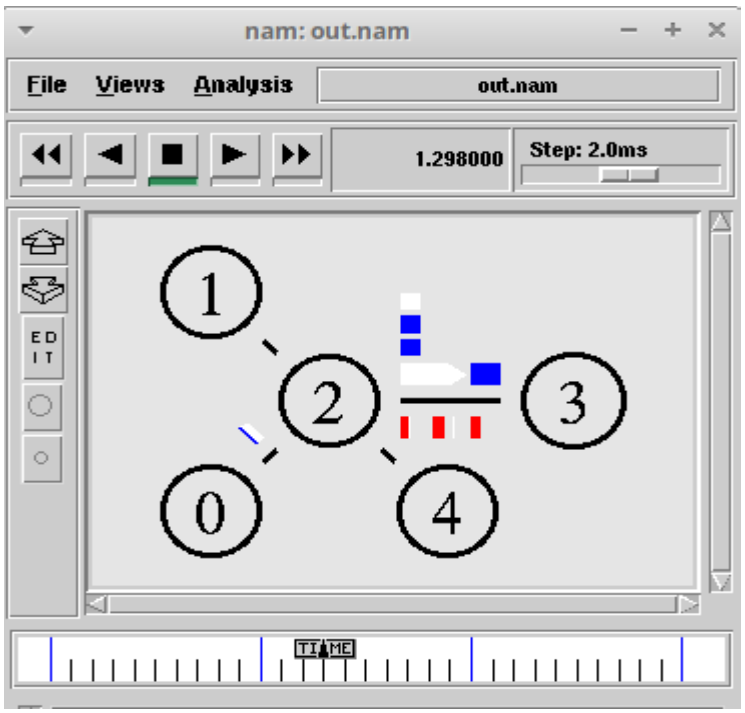
set ns [new Simulator]

$ns color 0 blue
$ns color 1 red
$ns color 2 white

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

set f [open out.tr w]
$ns trace-all $f
set nf [open out.nam w]
$ns namtrace-all $nf

$ns duplex-link $n0 $n2 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 5Mb 2ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
$ns duplex-link $n2 $n4 1.5Mb 10ms DropTail

$ns duplex-link-op $n0 $n2 orient right-up
$ns duplex-link-op $n1 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n2 $n4 orient right-down

$ns duplex-link-op $n2 $n3 queuePos 0.5

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0

set udp1 [new Agent/UDP]
$ns attach-agent $n3 $udp1
$udp1 set class_ 1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1

set null0 [new Agent/Null]
$ns attach-agent $n3 $null0

set null1 [new Agent/Null]
$ns attach-agent $n1 $null1

set null2 [new Agent/Null]
$ns attach-agent $n4 $null2

$ns connect $udp0 $null0
$ns connect $udp1 $null1

$ns at 1.0 "$cbr0 start"
$ns at 1.1 "$cbr1 start"
```

```
set tcp [new Agent/TCP]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 1.2 "$ftp start"

$ns at 1.35 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

puts [$cbr0 set packetSize_]
puts [$cbr0 set interval_]

$ns at 3.0 "finish"

proc finish {} {
    global ns f nf
    $ns flush-trace
    close $f
    close $nf

    puts "running nam..."
    exec nam out.nam &
    exit 0
}

$ns run
```

```
universe@lenovo18:~/Desktop/      awk -f exp.awk out.tr
200580  105840  1.00    3.00
```



Post Lab Questions:

1. Describe your observations about output.
   The TCL script uses ns-2 to simulate a 4-node network with duplex links and TCP traffic. It generates trace and visualization files for analysis using nam. Users can customize parameters to observe network behavior, including packet transmissions and link utilization, in the resulting trace files.

2. Explain the working of DSDV protocol.
   Destination Sequenced Distance Vector Routing protocol is a modified version of Bellman Ford Algorithm and is based upon the concepts of Distance Vector Routing.

   In Distance Vector Routing(DVR), each node broadcasts a table containing its distance from nodes which are directly connected and based upon this, other nodes broadcasts the updated routing. Those nodes which are unreachable directly are labelled as “infinite”.

   But, this updation of routing tables keeps on happening and an infinite loop is generated which is commonly known as Count-To-Infinity problem.

   To overcome this problem of count to infinity by generating sequence number in the routing table, every time the routing table is updated. The process of DSDV is same as that of Distance Vector Routing but an extra attribute of sequence number is added.

   DSDV protocol uses and maintains a single table only, for every node individually. The table contains the following attributes.

Routing Table : It contains the distance of a node from all the neighboring nodes along with the sequence number(SEQ No means the time at which table is updated).

Format :

Node	Destination	Next Hop	Distance	SEQ No

Destination Sequenced Distance Vector Routing : Format

This table is updated on every step and ensures that each node broadcast as well as receives correct information about all the nodes including their distance and sequence number.

Experiment No.: 2

Aim: *To* implement a Bluetooth network with application as transfer of a file from one device to another.

Theory:

Bluetooth is a wireless technology standard used for exchanging data between fixed and mobile devices over short distances using UHF radio waves in the industrial, scientific and medical radio bands, from 2.402 GHz to 2.480 GHz, and building personal area networks (PANs). It was originally conceived as a wireless alternative to RS-232 data cables.
Bluetooth is managed by the Bluetooth Special Interest Group (SIG), which has more than 35,000 member companies in the areas of telecommunication, computing, networking, and consumer electronics. The IEEE standardized Bluetooth as IEEE 802.15.1, but no longer maintains the standard. The Bluetooth SIG oversees development of the specification, manages the qualification program, and protects the trademarks. A manufacturer must meet Bluetooth SIG standards to market it as a Bluetooth device.

Transfer of words between two phones using Bluetooth is done below.
https://www.youtube.com/watch?v=5M4o5dGigbY

CODE:
Client (exp2.c):

```
#include <arpa/inet.h> // inet_addr()
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h> // bzero()
#include <sys/socket.h>
#include <unistd.h> // read(), write(), close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
char buff[MAX];
int n;
for (;;) {
bzero(buff, sizeof(buff));
printf("Enter the string : ");
n = 0;
while ((buff[n++] = getchar()) != '\n')
;
```

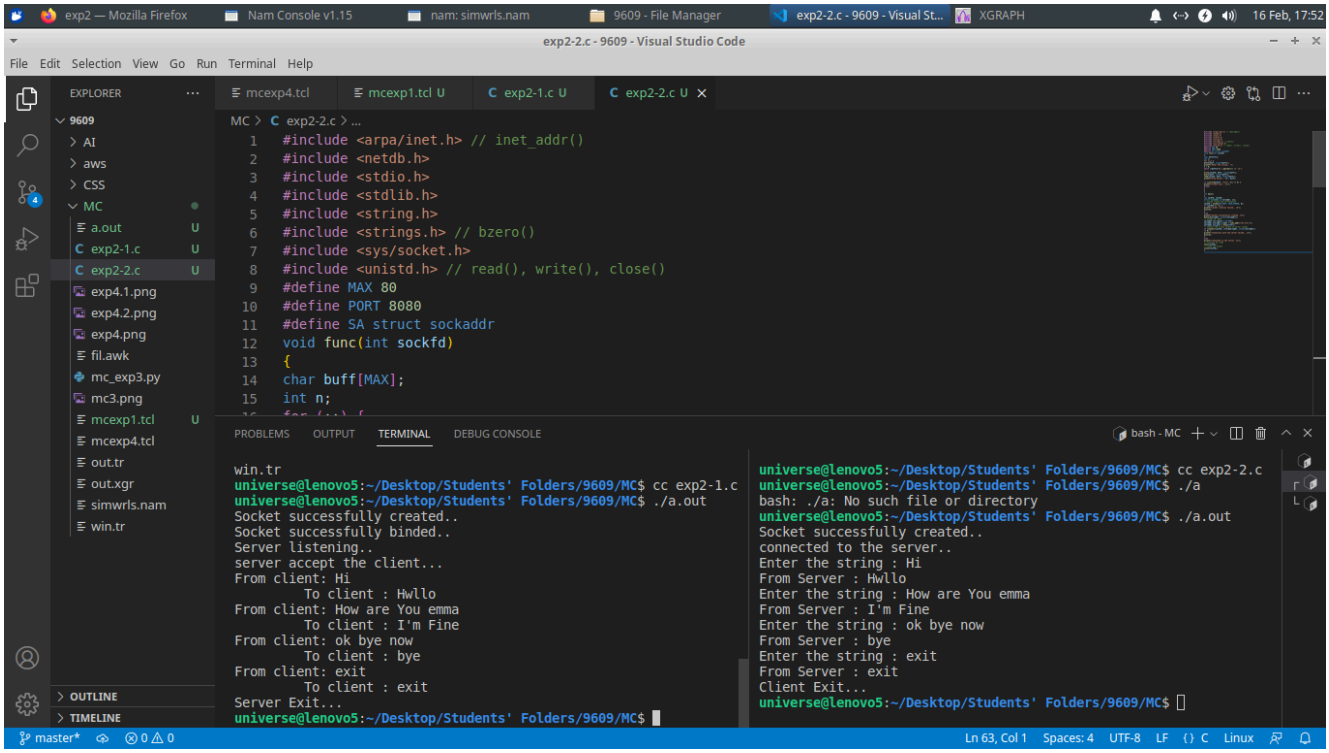
```
write(sockfd, buff, sizeof(buff));
bzero(buff, sizeof(buff));
read(sockfd, buff, sizeof(buff));
printf("From Server : %s", buff);

if ((strncmp(buff, "exit", 4)) == 0) {
printf("Client Exit...\n");
break;
}
}
}
int main()
{
int sockfd, connfd;
struct sockaddr_in servaddr, cli;
// socket create and verification
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd == -1) {
printf("socket creation failed...\n");
exit(0);
}
else
printf("Socket successfully created..\n");
bzero(&servaddr, sizeof(servaddr));
// assign IP, PORT
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
servaddr.sin_port = htons(PORT);
// connect the client socket to server socket
if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr))
!= 0) {
printf("connection with the server failed...\n");
exit(0);
}
else
printf("connected to the server..\n");
// function for chat
func(sockfd);
// close the socket
close(sockfd);
}
```

Server (exp2c.c):

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>

#include <unistd.h> // read(), write(), close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
// Function designed for chat between client and server.
void func(int connfd)
{
char buff[MAX];
int n;
// infinite loop for chat
for (;;) {
bzero(buff, MAX);
// read the message from client and copy it in buffer
read(connfd, buff, sizeof(buff));
// print buffer which contains the client contents
printf("From client: %s\t To client : ", buff);
bzero(buff, MAX);
n = 0;
// copy server message in the buffer
while ((buff[n++] = getchar()) != '\n')
;
// and send that buffer to client
write(connfd, buff, sizeof(buff));
// if msg contains "Exit" then server exit and chat ended.
if (strncmp("exit", buff, 4) == 0) {
printf("Server Exit...\n");
break;
}
}
}
// Driver function
int main()
{
int sockfd, connfd, len;
struct sockaddr_in servaddr, cli;
```

```
// socket create and verification
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd == -1) {
printf("socket creation failed...\n");
exit(0);
}
else
printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));
// assign IP, PORT
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);
// Binding newly created socket to given IP and verification
if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
printf("socket bind failed...\n");
exit(0);
}
else
printf("Socket successfully binded..\n");
// Now server is ready to listen and verification
if ((listen(sockfd, 5)) != 0) {
printf("Listen failed...\n");
exit(0);
}
else
printf("Server listening..\n");
len = sizeof(cli);
// Accept the data packet from client and verification
connfd = accept(sockfd, (SA*)&cli, &len);
if (connfd < 0) {
printf("server accept failed...\n");
exit(0);
}
else
printf("server accept the client...\n");
// Function for chatting between client and server
func(connfd);
// After chatting close the socket
close(sockfd);
}
```

OUTPUT:



Conclusion:

Thus, we have performed successfully the experiment of transferring data between two mobile phone using Bluetooth network and after that have checked and it performed.

Experiment No.: 3

Aim: To understand the cellular frequency reuse concept to find the co-channel cells for a particular cell.

Theory:

In mobile communication systems a slot of a carrier frequency / code in a carrier frequency is a radio resource unit. This radio resource unit is assigned to a user in order to support a call/ session. The number of available such radio resources at a base station thus determines the number of users who can be supported in the call. Since in wireless channels a signal is "broadcast" i.e. received by all entities therefore one a resource is allocated to a user it cannot be reassigned until the user finished the call/ session. Thus, the number of users who can be supported in a wireless system is highly limited.

In order to support a large no. of users within a limited spectrum in a region the concept of frequency re-use is used.

The signal radiated from the transmitter antenna gets attenuated with increasing distance. At a certain distance the signal strength falls below noise threshold and is no longer identifiable.

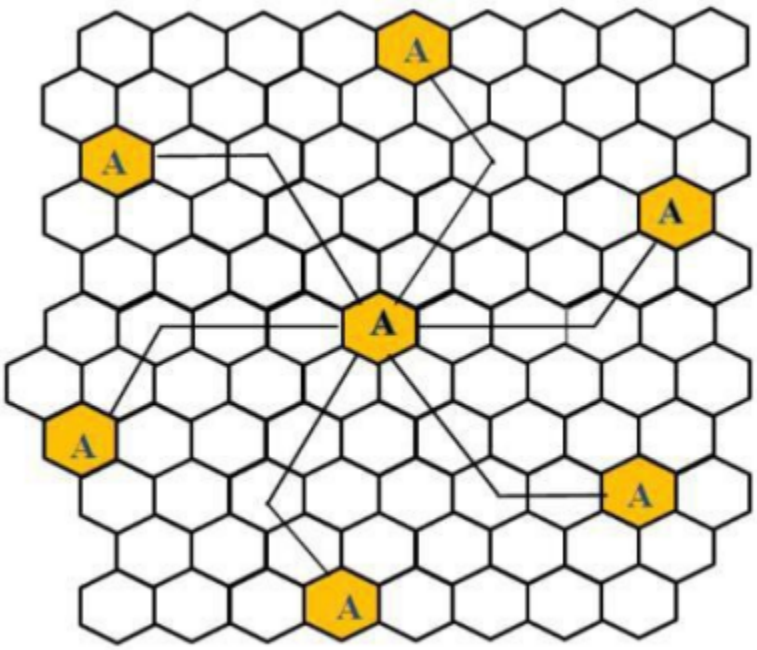
Cellular Frequency Reuse:

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.

Co-channel Cells:

A larger cluster size causes the ratio between the cell radius and the distance between co-channel cells to decrease reducing co-channel interference. The value of N is a function of how much interference a mobile or base station can tolerate while maintaining a sufficient quality of communications. Since each hexagonal cell has six equidistant neighbors and the line joining the centers of any cell and each of its neighbors are separated by multiples of 60 degrees, only certain cluster sizes and cell layouts are possible. To connect without gaps between adjacent cells, the geometry of hexagons is such that the numbers of cells per cluster, N, can only have values that satisfy,

`N=i^2+ij+j^2`



Method of locating co-channel cells in a cellular system. In this figure, N=19(i.e, i =3, j=2).

In this example N = 19 (i.e., i = 3, j = 2).

Where,

i and j are non-negative integers.

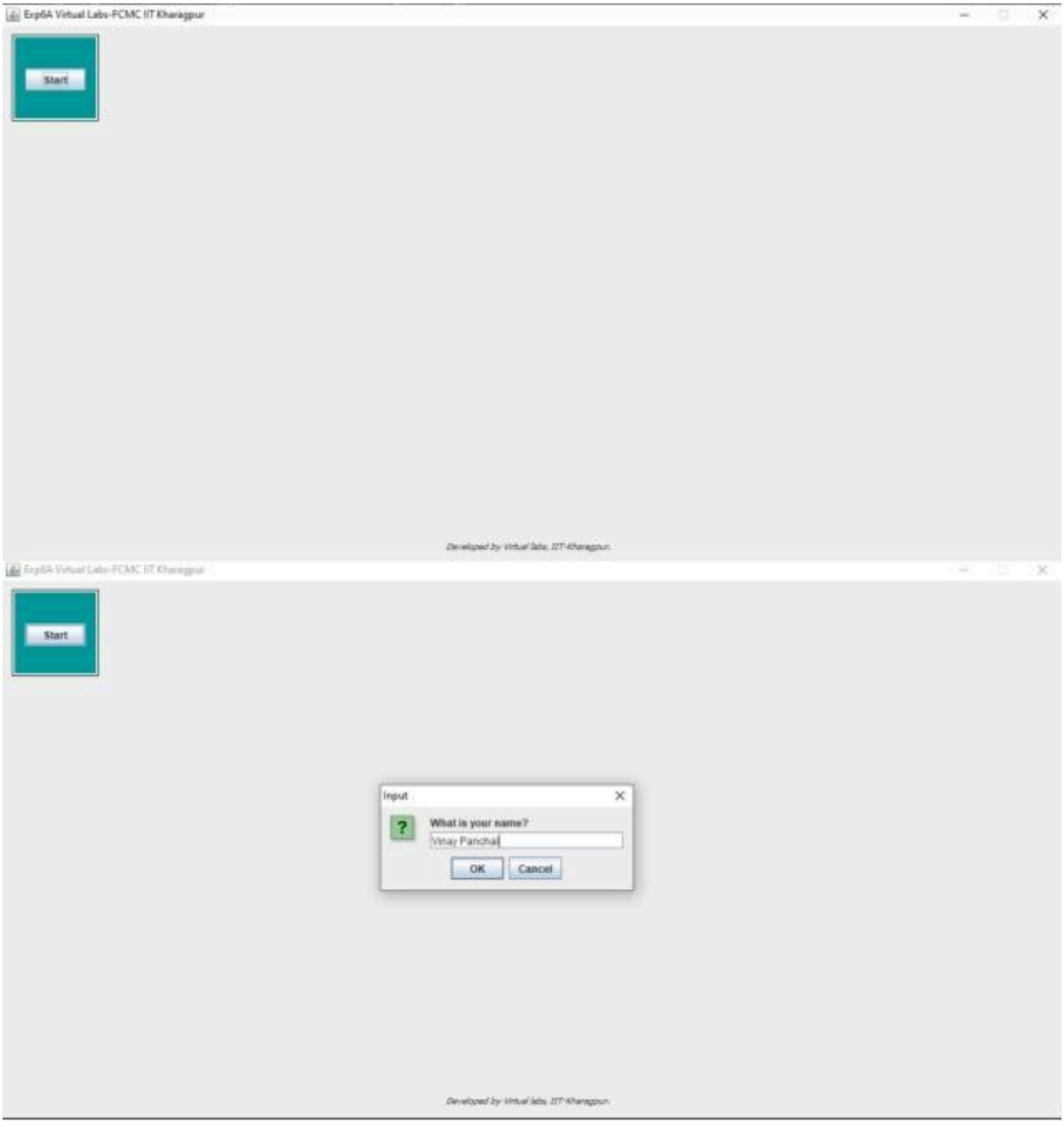
To nd the nearest co-channel neighbours of a particular cell,

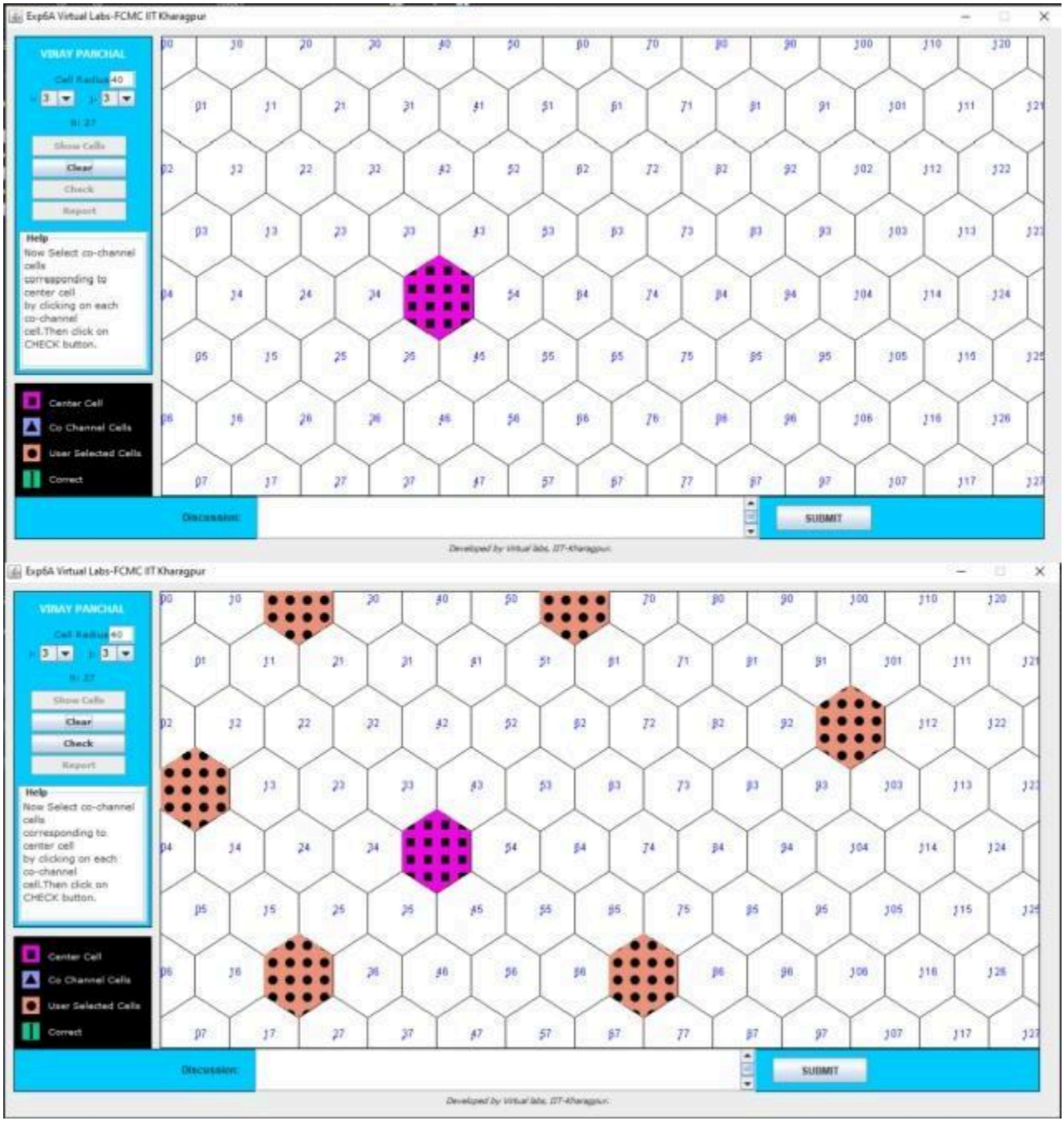
a. move i cells along any chain of hexagons then,

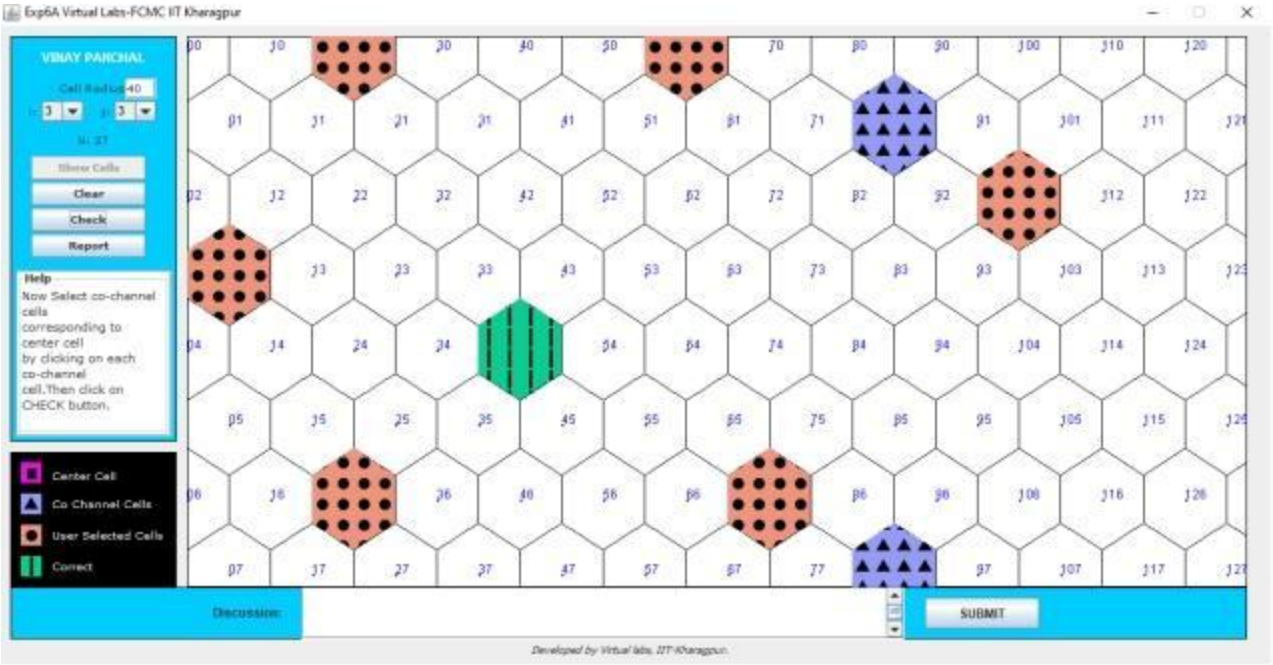
b. turn 60 degrees counter-clockwise and move j cells.

https://www.youtube.com/watch?v=6axnMhPKyhU

Output:


Exp6A Virtual Labs-FCMC IIT Kharagpur
Start
Developed by Virtual labs, IIT-Kharagpur
Exp6A Virtual Labs-FCMC IIT Kharagpur
Start
Input
What is your name?
Vinay Pancha
OK
Cancel
Developed by Virtual labs, IIT-Kharagpur


Exp6A Virtual Labs-FCMC IIT Kharagpur
VINAY PANCHAL
Cell Radius 40
Show Cells
Clear
Check
Report
Help
Now Select co-channel cells corresponding to center cell by clicking on each co-channel cell.Then click on CHECK button.
Center Cell
Co Channel Cells
User Selected Cells
Correct
Discussion
SUBMIT
Developed by Virtual labs, IIT-Kharagpur
Exp6A Virtual Labs-FCMC IIT Kharagpur
VINAY PANCHAL
Cell Radius 40
Show Cells
Clear
Check
Report
Help
Now Select co-channel cells corresponding to center cell by clicking on each co-channel cell.Then click on CHECK button.
Center Cell
Co Channel Cells
User Selected Cells
Correct
Discussion
SUBMIT
Developed by Virtual labs, IIT-Kharagpur



Conclusion:

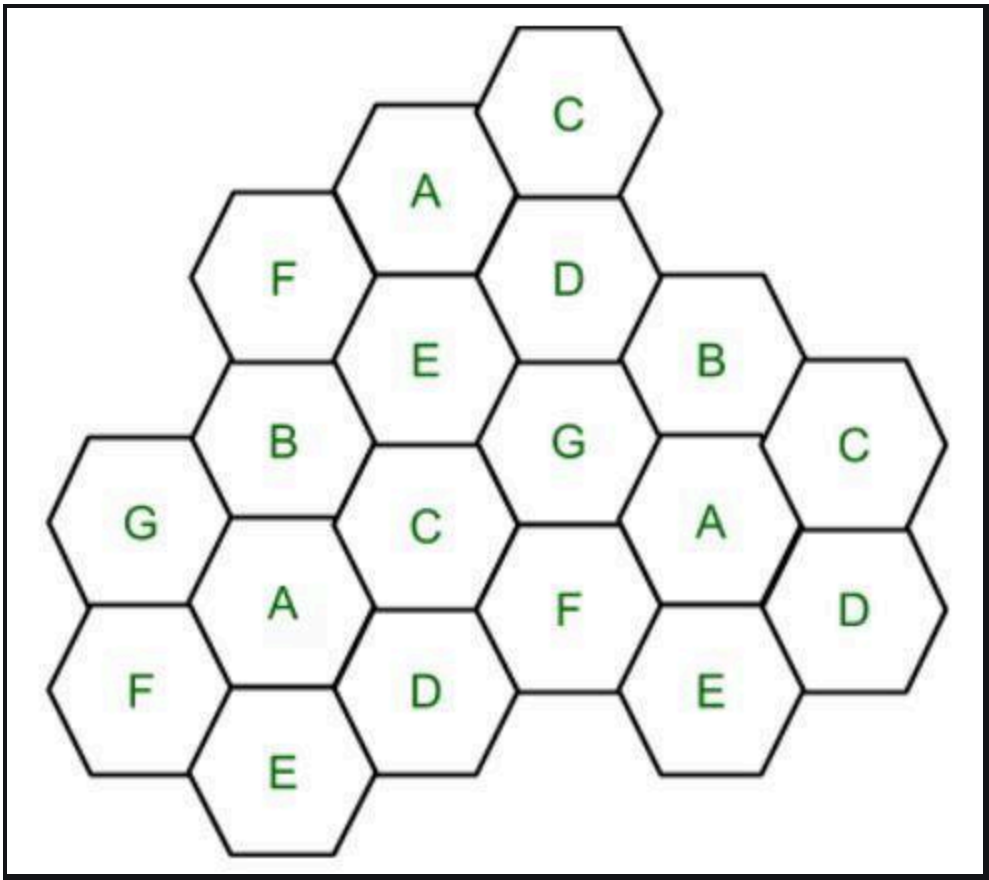
Thus, we have performed the frequency reuse experiment wherein we select a hexagon with 1particular frequency and then select other hexagons where the after which frequency can be used using above formula, we performed the experiment properly.

https://www.geeksforgeeks.org/frequency-reuse/

Frequency Reuse is the scheme in which allocation and reuse of channels throughout a coverage region is done. Each cellular base station is allocated a group of radio channels or Frequency sub-bands to be used within a small geographic area known as a cell. The shape of the cell is Hexagonal. The process of selecting and allocating the frequency sub-bands for all of the cellular base station within a system is called **Frequency reuse** or **Frequency Planning**.

Salient features of using Frequency Reuse:

- Frequency reuse improve the spectral efficiency and signal Quality (QoS).
- Frequency reuse classical scheme proposed for GSM systems offers a protection against interference.
- The number of times a frequency can be reused is depend on the tolerance capacity of the radio channel from the nearby transmitter that is using the same frequencies.
- In Frequency Reuse scheme, total bandwidth is divided into different sub-bands that are used by cells.
- Frequency reuse scheme allow WiMax system operators to reuse the same frequencies at different cell sites.



Cell with the same letter uses the same set of channels group or frequencies sub-band.

To find the total number of channel allocated to a cell:

S =

To find the total number of channel allocated to a cell:

S = Total number of duplex channels available to use
k = Channels allocated to each cell (k<S)
N = Total number of cells or Cluster Size

Then Total number of channels (S) will be, S=KN, Frequency Factor=1/N

In the above diagram cluster size is 7 (A,B,C,D,E,F,G) thus frequency reuse factor is 1/7.

N is the number of cells which collectively use the complete set of available frequencies is called a Cluster. The value of N is calculated by the following formula:

Hence, possible values of N are 1,3,4,7,9,12,13,16,19 and so on.

If a Cluster is replicated or repeated M times within the cellular system, then Capacity, C, will be,

In Frequency reuse there are several cells that use the same set of frequencies. These cells are called Co-Channel Cells. These Co-Channel cells results in interference. So to avoid the Interference cells that use the same set of channels or frequencies are separated from one another by a larger distance. The distance between any two Co-Channels can be calculated by the following formula:

D=R* (3*N) op power ½

Where,
R = Radius of a cell
N = Number of cells in a given cluster

Below is the Python code for visualizing the Frequency Reuse concept

```
#!/usr/bin/python

from math import *

# import everything from Tkinter module
from tkinter import *

# Base class for Hexagon shape
class Hexagon(object):
    def __init__(self, parent, x, y, length, color, tags):
        self.parent = parent
        self.x = x
        self.y = y
        self.length = length
        self.color = color
        self.size = None
        self.tags = tags
        self.draw_hex()

    # draw one hexagon
    def draw_hex(self):
        start_x = self.x
        start_y = self.y
        angle = 60
        coords = []
        for i in range(6):
            end_x = start_x + self.length * cos(radians(angle * i))
```

```
            end_y = start_y + self.length * sin(radians(angle * i))
            coords.append([start_x, start_y])
            start_x = end_x
            start_y = end_y
        self.parent.create_polygon(coords[0][0],
                                   coords[0][1],
                                   coords[1][0],
                                   coords[1][1],
                                   coords[2][0],
                                   coords[2][1],
                                   coords[3][0],
                                   coords[3][1],
                                   coords[4][0],
                                   coords[4][1],
                                   coords[5][0],
                                   coords[5][1],
                                   fill=self.color,
                                   outline="black",
                                   tags=self.tags)

# class holds frequency reuse logic and related methods
class FrequencyReuse(Tk):
    CANVAS_WIDTH = 800
    CANVAS_HEIGHT = 650
    TOP_LEFT = (20, 20)
    BOTTOM_LEFT = (790, 560)
    TOP_RIGHT = (780, 20)
    BOTTOM_RIGHT = (780, 560)

    def __init__(self, cluster_size, columns=16, rows=10, edge_len=30):
        Tk.__init__(self)
        self.textbox = None
        self.curr_angle = 330
        self.first_click = True
        self.reset = False
        self.edge_len = edge_len
        self.cluster_size = cluster_size
        self.reuse_list = []
        self.all_selected = False
        self.curr_count = 0
        self.hexagons = []
        self.co_cell_endp = []
        self.reuse_xy = []
        self.canvas = Canvas(self,
                             width=self.CANVAS_WIDTH,
```

```
                            height=self.CANVAS_HEIGHT,
                            bg="#4dd0e1")
        self.canvas.bind("<Button-1>", self.call_back)
        self.canvas.focus_set()
        self.canvas.bind('<Shift-R>', self.resets)
        self.canvas.pack()
        self.title("Frequency reuse and co-channel selection")
        self.create_grid(16, 10)
        self.create_textbox()
        self.cluster_reuse_calc()

    # show lines joining all co-channel cells
    def show_lines(self):
        # center(x,y) of first hexagon
        approx_center = self.co_cell_endp[0]
        self.line_ids = []
        for k in range(1, len(self.co_cell_endp)):

            end_xx = (self.co_cell_endp[k])[0]
            end_yy = (self.co_cell_endp[k])[1]

            # move i^th steps
            l_id = self.canvas.create_line(approx_center[0],
approx_center[1],
                                           end_xx, end_yy)
            if j == 0:
                self.line_ids.append(l_id)
                dist = 0
            elif i >= j and j != 0:
                self.line_ids.append(l_id)
                dist = j
                # rotate counter-clockwise and move j^th step
                l_id = self.canvas.create_line(
                    end_xx, end_yy, end_xx + self.center_dist * dist *
                    cos(radians(self.curr_angle - 60)),
                    end_yy + self.center_dist * dist *
                    sin(radians(self.curr_angle - 60)))
                self.line_ids.append(l_id)
            self.curr_angle -= 60

    def create_textbox(self):
        txt = Text(self.canvas,
                   width=80,
                   height=1,
                   font=("Helvatica", 12),
```

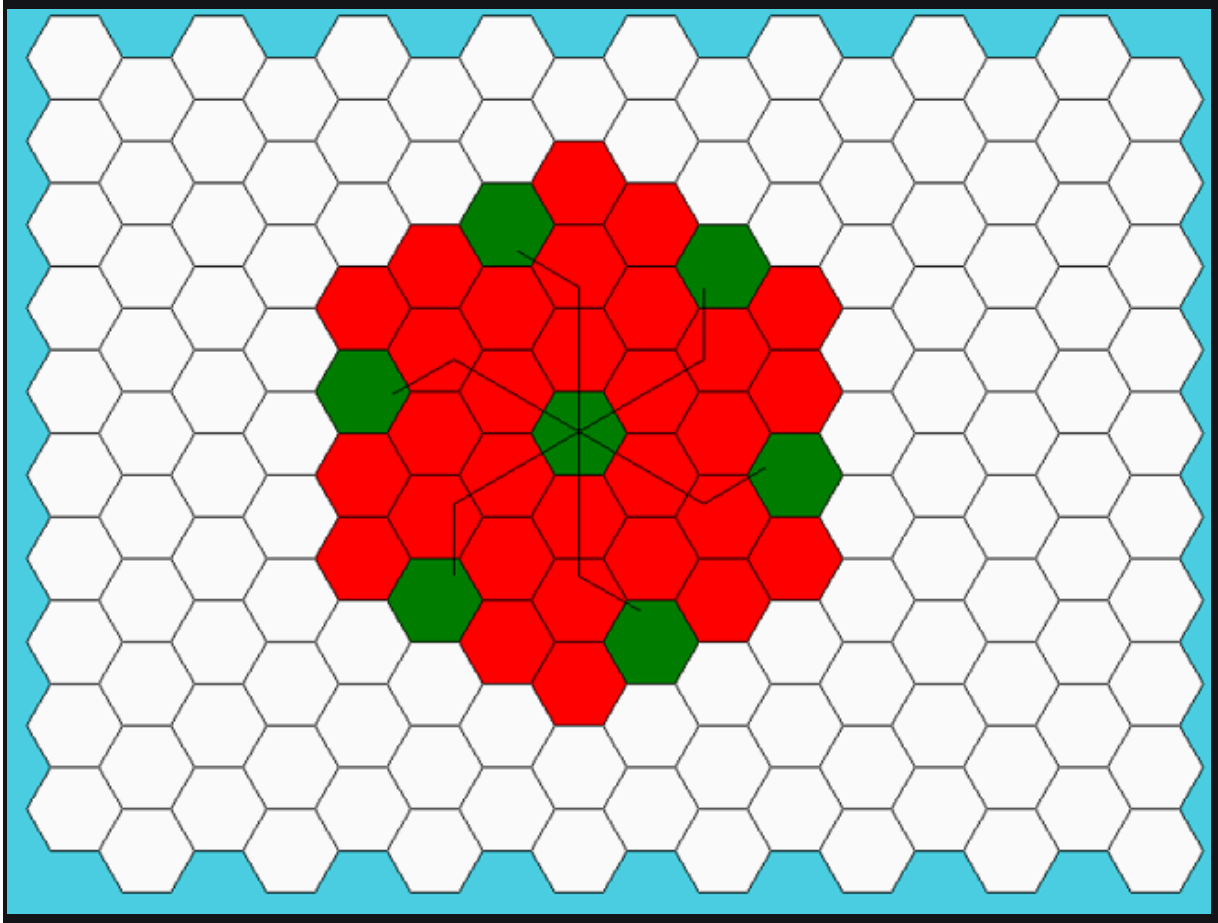
```
                padx=10,
                pady=10)
        txt.tag_configure("center", justify="center")
        txt.insert("1.0", "Select a Hexagon")
        txt.tag_add("center", "1.0", "end")
        self.canvas.create_window((0, 600), anchor='w', window=txt)
        txt.config(state=DISABLED)
        self.textbox = txt

    def resets(self, event):
        if event.char == 'R':
            self.reset_grid()

    # clear hexagonal grid for new i/p
    def reset_grid(self, button_reset=False):
        self.first_click = True
        self.curr_angle = 330
        self.curr_count = 0
        self.co_cell_endp = []
        self.reuse_list = []
        for i in self.hexagons:
            self.canvas.itemconfigure(i.tags, fill=i.color)

        try:
            self.line_ids
        except AttributeError:
            pass
        else:
            for i in self.line_ids:
                self.canvas.after(0, self.canvas.delete, i)
            self.line_ids = []

        if button_reset:
            self.write_text("Select a Hexagon")

    # create a grid of Hexagons
    def create_grid(self, cols, rows):
        size = self.edge_len
        for c in range(cols):
            if c % 2 == 0:
                offset = 0
            else:
                offset = size * sqrt(3) / 2
            for r in range(rows):
```

```
            x = c * (self.edge_len * 1.5) + 50
            y = (r * (self.edge_len * sqrt(3))) + offset + 15
            hx = Hexagon(self.canvas, x, y, self.edge_len, "#fafafa",
                         "{},{}".format(r, c))
            self.hexagons.append(hx)

    # calculate reuse distance, center distance and radius of the hexagon
    def cluster_reuse_calc(self):
        self.hex_radius = sqrt(3) / 2 * self.edge_len
        self.center_dist = sqrt(3) * self.hex_radius
        self.reuse_dist = self.hex_radius * sqrt(3 * self.cluster_size)

    def write_text(self, text):
        self.textbox.config(state=NORMAL)
        self.textbox.delete('1.0', END)
        self.textbox.insert('1.0', text, "center")
        self.textbox.config(state=DISABLED)

    #check if the co-channels are within visible canvas
    def is_within_bound(self, coords):
        if self.TOP_LEFT[0] < coords[0] < self.BOTTOM_RIGHT[0] \
        and self.TOP_RIGHT[1] < coords[1] < self.BOTTOM_RIGHT[1]:
            return True
        return False

    #gets called when user selects a hexagon
    #This function applies frequency reuse logic in order to
    #figure out the positions of the co-channels
    def call_back(self, evt):

        selected_hex_id = self.canvas.find_closest(evt.x, evt.y)[0]
        hexagon = self.hexagons[int(selected_hex_id - 1)]
        s_x, s_y = hexagon.x, hexagon.y
        approx_center = (s_x + 15, s_y + 25)

        if self.first_click:
            self.first_click = False
            self.write_text(
                """Now, select another hexagon such
                   that it should be a co-cell of
                   the original hexagon."""
            )
            self.co_cell_endp.append(approx_center)
            self.canvas.itemconfigure(hexagon.tags, fill="green")
```

```
            for _ in range(6):

                end_xx = approx_center[0] + self.center_dist * i * cos(
                    radians(self.curr_angle))
                end_yy = approx_center[1] + self.center_dist * i * sin(
                    radians(self.curr_angle))

                reuse_x = end_xx + (self.center_dist * j) * cos(
                    radians(self.curr_angle - 60))
                reuse_y = end_yy + (self.center_dist * j) * sin(
                    radians(self.curr_angle - 60))

                if not self.is_within_bound((reuse_x, reuse_y)):
                    self.write_text(
                        """co-cells are exceeding canvas boundary.
                           Select cell in the center"""
                    )
                    self.reset_grid()
                    break

                if j == 0:
                    self.reuse_list.append(
                        self.canvas.find_closest(end_xx, end_yy)[0])
                elif i >= j and j != 0:
                    self.reuse_list.append(
                        self.canvas.find_closest(reuse_x, reuse_y)[0])

                self.co_cell_endp.append((end_xx, end_yy))
                self.curr_angle -= 60

        else:
            curr = self.canvas.find_closest(s_x, s_y)[0]
            if curr in self.reuse_list:
                self.canvas.itemconfigure(hexagon.tags, fill="green")
                self.write_text("Correct! Cell {} is a co-cell.".format(
                    hexagon.tags))
                if self.curr_count == len(self.reuse_list) - 1:
                    self.write_text("Great! Press Shift-R to restart")
                    self.show_lines()
                self.curr_count += 1

            else:
                self.write_text("Incorrect! Cell {} is not a
co-cell.".format(
```

```
                    hexagon.tags))
                self.canvas.itemconfigure(hexagon.tags, fill="red")

if __name__ == '__main__':
    print(
        """Enter i & j values. common (i,j) values are:
           (1,0), (1,1), (2,0), (2,1), (3,0), (2,2)"""
    )
    i = int(input("Enter i: "))
    j = int(input("Enter j: "))
    if i == 0 and j == 0:
        raise ValueError("i & j both cannot be zero")
    elif j > i:
        raise ValueError("value of j cannot be greater than i")
    else:
        N = (i**2 + i * j + j**2)
        print("N is {}".format(N))
    freqreuse = FrequencyReuse(cluster_size=N)
    freqreuse.mainloop()
```

OUTPUT:



Mobile Computing Experiment - 3

Q.) Frequency Reuse

→ Frequency reuse is the scheme in which allocation and reuse of channels through a coverage of region is done.

Q.) How to make a cluster.

→ Cells with the same letter use the same set of frequency called reusing cells.
N cell which collectively use the available frequency is known as cluster.

$S = K \cdot N$

$N \Rightarrow$ No. of cells.
$K \Rightarrow$ channel allocated to each cell.
$S \Rightarrow$ No. of duplex channels.

Q.) What is D and R?

→ D is the distance between any two co-channels
$R \Rightarrow$ Radius of the cells.
$N \Rightarrow$ No. of cells.
$D = R * (3 * N)^{1/2}$

Q.) Result Analysis

$S = K \cdot N$

$N = 7$

$K =$ No. of duplex channels $= 95$

$S = 95 \times 7 = 665$

Experiment No.: 4

Aim: Illustration of Hidden Terminal Problem

(NS-2) **Theory:**

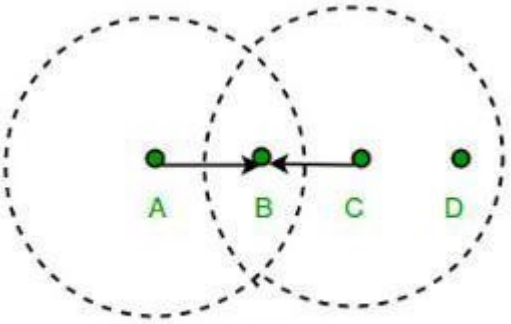
A wireless network with lack of centralized control entity, sharing of wireless bandwidth among network access nodes i.e. medium access control (MAC) nodes must be organized in decentralized manner. The hidden terminal problem occurs when a terminal is visible from a wireless access point (APs), but not from other nodes communicating with that AP. This situation leads the difficulties in medium access control sublayer over wireless networking.

In a formal way hidden terminal are nodes in a wireless network that are out of range of other node or a collection of nodes. Consider a wireless networking, each node at the far edge of the access point's range, which is known as A, can see the access point, but it is unlikely that the same node can see a node on the opposite end of the access point's range, C. These nodes are known as hidden. The problem is when nodes A and C start to send packets simultaneously to the access point B. Because the nodes A and C are out of range of each other and so cannot detect a collision while transmitting, Carrier sense multiple access with collision detection (CSMA/CD) does not work, and collisions occur, which then corrupt the data received by the access point. To overcome the hidden node problem, RTS/CTS handshaking (IEEE 802.11 RTS/CTS) is implemented in conjunction with the Carrier sense multiple accesses with collision avoidance (CSMA/CA) scheme. The same problem exists in a MANET.

The transmission range of access point A reaches at B, but not at access point C, similarly transmission range of access point C reaches B, but not at A. These nodes are known as hidden terminals. The problem occurs when nodes A and C start to send data packets simultaneously to the access point B. Because the access points A and C are out of range of each other and resultant they cannot detect a collision while transmitting, Carrier sense multiple access with collision detection (CSMA/CD) does not work, and collisions occur, which then corrupt the data received by the access point B due to the hidden terminal problem.

The hidden terminal analogy is described as follows:

- Terminal A sends data to B, terminal C cannot hear A
- Terminal C wants to send data to B, terminal C senses a "free" medium (CS fails) and starts transmitting
- Collision at B occurs, A cannot detect this collision (CD fails) and continues with its transmission to B
- Terminal A is "hidden" from C and vice versa.



Hidden Node Problem

The solution of hidden terminal problem is as follows.

When A wants to send a packet to B, A first sends a Request-to-send (RTS) to B. On receiving RTS, B responds by sending Clear-to-Send (CTS).

When C overhears a CTS, it keeps quiet for the duration of the transfer. Transfer duration is included in both RTS and CTS.

RTS and CTS are short frames, reduces collision chance.

Experiment 4

CODE:

Exp4.tcl

```
# Define options
set val(chan)           Channel/WirelessChannel     ;# channel type
set val(prop)           Propagation/FreeSpace   ;# radio-propagation model
set val(netif)          Phy/WirelessPhy       ;# network interface type
set val(mac)            Mac/802_11            ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue             ;# interface queue type
set val(ll)       LL                    ;# link layer type
set val(ant)            Antenna/OmniAntenna         ;# antenna model
set val(ifqlen)         10000                 ;# max packet in ifq
set val(nn)             5               ;# number of mobilenodes
set val(rp)             DSR                   ;# routing protocol
set val(x)              600             ;# X dimension of topography
set val(y)              600             ;# Y dimension of topography
set val(stop)   100             ;# time of simulation end
set val(R)      300
set opt(tr)     out.tr
set ns          [new Simulator]
set tracefd  [open $opt(tr) w]
set windowVsTime2 [open win.tr w]
set namtrace   [open simwrls.nam w]
  Mac/802_11 set dataRate_          1.2e6
Mac/802_11 set RTSThreshold_      100
$ns trace-all $tracefd
#$ns use-newtrace
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo        [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

#
#  Create nn mobilenodes [$val(nn)] and attach them to the channel.
#

# configure the nodes

        $ns node-config -adhocRouting $val(rp) \
        -llType $val(ll) \
        -macType $val(mac) \
        -ifqType $val(ifq) \
        -ifqLen $val(ifqlen) \
        -antType $val(ant) \
        -propType $val(prop) \
```

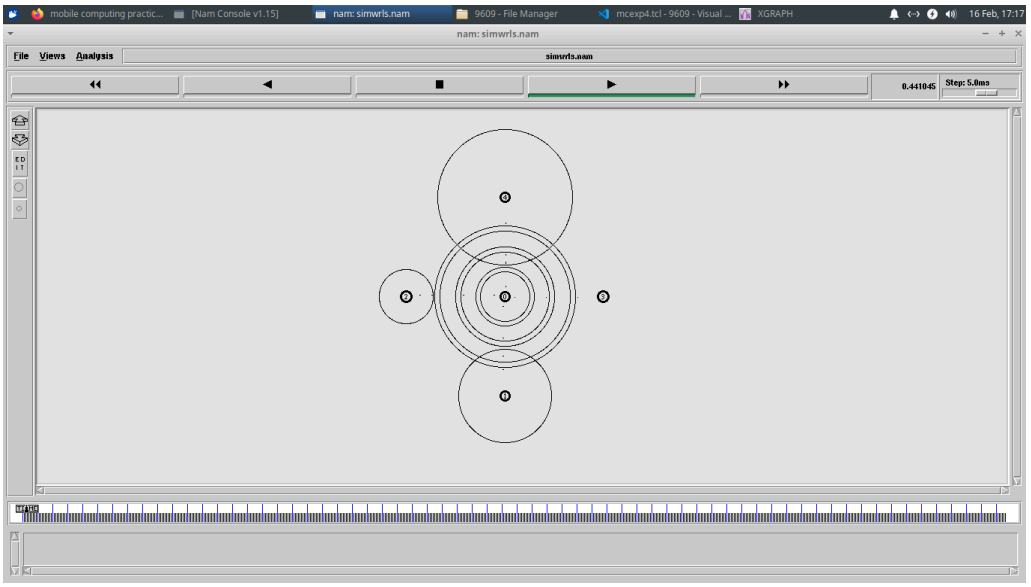
```
        -phyType $val(netif) \
        -channelType $val(chan) \
        -topoInstance $topo \
        -agentTrace ON \
        -routerTrace ON \
        -macTrace ON \
        -movementTrace ON

Phy/WirelessPhy set CSThresh 30.5e-10
        for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
        }
$node_(0) set X_ $val(R)
$node_(0) set Y_ $val(R)
$node_(0) set Z_ 0
$node_(1) set X_ $val(R)
$node_(1) set Y_ 0
$node_(1) set Z_ 0
$node_(2) set X_ 0
$node_(2) set Y_ $val(R)
$node_(2) set Z_ 0
$node_(3) set X_ [expr $val(R) *2]
$node_(3) set Y_ $val(R)
$node_(3) set Z_ 0
$node_(4) set X_ $val(R)
$node_(4) set Y_ [expr $val(R) *2]
$node_(4) set Z_ 0
 for {set i 0} {$i<$val(nn)} {incr i} {
  $ns initial_node_pos $node_($i) 30
 }
# Generation of movements
 $ns at 0 "$node_(1) setdest $val(R) $val(R) 3.0"
$ns at 0 "$node_(2) setdest $val(R) $val(R) 3.0"
$ns at 0 "$node_(3) setdest $val(R) $val(R) 3.0"
$ns at 0 "$node_(4) setdest $val(R) $val(R) 3.0"
# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP/Newreno]
#$tcp set class_ 2
set tcp [new Agent/UDP]
$tcp set class_ 2
set sink [new Agent/Null]
$ns attach-agent $node_(1) $tcp
$ns attach-agent $node_(0) $sink
$ns connect $tcp $sink
set ftp [new Application/Traffic/CBR]
$ftp attach-agent $tcp
$ns at 0.0 "$ftp start"
# ##################################################
# For coloring but doesnot work
# ##################################################
$tcp set fid_ 1
```

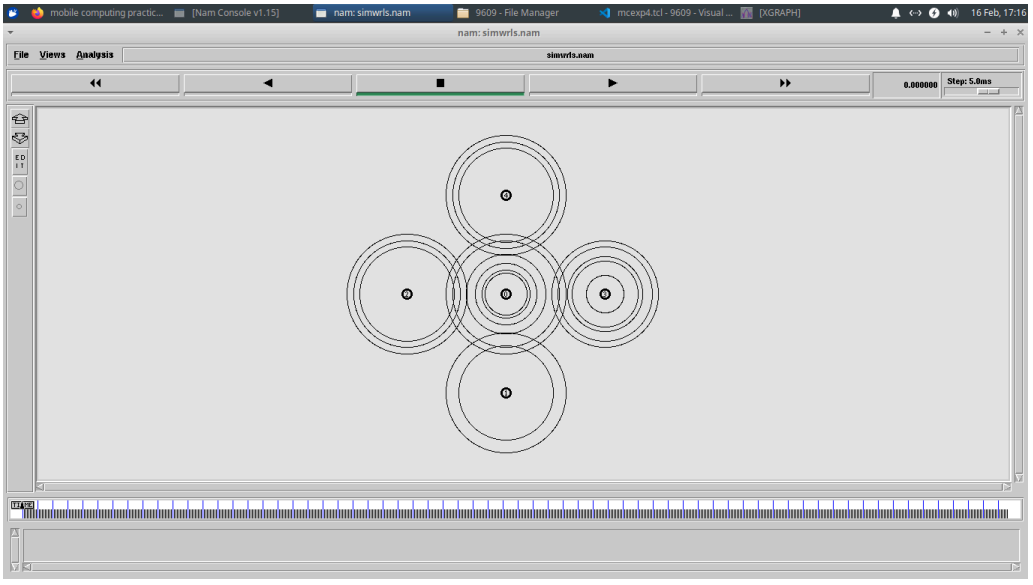
```
$ns color 1 blue
#//////////////////////////////////////////////
set tcp [new Agent/UDP]
$tcp set class_ 2
set sink [new Agent/Null]
$ns attach-agent $node_(2) $tcp
$ns attach-agent $node_(0) $sink
$ns connect $tcp $sink
set ftp [new Application/Traffic/CBR]
$ftp attach-agent $tcp
$ns at  0.0 "$ftp start"
set tcp [new Agent/UDP]
$tcp set class_ 2
set sink [new Agent/Null]
$ns attach-agent $node_(3) $tcp
$ns attach-agent $node_(0) $sink
$ns connect $tcp $sink
set ftp [new Application/Traffic/CBR]
$ftp attach-agent $tcp
$ns at  0.0 "$ftp start"
set tcp [new Agent/UDP]
$tcp set class_ 2
set sink [new Agent/Null]
$ns attach-agent $node_(4) $tcp
$ns attach-agent $node_(0) $sink
$ns connect $tcp $sink
set ftp [new Application/Traffic/CBR]
$ftp attach-agent $tcp
$ns at  0.0 "$ftp start"
# Telling nodes when the simulation ends
#for {set i 0} {$i < $val(nn) } { incr i } {
 #   $ns at $val(stop) "$node_($i) reset";
#}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at $val(stop) "puts \"end simulation\" ; $ns halt"
proc stop {} {
 exec awk -f fil.awk out.tr > out.xgr
exec xgraph out.xgr &

        global ns tracefd namtrace
        $ns flush-trace
        close $tracefd
   close $namtrace
        exec nam simwrls.nam &
}

$ns run
```

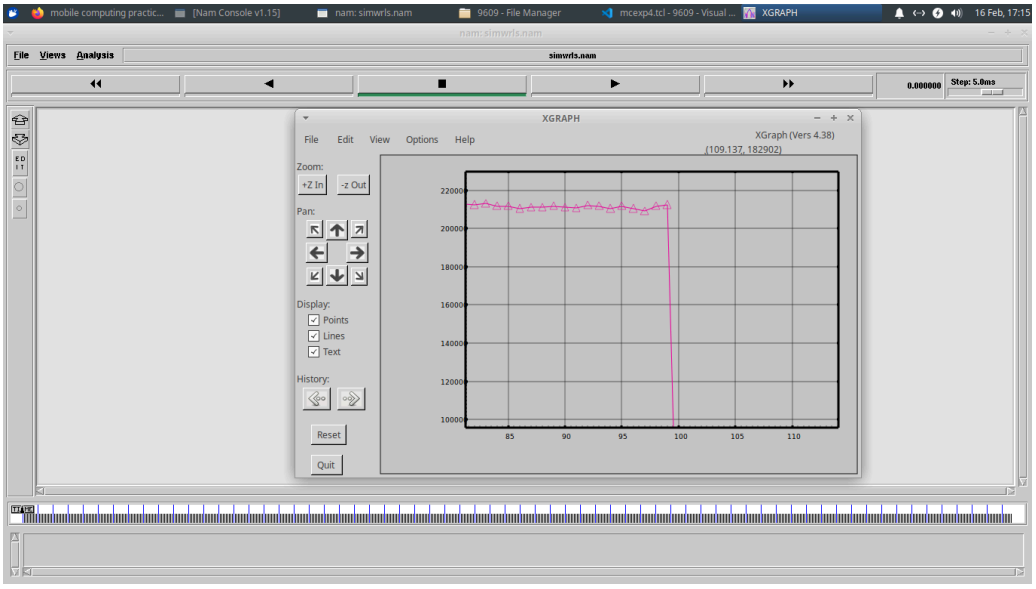
Fil.awk

```
BEGIN{
        sim_end = 200;
        i=0;
        while (i<=sim_end) {sec[i]=0; i+=1;};
}

{
        if ($1=="r" && $7=="cbr"&& $3=="_0_") {
        sec[int($2)]+=$8;
        };
}

END{
        i=0;
        while (i<=sim_end) {print i " " sec[i]; i+=1;};
}
```

OUTPUT:


nam: simwrls.nam
File Views Analysis
simwrls.nam
Step: 5.0ms


nam: simwrls.nam
File Views Analysis
simwrls.nam
Step: 5.0ms


nam: simwrls.nam
File Views Analysis
simwrls.nam
Step: 5.0ms
XGRAPH
File Edit View Options Help
XGraph (Vers 4.38)
(109.137, 182902)
Zoom:
+Z In
-z Out
Pan:
Display:
Points
Lines
Text
History:
Reset
Quit
220000
200000
180000
160000
140000
120000
100000
85
90
95
100
105
110

POSTLAB:

1. **Explain in brief what is the hidden terminal problem**
2. **How does HTP affect performance of wireless network**
3. **What is solution to hidden terminal problem?**
4. **How does hidden terminal problem differ from exposed terminal problem**

Explain in brief what is the hidden terminal problem?

The hidden terminal problem refers to a scenario in wireless networks where nodes are within range of a central node but out of range of each other, causing them to be unaware of each other's transmissions. This situation can lead to collisions when multiple nodes attempt to transmit data simultaneously, resulting in packet loss and degraded network performance.

How does HTP affect the performance of a wireless network?

HTP negatively impacts the performance of wireless networks by increasing the likelihood of collisions and packet loss. When nodes transmit data without being able to detect each other's transmissions, collisions occur at the central node, leading to retransmissions and reduced throughput. This results in decreased network efficiency, reliability, and increased latency.

What is the solution to the hidden terminal problem?

One solution to the hidden terminal problem is the implementation of Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocols. These protocols enable nodes to sense the wireless medium before transmitting data, allowing them to avoid collisions by waiting for the medium to be clear. Additionally, techniques such as Request-to-Send (RTS) and Clear-to-Send (CTS) can be employed to reserve the medium for transmission, reducing the occurrence of collisions.

How does the hidden terminal problem differ from the exposed terminal problem?

The hidden terminal problem differs from the exposed terminal problem in that the former occurs when nodes cannot detect each other's transmissions, leading to collisions when they simultaneously transmit data. On the other hand, the exposed terminal problem arises when a node refrains from transmitting data incorrectly believing that the medium is busy due to transmissions from other nodes, even though these transmissions would not interfere with its own transmission. This situation results in unnecessary delays in data transmission.

Experiment No.: 5

Aim: To implement a basic function of Code Division Multiple Access (CDMA) to test the orthogonality and autocorrelation of a code to be used for CDMA operation. Write an application based on the above concept.

Theory:

Code-division multiple access (CDMA) is a channel access method used by various radio communication technologies. CDMA is an example of multiple access, where several transmitters can send information simultaneously over a single communication channel. This allows several users to share a band of frequencies (see bandwidth). To permit this without undue interference between the users, CDMA employs spread spectrum technology and a special coding scheme (where each transmitter is assigned a code).

CDMA is used as the access method in many mobile phone standards. IS-95, also called "cdmaOne", and its 3G evolution CDMA2000, are often simply referred to as "CDMA", but UMTS, the 3G standard used by GSM carriers, also uses "wideband CDMA", or W-CDMA, as well as TD-CDMA and TD-SCDMA, as its radio technologies.

The intended 4G successor to CDMA2000 was UMB (Ultra Mobile Broadband); however, in November 2008, Qualcomm announced it was ending development of the technology, favoring LTE instead

CDMA Orthogonality:

Techniques generally used are direct sequence spread spectrum modulation (DS-CDMA), frequency hopping or mixed CDMA detection (JDCDMA). Here, a signal is generated which extends over a wide bandwidth. A code called spreading code is used to perform this action. Using a group of codes, which are orthogonal to each other, it is possible to select a signal with a given code in the presence of many other signals with different orthogonal codes.

CDMA Autocorrelation:

Autocorrelation of the sequence, it determines the ability to synchronize and lock the spreading code for the received signal.

Code:

```
import numpy as np
c1=[1,1,1,1]
c2=[1,-1,1,-1]
c3=[1,1,-1,-1]
c4=[1,-1,-1,1]
rc=[]

print("Enter the data bits :")

d1=int(input("Enter D1 :"))
d2=int(input("Enter D2 :"))
d3=int(input("Enter D3 :"))
d4=int(input("Enter D4 :"))
r1=np.multiply(c1,d1)
r2=np.multiply(c2,d2)
r3=np.multiply(c3,d3)
r4=np.multiply(c4,d4)
resultant_channel=r1+r2+r3+r4;
print("Resultant Channel",resultant_channel)
Channel=int(input("Enter  the station to listen for C1=1 ,C2=2, C3=3 C4=4 :
"))

if Channel==1:
 rc=c1
elif Channel==2:
 rc=c2
elif Channel==3:
 rc=c3
elif Channel==4:
 rc=c4
inner_product = np.multiply(resultant_channel,rc)

print("Inner Product",inner_product)
res1=sum(inner_product)

data = res1/len(inner_product)
print("Data bit that was sent",data)
```

Output:

```
Enter the data bits :
Enter D1 :23
Enter D2 :5
Enter D3 :456
Enter D4 :56
Resultant Channel [ 540  418 -484 -382]
Enter  the station to listen for C1=1 ,C2=2, C3=3 C4=4 : 1
Inner Product [ 540  418 -484 -382]
Data bit that was sent 23.0
```

Conclusion:

Thus, we have studied the CDMA code to test autocorrelation and orthogonality of codes and executed the same using the java code as above and got proper output for it.

POST LAB

1. **What is CDMA and how does it differ from other multiple access techniques?**

CDMA, or Code Division Multiple Access, is a telecommunications technique enabling multiple users to share the same frequency band concurrently. Unlike FDMA and TDMA, which allocate exclusive frequency channels or time slots to users, CDMA assigns unique codes to each user for data transmission. These codes are designed to be orthogonal, minimizing interference between users. In contrast to OFDMA, which divides the frequency spectrum into subcarriers, CDMA does not require such division, allowing for efficient spectrum utilization. One advantage of CDMA is its increased capacity, as multiple users can transmit simultaneously on the same frequency band. Moreover, CDMA offers improved spectral efficiency compared to other techniques. However, CDMA necessitates complex signal processing methods and careful interference management to ensure efficient operation. Overall, CDMA provides enhanced capacity, spectral efficiency, and security in telecommunications systems.

2. **How does spread spectrum technology contribute to CDMA?**

Spread spectrum technology plays a crucial role in CDMA by enabling multiple users to share the same frequency band simultaneously while minimizing interference. In CDMA, each user's data is spread over a wider bandwidth using a unique spreading code. This spreading process "spreads" the signal, making it appear as noise to other users not using the same spreading code.

Spread spectrum technology contributes to CDMA in several ways:

1. Increased Capacity : By spreading each user's signal over a wider bandwidth, CDMA can accommodate more users within the same frequency band compared to other multiple access techniques. This leads to increased capacity in the system.
2. Enhanced Security : The spreading process in CDMA makes it resistant to interference and jamming. Since the signal appears as noise to unauthorized users, it is difficult for them to intercept or disrupt communication, enhancing the security of the system.
3. Improved Robustness : Spread spectrum technology helps CDMA systems to tolerate multipath interference and fading effects more effectively. The spread signal can be reconstructed even if parts of it are lost or corrupted during transmission.
4. Flexible Allocation : CDMA allows for flexible allocation of bandwidth among users since each user is assigned a unique spreading code. This enables dynamic resource allocation based on users' varying needs and traffic demands.

Overall, spread spectrum technology is fundamental to the operation of CDMA, enabling efficient and secure communication among multiple users within the same frequency band.

3. **What are the advantages of CDMA over other multiple access techniques like FDMA and TDMA?**

CDMA (Code Division Multiple Access) offers several advantages over other multiple access techniques such as FDMA (Frequency Division Multiple Access) and TDMA (Time Division Multiple Access). Firstly, CDMA provides increased capacity by allowing multiple users to share the same frequency band simultaneously without the need for frequency or time slot allocation. This leads to more efficient spectrum utilization. Secondly, CDMA offers enhanced security as each user's data is

encoded with a unique code, making it difficult for unauthorized users to intercept or disrupt communication. Additionally, CDMA exhibits better resistance to interference and fading, resulting in improved call quality and reliability. Furthermore, CDMA allows for seamless handoffs between cells, leading to smoother mobility management. CDMA also supports variable data rates, catering to diverse user requirements. Moreover, CDMA systems require less complex infrastructure compared to FDMA and TDMA, leading to cost savings in deployment and maintenance. Additionally, CDMA is well-suited for multimedia applications due to its robustness and flexibility. Overall, CDMA stands out for its capacity, security, reliability, flexibility, and cost-effectiveness compared to other multiple access techniques.

4. **Can you explain the concept of "orthogonality" in CDMA?**

In CDMA (Code Division Multiple Access), "orthogonality" refers to the property where the spreading codes used by different users are mathematically orthogonal to each other. This means that when overlaid on top of each other, the codes do not interfere with one another. Orthogonality ensures that each user's signal can be accurately distinguished from others, even when transmitted simultaneously over the same frequency band. This property is essential for enabling multiple users to share the same spectrum without causing mutual interference. The orthogonal nature of spreading codes allows CDMA systems to achieve high spectral efficiency by accommodating multiple users within the same bandwidth. It also contributes to increased capacity and improved system performance. The design of orthogonal spreading codes involves carefully selecting sequences that exhibit minimal cross-correlation with each other. Maintaining orthogonality is crucial for the successful operation of CDMA networks, ensuring reliable communication and efficient spectrum utilization.

5. **What are the key challenges in implementing CDMA systems?**

Implementing CDMA systems comes with several key challenges. Firstly, managing interference is critical due to the shared spectrum nature of CDMA, requiring sophisticated interference mitigation techniques. Secondly, maintaining orthogonality among spreading codes is challenging as the number of users increases, necessitating careful code design and synchronization. Thirdly, achieving optimal power control is crucial to mitigate near-far effects and maintain system fairness. Additionally, handling multipath propagation and fading poses challenges in maintaining signal quality and reliability. Moreover, accommodating varying data rates and Quality of Service (QoS) requirements for different users adds complexity to system design and resource allocation. Ensuring secure communication in CDMA systems requires robust encryption and authentication mechanisms. Furthermore, supporting mobility management and seamless handoffs between cells demands efficient signaling and location tracking mechanisms. Optimizing CDMA performance in heterogeneous network environments with diverse user densities and coverage areas is another challenge. Balancing system capacity, coverage, and spectral efficiency while minimizing costs is critical for successful CDMA deployment. Lastly, evolving standards and technologies require continuous adaptation and interoperability considerations in CDMA system implementation.

Experiment No. 6

Aim: Develop an application that uses GUI components.

Theory:

A typical user interface of an android application consists of action bar and the application content area.

- Main Action Bar
- View Control
- Content Area
- Split Action Bar

The basic unit of android application is the activity. A UI is defined in an xml file. During compilation, each element in the XML is compiled into equivalent Android GUI class with attributes represented by methods.

View and ViewGroups

An activity is consist of views. A view is just a widget that appears on the screen. It could be button etc. One or more views can be grouped together into one GroupView. Example of ViewGroup includes layouts.

Types of layout

There are many types of layout. Some of which are listed below −

- Linear Layout
- Absolute Layout
- Table Layout
- Frame Layout
- Relative Layout

The basic building block for user interface is a **View** object which is created from the View class and occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components like buttons, text fields, etc.

The **ViewGroup** is a subclass of **View** and provides invisible container that hold other Views or other ViewGroups and define their layout properties.

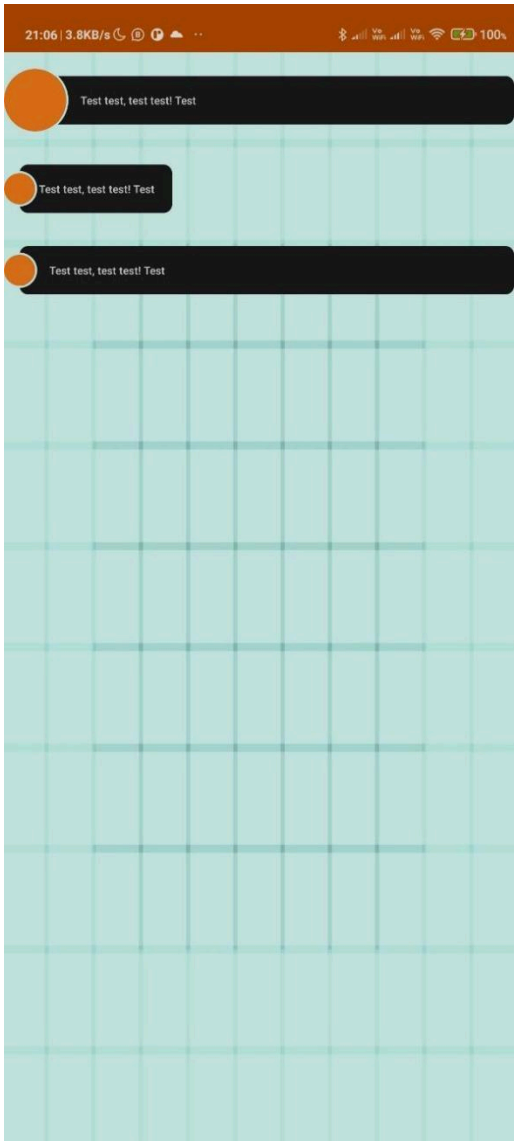
At third level we have different layouts which are subclasses of ViewGroup class and a typical layout defines the visual structure for an Android user interface and can be created either at run time using **View/ViewGroup** objects or you can declare your layout using simple XML file **main_layout.xml** which is located in the res/layout folder of your project.

Code:

```
object Components {

    const val ratingBar = ":libraries:rating-bar"
    const val dialogs = ":libraries:dialogs"
    const val imageSlider = ":libraries:image-slider"
    const val phoneNumber = ":libraries:phonenumber"
    const val toolbar = ":libraries:toolbar"
    const val suggestionInputView = ":libraries:suggestion-input-view"
    const val cardInputView = ":libraries:card-input-view"
    const val quantityPickerView = ":libraries:quantity-picker-view"
    const val timelineView = ":libraries:timeline-view"
    const val touchDelegator = ":libraries:touch-delegator"
    const val fitOptionMessageView = ":libraries:fit-option-message-view"

}
https://github.com/vinaynpp/mcc
object ComponentVersions {

    const val toolbarVersion = "2.0.5"
    const val suggestionInputViewVersion = "1.0.14"
    const val ratingBarVersion = "1.0.2"
    const val imageSliderVersion = "1.0.8"
    const val phoneNumberVersion = "1.0.2"
    const val dialogsVersion = "1.2.5"
    const val cardInputViewVersion = "1.1.2"
    const val quantityPickerViewVersion = "1.2.4"
    const val timelineViewVersion = "1.0.0"
    const val touchDelegatorVersion = "1.0.0"
    const val fitOptionMessageView = "1.0.0"
}

object Configs {

    const val compileSdkVersion = 29
    const val minSdkVersion = 21
    const val targetSdkVersion = 29
    const val buildToolsVersion = "29.0.3"

    const val applicationId = "com.trendyol.uicomponents"
    const val group = "com.trendyol.ui-components"
}

object Dependencies {
```

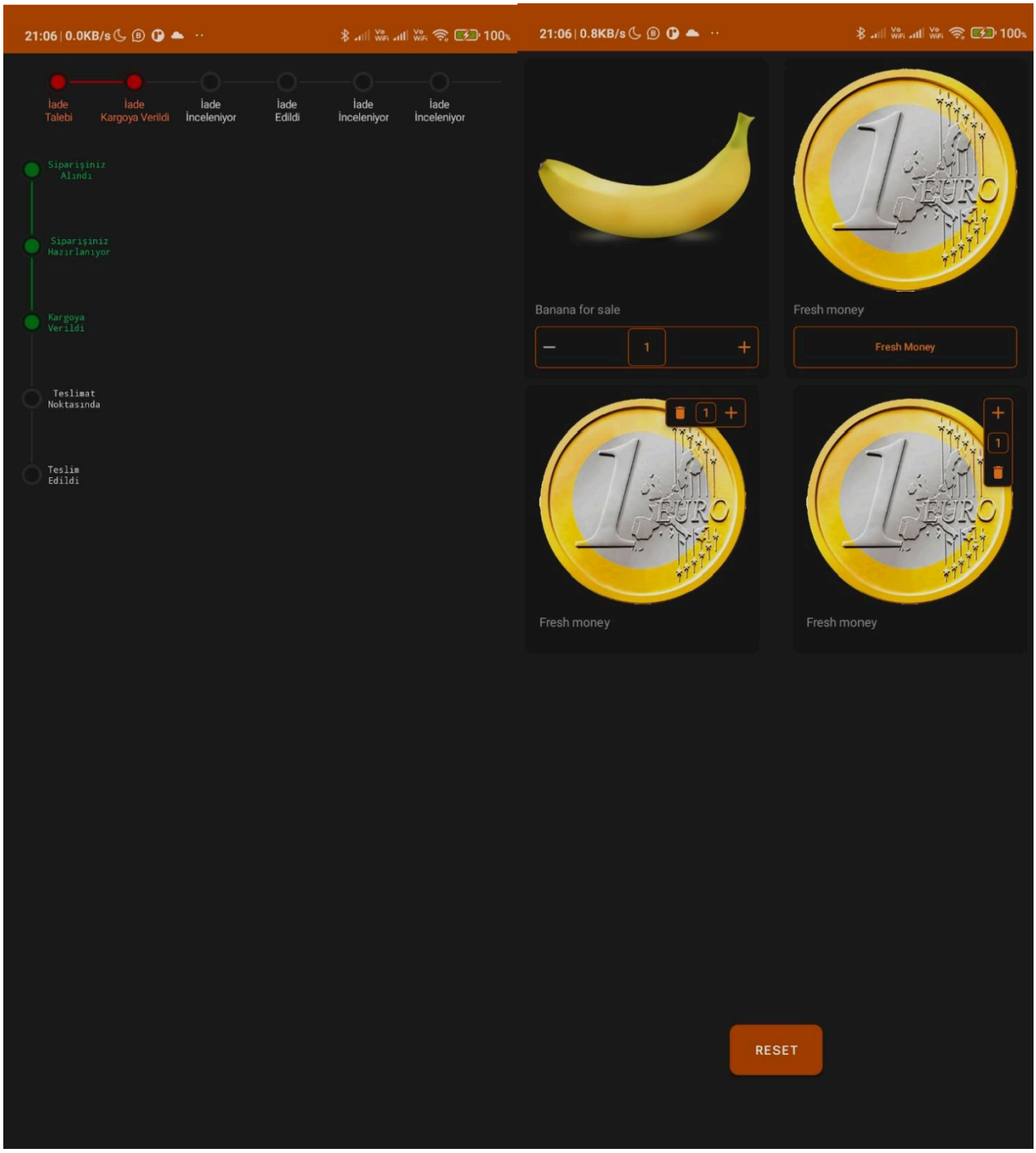
```
    const val kotlinJDK = "org.jetbrains.kotlin:kotlin-stdlib-jdk8:1.3.61"

    const val coreKtx = "androidx.core:core-ktx:1.3.2"
    const val appCompat = "androidx.appcompat:appcompat:1.2.0"
    const val material = "com.google.android.material:material:1.4.0"
    const val constraintLayout = "androidx.constraintlayout:constraintlayout:2.0.4"
    const val recyclerView = "androidx.recyclerview:recyclerview:1.2.0"
    const val circleIndicator = "com.github.MertNYuksel:CircleIndicator:2a2e973374"
    const val glide = "com.github.bumptech.glide:glide:4.11.0"
    const val glideCompiler = "com.github.bumptech.glide:compiler:4.11.0"
    const val lifecycleExtensions = "androidx.lifecycle:lifecycle-extensions:2.1.0"
    const val lifecycleCompiler = "androidx.lifecycle:lifecycle-compiler:2.1.0"
}

object Plugins {

    const val androidGradlePlugin = "com.android.tools.build:gradle:4.2.1"
    const val kotlinGradlePlugin = "org.jetbrains.kotlin:kotlin-gradle-plugin:1.5.0"
    const val mavenGradlePlugin = "com.github.dcendents:android-maven-gradle-plugin:2.1"

    const val androidApplication = "com.android.application"
    const val androidLibrary = "com.android.library"
    const val kotlinAndroid = "kotlin-android"
    const val kotlinKapt = "kotlin-kapt"
    const val kotlinParcelize = "kotlin-parcelize"
    const val androidMaven = "com.github.dcendents.android-maven"
}
```

Output:

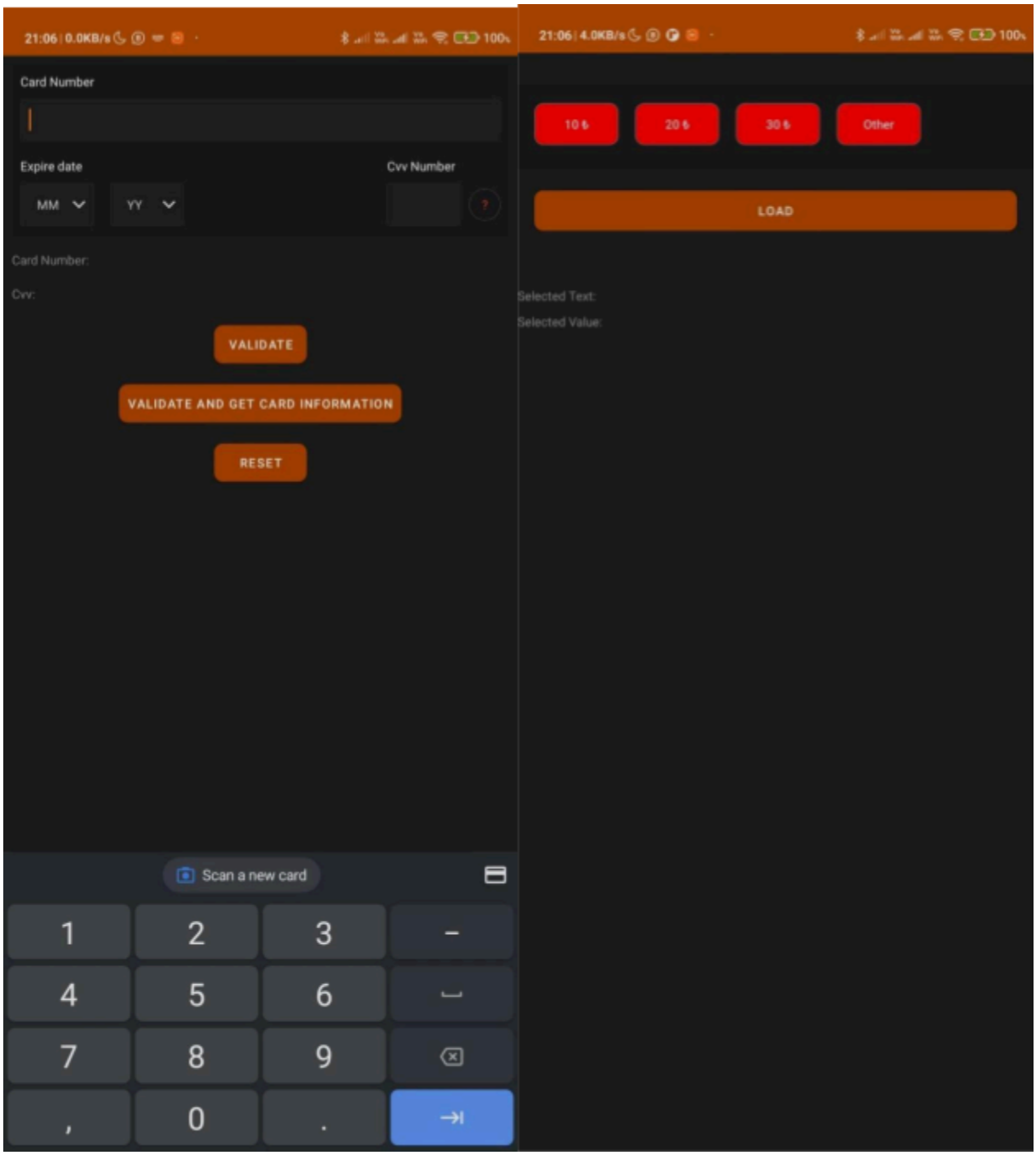

21:06 | 3.8KB/s
100
Test test, test test! Test
Test test, test test! Test
Test test, test test! Test

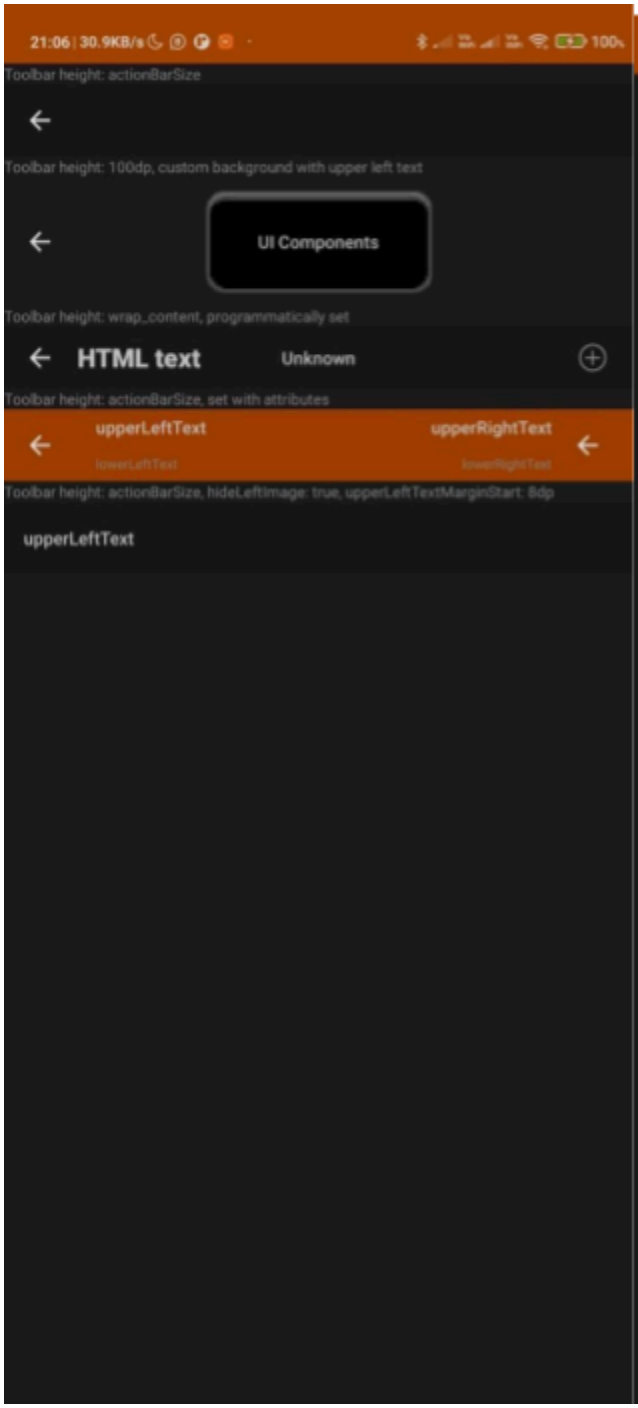
Fr. Conceicao Rodrigues College of Engineering

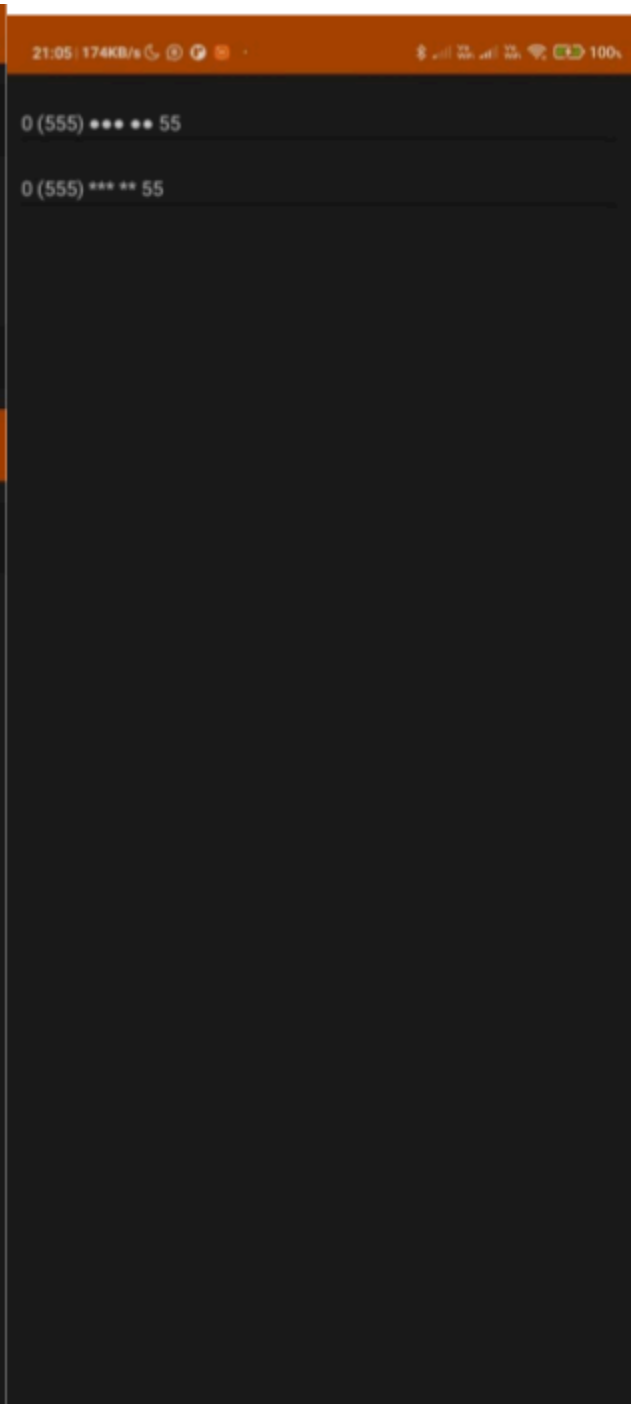
Fr. Agnel Ashram , Bandstand Bandra (west)



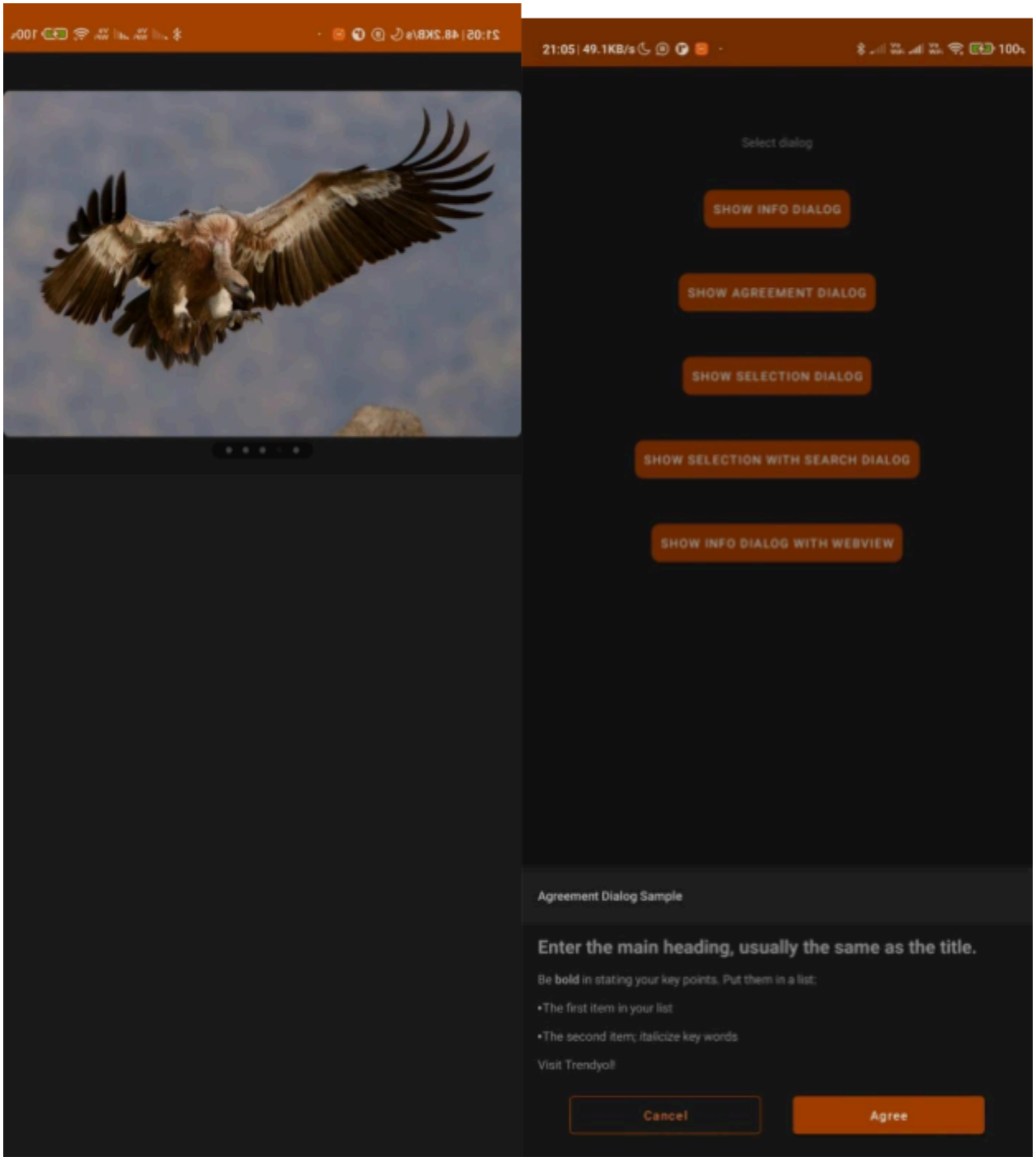
Fr. Conceicao Rodrigues College of Engineering
Fr. Agnel Ashram , Bandstand Bandra (west)




21:06 | 30.9KB/s
100
Toolbar height: actionBarSize
Toolbar height: 100dp, custom background with upper left text
UI Components
Toolbar height: wrap_content, programmatically set
HTML text
Unknown
Toolbar height: actionBarSize, set with attributes
upperLeftText
upperRightText
lowerLeftText
lowerRightText
Toolbar height: actionBarSize, hideLeftImage: true, upperLeftTextMarginStart: 8dp
upperLeftText

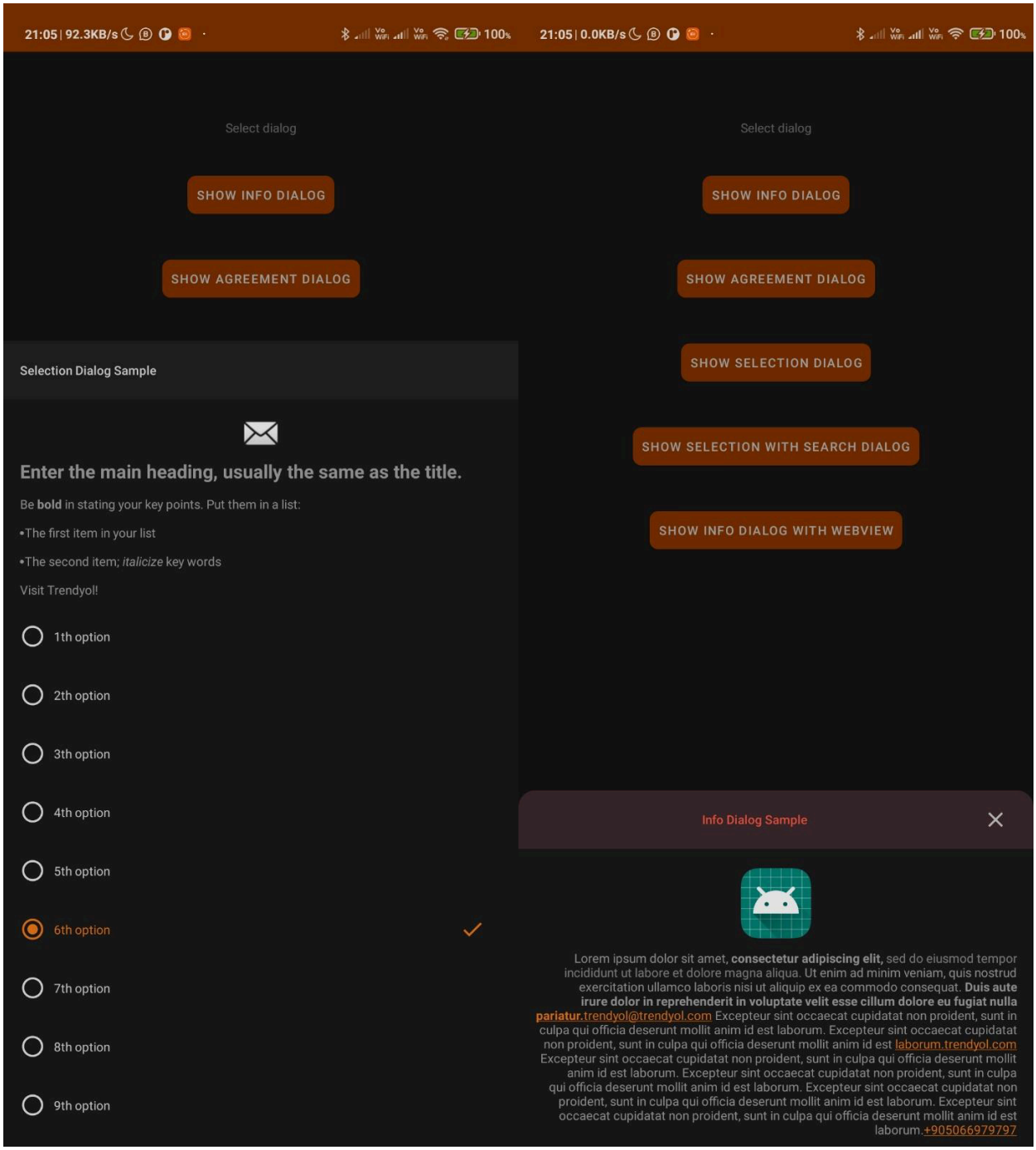

21:05 | 174KB/s
100
0 (555) ●●● ●● 55
0 (555) *** ** 55

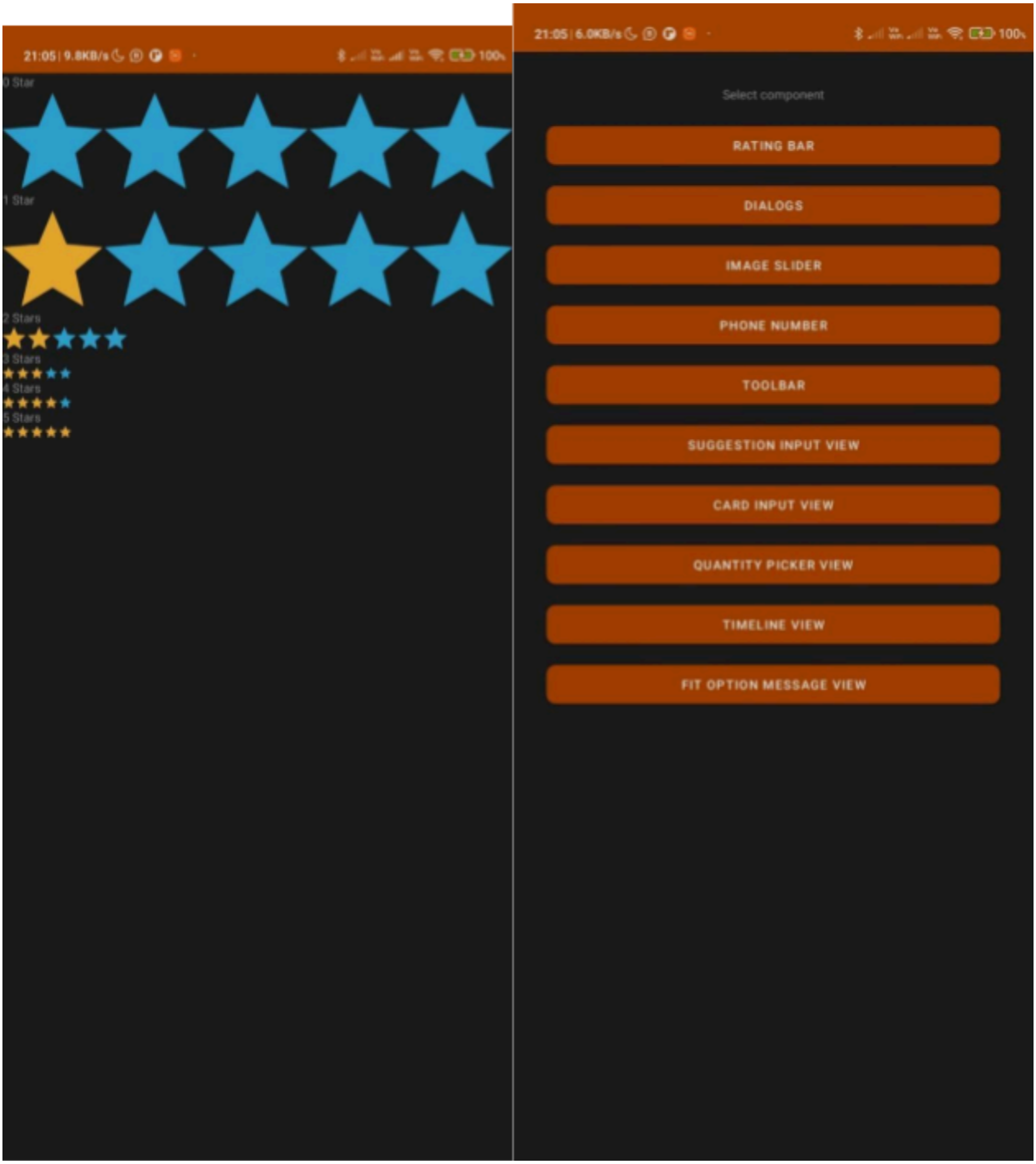
Fr. Conceicao Rodrigues College of Engineering
Fr. Agnel Ashram , Bandstand Bandra (west)



Fr. Conceicao Rodrigues College of Engineering

Fr. Agnel Ashram , Bandstand Bandra (west)





Conclusion:

Thus, we have performed the experiment to use the GUI components in the android studio and made an app which shows the color change the text and font type used also different size of the text, background color and successfully executed it.

Name: Emmanuel Gudinho

Class: TE COMPS B

Roll No: 9609

EXP 7:GSM SECURITY

PART A

(PART A: TO BE REFFERED BY STUDENTS)

A.1 Aim: To implement GSM security algorithms(A3/A5/A8)

A.2 Objectives: To understand the security algorithms in mobilenetworks

A.3 Outcomes: Student will be able to implement security algorithms for mobile communication network.(LO-4)

A.4 Tools Used/programming language: Java, Python etc

A.5 Theory:

- Authentication verifies identity and validity of SIM card to the network and ensures that subscriber has access to thenetwork.
- Termused
  - ✔ Ki= **individual subscriber authentication key**, it is 32 bit number and present only in SIM card and stored in authenticationcenter.
  - ✔ RAND= **random 128 bit number generated by AUC** (authentication center) when network request to authenticate thesubscribers.

✔ SRES (signed responses) = 32 bit crypto variable used in authenticationprocess.

✔ Kc = 64 bit cipherkey.

• MS is challenged by given RAND by thenetwork.

▪ Security inGSM

• **Three algorithms** have been specified to provide security services in GSM. **Algorithm A3** is used for **authentication**, **A5** for **encryption**, and **A8** for the **generation of a cipherkey**.

• In the GSM standard**only algorithm A5 was publicly available, whereas A3 and A8 were secret**, but standardized with openinterfaces.

• **Network providers can use stronger algorithms for authentication**– or users can apply stronger end-to-endencryption.

• Algorithms A3 and A8 (or their replacements) are located on the SIM and in the AUC and can beproprietary.

• Only A5 which is implemented in the devices has to be identical for all providers.

Subscriber Authentication

For subscriber authentication algorithm used is A3

1. A3 algorithm is inbuilt inside SIM and AUC, Input for A3 is Ki andRAND 2. Ki=Stored inside SIM(kiis encrypted inside SIM card) and not share on network and also present in AUC of MSC.

3. Before a subscriber can use any service from the GSM network, he or she must be authenticated. Authentication is based on the SIM, which stores the individual**authentication key K_i**, the **user identification IMSI**, and the algorithm used for authentication**A3**.

4. When user want to access GSM network IMSI number from SIM send to MSC then HLR then toAUC.

5. Now AUC check IMSI number is present or not and identify associated Ki value (Ki is fixed), in this procedure AUC generate RAND number which is different for every new user request.

6. AUC using authentication algorithm A3(input to A3 are ki and RAND) calculate SRES as output of A3 and AUC using algorithm A8 of cipher generation (input to A8arekiandRAND)calculateKcandsendtheseSRES,KcandRANDtoHLR59

then from HLR to MSC. These three terms SRES, Kc and RAND are called as triplet.

7. MSC now send only RAND value toMS

8. MS using algorithm A3 (input to A3 is Ki and RAND)calculate SRES and using algorithm A8 calculate Kc and send these SRES and kc toMSC

9. MSC check SRES receive from MS and Network are same or not. If both are same user is authenticated and connection is setup.

Figure: Subscriber Authentication

Encryption

1. To ensure privacy, all messages containing user-related information are encrypted in GSM over the airinterface.

2. After authentication, MS and BSS can start using encryption by applying the cipher key K_c

3. K_c is generated using the individual key K_i and a random value by applying the algorithm A8. Note that the SIM in the MS and the network both calculate the same K_c based on the random value RAND. **The key K_c itself is not transmitted over the airinterface.**

4. MS and BTS can now encrypt and decrypt data using the algorithm A5andthe cipher key K_c. As Figure shows, $K_c$should be a 64 bit key—60

which is not very strong, but is at least a good protection against simple

eavesdropping. However, the publication of A3 and A8 on the internet showed that in certain implementations 10 of the 64 bits are always set to 0, **so that the real length of the key is thus only 54 consequently**, the encryption is muchweaker.

5. Note: An **eavesdropping attack**, also known as a sniffing or snooping **attack**, is a theft of information as it is transmitted over a network by a computer, smart phone, or another connected device. The **attack** takes advantage of unsecured network communications to access data as it is being sent or received by its user. **Eavesdropping** is the act of intercepting communications between twopoints.

Figure: Data Encryption

A.6 Sample SourceCode:

https://www.theprogrammingcodeswarehouse.com/2020/04/implementation-of-a3-security.html

Code:

```
import random


k=random.getrandbits(128)

m=random.getrandbits(128)

kb=bin(k)[2:]

mb=bin(m)[2:]

kbl=kb[0:64]

kbr=kb[64:]

mbl=mb[0:64]

mbr=mb[64:]
```

a1=int(kbl,2)^int(mbr,2)

a2=int(kbr,2)^int(mbl,2)

a3=a1^a2

a4=bin(a3)[2:].zfill(64)

a5=a4[0:32]

a6=a4[32:]

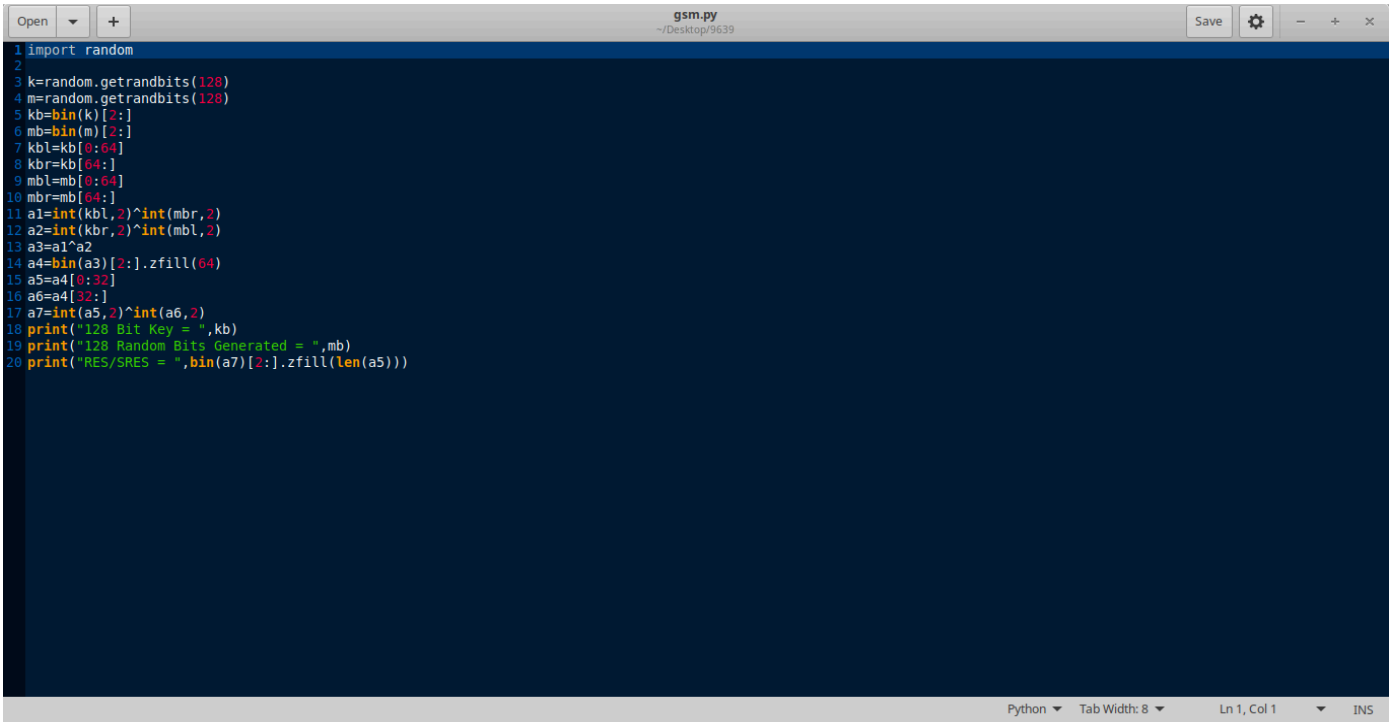
a7=int(a5,2)^int(a6,2)

print("128 Bit Key = ",kb)

print("128 Random Bits Generated = ",mb)

print("RES/SRES = ",bin(a7)[2:].zfill(len(a5)))

Output:



A.6 Sample Output:

128 Bit Key

=11111011101001100100000100100110001001110011111010011101011010001111000011100000

1 111

01110111011011101010001011010100011101000 1

128 Random Bits Generated

=11000001000100010110001011100100110110101100110010001101011100010010000010100 10

1 001

00000100111100000010000110010011111111000100

RES/SRES=11110110110100000010111110001101

Experiment No.: 8

Aim: To make an application that draws basic graphical primitives on the screen.

Theory:

The android.graphics.Canvas can be used to draw graphics in android. It provides methods to draw oval, rectangle, picture, text, line etc.

The android.graphics.Paint class is used with canvas to draw objects. It holds the information of color and style.

Android Canvas class encapsulates the bitmaps used as surface. It exposes the draw methods which can be used for designing. Let us first clear the following terms:
Bitmap: The surface being drawn on.
Paint: It lets us specify how to draw the primitives on bitmap. It is also referred to as "Brush".
Canvas: It supplies the draw methods used to draw primitives on underlying bitmap.
Each drawing object specifies a paint object to render. Let us see the available list of drawing objects and they are as follows:
drawArc: This draws an arc between the two angles bounded by an area of rectangle.
drawBitmap: It draws an bitmap on canvas.
drawRGB/drawARGB/drawColor: This fills the canvas with a single color.
drawBitmapMesh: It draws a bitmap using a mesh. It manipulates the appearance of target by moving points on it.
drawCircle: This draws a circle on a specified radius centered on a given point.
drawLine(s):it draws a line (or series of lines) between points.
drawOval: it draws an oval which is bounded by the area of
rectangle. **drawPaint**: It fills the entire canvas with a specific paint.
drawPath: It draws a path as per specification.
drawPicture: It draws a picture specified on a rectangular area.
drawPosText: it draws a text string specifying the offset of each character.
drawRect: It draws a rectangle.
drawRoundRect: it draws a rectangle with round edges.
drawText: It draws a text string on canvas.
The **Paint** class consists of a paint brush and a palette. It lets us choose how to render the primitives drawn into canvas by draw methods. We can control the color, style, font, special effects etc can be modified by modifying the paint object. For instance, **setColor** method can be used to select the color of Paint. Paint class supports transparency so it can be used to control variety of shades or effects, etc. Let us create a simple example and see the basic usage of canvas and paint.

Code:

```
package co.martinbaciga.drawingtest.ui.component;
https://github.com/vinaynpp/mcc
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.PorterDuff;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;

import java.util.ArrayList;

public class DrawingView extends View
{
	private Path mDrawPath;
	private Paint mBackgroundPaint;
	private Paint mDrawPaint;
	private Canvas mDrawCanvas;
	private Bitmap mCanvasBitmap;

	private ArrayList<Path> mPaths = new ArrayList<>();
	private ArrayList<Paint> mPaints = new ArrayList<>();
	private ArrayList<Path> mUndonePaths = new ArrayList<>();
	private ArrayList<Paint> mUndonePaints = new ArrayList<>();

	// Set default values
	private int mBackgroundColor = 0xFFFFFFFF;
	private int mPaintColor = 0xFF660000;
	private int mStrokeWidth = 10;

	public DrawingView(Context context, AttributeSet attrs)
	{
		super(context, attrs);
		init();
	}

	private void init()
	{
		mDrawPath = new Path();
		mBackgroundPaint = new Paint();
		initPaint();
	}
```

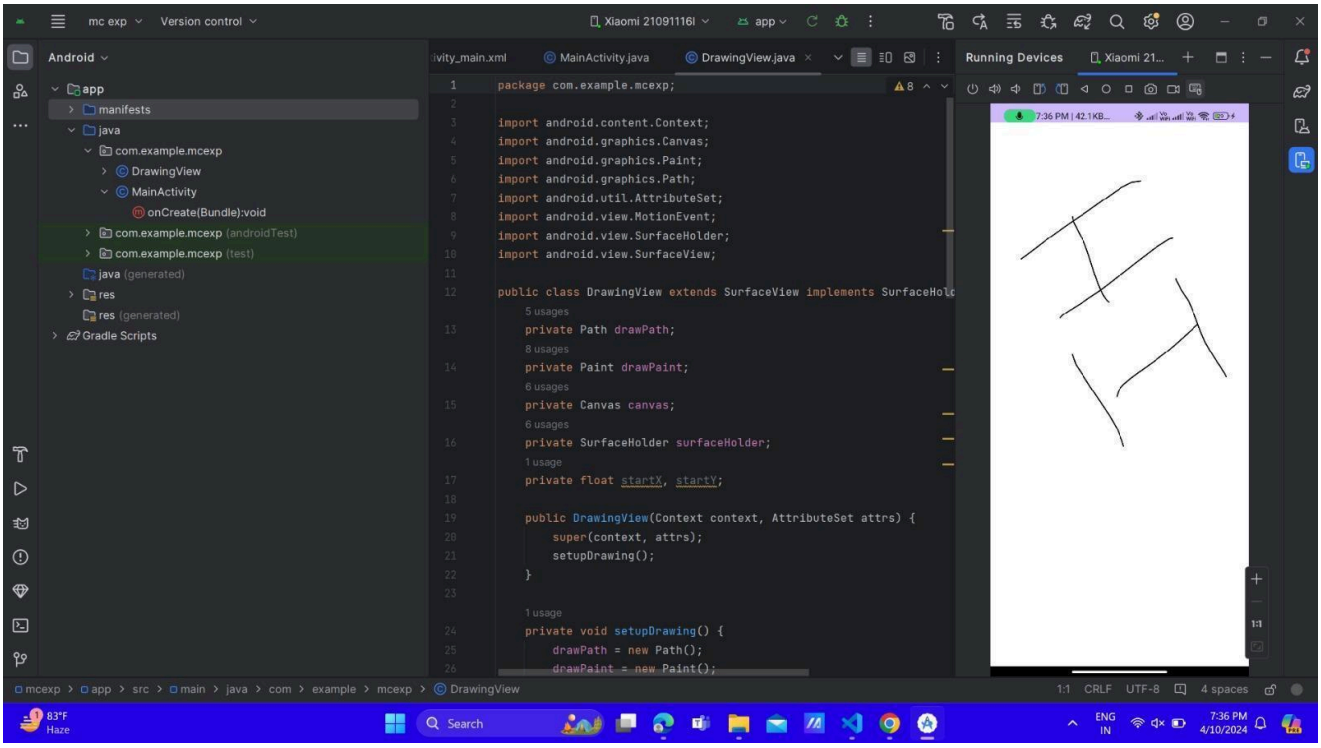
private void initPaint()

```
{
        mDrawPaint = new Paint();
        mDrawPaint.setColor(mPaintColor);
        mDrawPaint.setAntiAlias(true);
        mDrawPaint.setStrokeWidth(mStrokeWidth);
        mDrawPaint.setStyle(Paint.Style.STROKE);
        mDrawPaint.setStrokeJoin(Paint.Join.ROUND);
        mDrawPaint.setStrokeCap(Paint.Cap.ROUND);
}

private void drawBackground(Canvas canvas)
{
        mBackgroundPaint.setColor(mBackgroundColor);
        mBackgroundPaint.setStyle(Paint.Style.FILL);
        canvas.drawRect(0, 0, this.getWidth(), this.getHeight(), mBackgroundPaint);
}

private void drawPaths(Canvas canvas)
{
        int i = 0;
        for (Path p : mPaths)
        {
                canvas.drawPath(p, mPaints.get(i));
                i++;
        }
}

@Override
protected void onDraw(Canvas canvas)
{
        drawBackground(canvas);
        drawPaths(canvas);

        canvas.drawPath(mDrawPath, mDrawPaint);
}

@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh)
{
        super.onSizeChanged(w, h, oldw, oldh);

        mCanvasBitmap = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);

        mDrawCanvas = new Canvas(mCanvasBitmap);
}

@Override
public boolean onTouchEvent(MotionEvent event)
{
        float touchX = event.getX();
        float touchY = event.getY();

        switch (event.getAction())
        {
```

```
                case MotionEvent.ACTION_DOWN:
                        mDrawPath.moveTo(touchX, touchY);
                        //mDrawPath.addCircle(touchX, touchY, mStrokeWidth/10, Path.Direction.CW);
                        break;
                case MotionEvent.ACTION_MOVE:
                        mDrawPath.lineTo(touchX, touchY);
                        break;
                case MotionEvent.ACTION_UP:
                        mDrawPath.lineTo(touchX, touchY);
                        mPaths.add(mDrawPath);
                        mPaints.add(mDrawPaint);
                        mDrawPath = new Path();
                        initPaint();
                        break;
                                        default:  return false;

                }


        invalidate();
        return true;
}

public void clearCanvas()
{
        mPaths.clear();
        mPaints.clear();
        mUndonePaths.clear();
        mUndonePaints.clear();
        mDrawCanvas.drawColor(0, PorterDuff.Mode.CLEAR);
        invalidate();
}

public void setPaintColor(int color)
{
        mPaintColor = color;
        mDrawPaint.setColor(mPaintColor);
}

public void setPaintStrokeWidth(int strokeWidth)
{
        mStrokeWidth = strokeWidth;
        mDrawPaint.setStrokeWidth(mStrokeWidth);
}

public void setBackgroundColor(int color)
{
        mBackgroundColor = color;
        mBackgroundPaint.setColor(mBackgroundColor);
        invalidate();
}

public Bitmap getBitmap()
{
        drawBackground(mDrawCanvas);
```

```
                drawPaths(mDrawCanvas)
                ; return mCanvasBitmap;
        }

        public void undo()
        {
                if (mPaths.size() > 0)
                {
                        mUndonePaths.add(mPaths.remove(mPaths.size() - 1));
                        mUndonePaints.add(mPaints.remove(mPaints.size() - 1));
                        invalidate();
                }
        }

        public void redo()
        {
                if (mUndonePaths.size() > 0)
                {
                        mPaths.add(mUndonePaths.remove(mUndonePaths.size() - 1));
                        mPaints.add(mUndonePaints.remove(mUndonePaints.size() - 1));
                        invalidate();
                }
        }
}


package co.martinbaciga.drawingtest.domain.manager;

import android.Manifest;

import android.app.Activity;
import android.content.pm.PackageManager;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;

public class PermissionManager
{
        public static final int REQUEST_WRITE_STORAGE = 112;

        public static boolean checkWriteStoragePermissions(Activity activity)
        {
                boolean hasPermission = (ContextCompat.checkSelfPermission(activity,
Manifest.permission.WRITE_EXTERNAL_STORAGE) == PackageManager.PERMISSION_GRANTED);
                if (!hasPermission) {
                        ActivityCompat.requestPermissions(activity,
                                        new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},
                                        REQUEST_WRITE_STORAGE);
                }
                return hasPermission;
        }
}
```

Output:


mc exp
Version control
Xiaomi 21091116I
app
Android
app
manifests
java
com.example.mcexp
DrawingView
MainActivity
onCreate(Bundle):void
com.example.mcexp (androidTest)
com.example.mcexp (test)
java (generated)
res
res (generated)
Gradle Scripts
MainActivity.java
DrawingView.java
Running Devices
Xiaomi 21...
package com.example.mcexp;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Path;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
public class DrawingView extends SurfaceView implements SurfaceHolc
5 usages
private Path drawPath;
8 usages
private Paint drawPaint;
6 usages
private Canvas canvas;
6 usages
private SurfaceHolder surfaceHolder;
1 usage
private float startX, startY;
public DrawingView(Context context, AttributeSet attrs) {
super(context, attrs);
setupDrawing();
}
1 usage
private void setupDrawing() {
drawPath = new Path();
drawPaint = new Paint();
mcexp
app
src
main
java
com
example
mcexp
DrawingView
1:1
CRLF
UTF-8
4 spaces
83°F
Haze
Search
ENG
IN
7:36 PM
4/10/2024

Fr. Conceicao Rodrigues College of Engineering

Fr. Agnel Ashram , Bandstand Bandra (west)

Conclusion:

Thus, we have performed the experiment to draw basic graphical primitives on the screen in the android app using canvas in android studio and over here we have drawn circle and rectangle as example and successfully executed it.

Experiment No.:

9 Aim: Develop an application that makes use of database

Theory:

SQLite is a opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation.

SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC,ODBC e.t.c

Database - Package

The main package is android.database.sqlite that contains the classes to manage your own databases

Database - Creation

In order to create a database you just need to call this method openOrCreateDatabase with your database name and mode as a parameter.

Database - Insertion

we can create table or insert data into table using execSQL method defined in SQLiteDatabase class.

Database - Fetching

We can retrieve anything from database using an object of the Cursor class. We will call a method of this class called rawQuery and it will return a resultset with the cursor pointing to the table. We can move the cursor forward and retrieve the data.

Database - Helper class

For managing all the operations related to the database , an helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and update of the database.

Code:

```
package com.example.inventory
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.navigation.NavController
import androidx.navigation.fragment.NavHostFragment
import androidx.navigation.ui.NavigationUI.setupActionBarWithNavController
```

```
class MainActivity : AppCompatActivity(R.layout.activity_main) {

    private lateinit var navController: NavController

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Retrieve NavController from the NavHostFragment
        val navHostFragment = supportFragmentManager
            .findFragmentById(R.id.nav_host_fragment) as NavHostFragment
        navController = navHostFragment.navController
        // Set up the action bar for use with the NavController
        setupActionBarWithNavController(this, navController)
    }

    /**
     * Handle navigation when the user chooses Up from the action bar.
     */
    override fun onSupportNavigateUp(): Boolean {
        return navController.navigateUp() || super.onSupportNavigateUp()
    }
}
```

https://github.com/vinaynpp/mcc

```
package com.example.inventory

import android.content.Context.INPUT_METHOD_SERVICE
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.inputmethod.InputMethodManager
import android.widget.TextView
import androidx.fragment.app.Fragment
import androidx.fragment.app.activityViewModels
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs
import com.example.inventory.data.Item
import com.example.inventory.databinding.FragmentAddItemBinding

/**
 * Fragment to add or update an item in the Inventory database.
 */
```

```
class AddItemFragment : Fragment() {

    // Use the 'by activityViewModels()' Kotlin property delegate from the fragment-ktx artifact
    // to share the ViewModel across fragments.
    private val viewModel: InventoryViewModel by activityViewModels {
        InventoryViewModelFactory(
            (activity?.application as InventoryApplication).database
                .itemDao()
        )
    }
    private val navigationArgs: ItemDetailFragmentArgs by navArgs()

    lateinit var item: Item

    // Binding object instance corresponding to the fragment_add_item.xml layout
    // This property is non-null between the onCreateView() and onDestroyView() lifecycle callbacks,
    // when the view hierarchy is attached to the fragment
    private var _binding: FragmentAddItemBinding? = null
    private val binding get() = _binding!!

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        _binding = FragmentAddItemBinding.inflate(inflater, container, false)
        return binding.root
    }

    /**
     * Returns true if the EditTexts are not empty
     */
    private fun isEntryValid(): Boolean {
        return viewModel.isEntryValid(
            binding.itemName.text.toString(),
            binding.itemPrice.text.toString(),
            binding.itemCount.text.toString(),
        )
    }

    /**
     * Binds views with the passed in [item] information.
     */
    private fun bind(item: Item) {
        val price = "%.2f".format(item.itemPrice)
        binding.apply {
```

```
            itemName.setText(item.itemName, TextView.BufferType.SPANNABLE)
            itemPrice.setText(price, TextView.BufferType.SPANNABLE)
            itemCount.setText(item.quantityInStock.toString(), TextView.BufferType.SPANNABLE)
            saveAction.setOnClickListener { updateItem() }
        }
    }

    /**
     * Inserts the new Item into database and navigates up to list fragment.
     */
    private fun addNewItem() {
        if (isEntryValid()) {
            viewModel.addNewItem(
                binding.itemName.text.toString(),
                binding.itemPrice.text.toString(),
                binding.itemCount.text.toString(),
            )
            val action = AddItemFragmentDirections.actionAddItemFragmentToItemListFragment()
            findNavController().navigate(action)
        }
    }

    /**
     * Updates an existing Item in the database and navigates up to list fragment.
     */
    private fun updateItem() {
        if (isEntryValid()) {
            viewModel.updateItem(
                this.navigationArgs.itemId,
                this.binding.itemName.text.toString(),
                this.binding.itemPrice.text.toString(),
                this.binding.itemCount.text.toString()
            )
            val action = AddItemFragmentDirections.actionAddItemFragmentToItemListFragment()
            findNavController().navigate(action)
        }
    }

    /**
     * Called when the view is created.
     * The itemId Navigation argument determines the edit item or add new item.
     * If the itemId is positive, this method retrieves the information from the database and
     * allows the user to update it.
     */
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
```

```
        val id = navigationArgs.itemId
        if (id > 0) {
            viewModel.retrieveItem(id).observe(this.viewLifecycleOwner) { selectedItem ->
                item = selectedItem
                bind(item)
            }
        } else {
            binding.saveAction.setOnClickListener {
                addNewItem()
            }
        }
    }

    /**
     * Called before fragment is destroyed.
     */
    override fun onDestroyView() {
        super.onDestroyView()
        // Hide keyboard.
        val inputMethodManager = requireActivity().getSystemService(INPUT_METHOD_SERVICE) as
                InputMethodManager
        inputMethodManager.hideSoftInputFromWindow(requireActivity().currentFocus?.windowToken, 0)
        _binding = null
    }
}
```

```
package com.example.inventory

import android.app.Application
import com.example.inventory.data.ItemRoomDatabase


class InventoryApplication : Application() {
    // Using by lazy so the database is only created when needed
    // rather than when the application starts
    val database: ItemRoomDatabase by lazy { ItemRoomDatabase.getDatabase(this) }
}
```

```
package com.example.inventory

import androidx.lifecycle.LiveData
import androidx.lifecycle.ViewModel
```

```
import androidx.lifecycle.ViewModelProvider
import androidx.lifecycle.asLiveData
import androidx.lifecycle.viewModelScope
import com.example.inventory.data.Item
import com.example.inventory.data.ItemDao
import kotlinx.coroutines.launch

/**
 * View Model to keep a reference to the Inventory repository and an up-to-date list of all items.
 *
 */
class InventoryViewModel(private val itemDao: ItemDao) : ViewModel() {

    // Cache all items form the database using LiveData.
    val allItems: LiveData<List<Item>> = itemDao.getItems().asLiveData()

    /**
     * Returns true if stock is available to sell, false otherwise.
     */
    fun isStockAvailable(item: Item): Boolean {
        return (item.quantityInStock > 0)
    }

    /**
     * Updates an existing Item in the database.
     */
    fun updateItem(
        itemId: Int,
        itemName: String,
        itemPrice: String,
        itemCount: String
    ) {
        val updatedItem = getUpdatedItemEntry(itemId, itemName, itemPrice, itemCount)
        updateItem(updatedItem)
    }


    /**
     * Launching a new coroutine to update an item in a non-blocking way
     */
    private fun updateItem(item: Item) {
        viewModelScope.launch {
            itemDao.update(item)
        }
    }
```

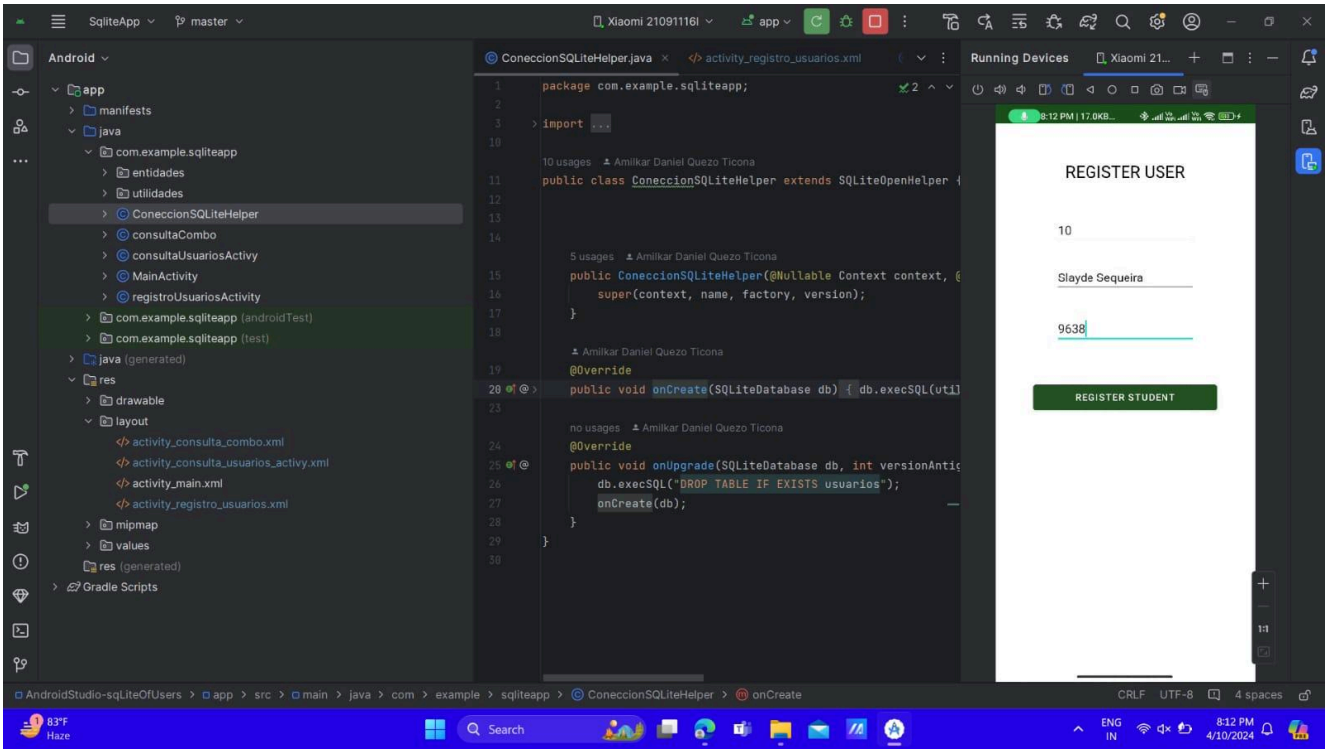
https://github.com/vinaynpp/mcc

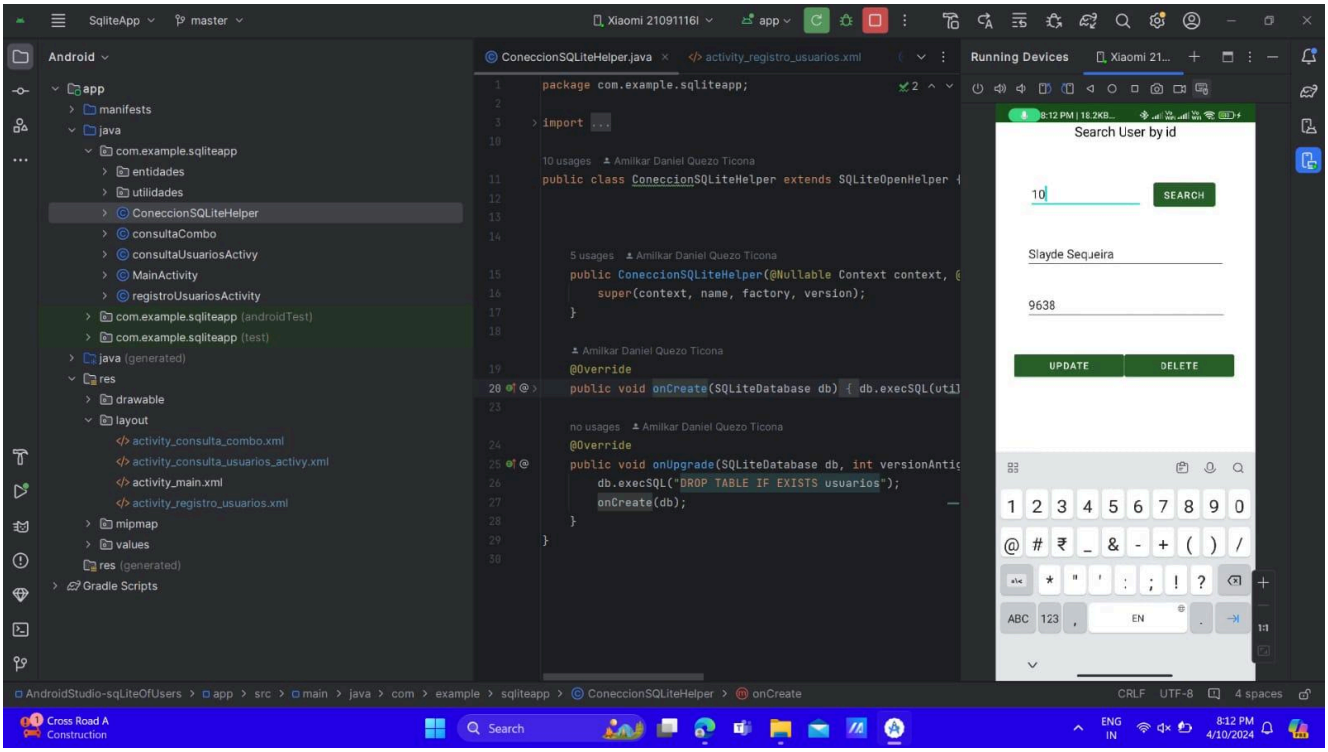
```
/**
 * Decreases the stock by one unit and updates the database.
 */
fun sellItem(item: Item) {
    if (item.quantityInStock > 0) {
        // Decrease the quantity by 1
        val newItem = item.copy(quantityInStock = item.quantityInStock - 1)
        updateItem(newItem)
    }
}

/**
 * Inserts the new Item into database.
 */
fun addNewItem(itemName: String, itemPrice: String, itemCount: String) {
    val newItem = getNewItemEntry(itemName, itemPrice, itemCount)
    insertItem(newItem)
}

/**
 * Launching a new coroutine to insert an item in a non-blocking way
 */
private fun insertItem(item: Item) {
    viewModelScope.launch {
        itemDao.insert(item)
    }
}

/**
 * Launching a new coroutine to delete an item in a non-blocking way
 */
fun deleteItem(item: Item) {
    viewModelScope.launch {
        itemDao.delete(item)
    }
}

/**
 * Retrieve an item from the repository.
 */
fun retrieveItem(id: Int): LiveData<Item> {
    return itemDao.getItem(id).asLiveData()
}

/**
 * Returns true if the EditTexts are not empty
```

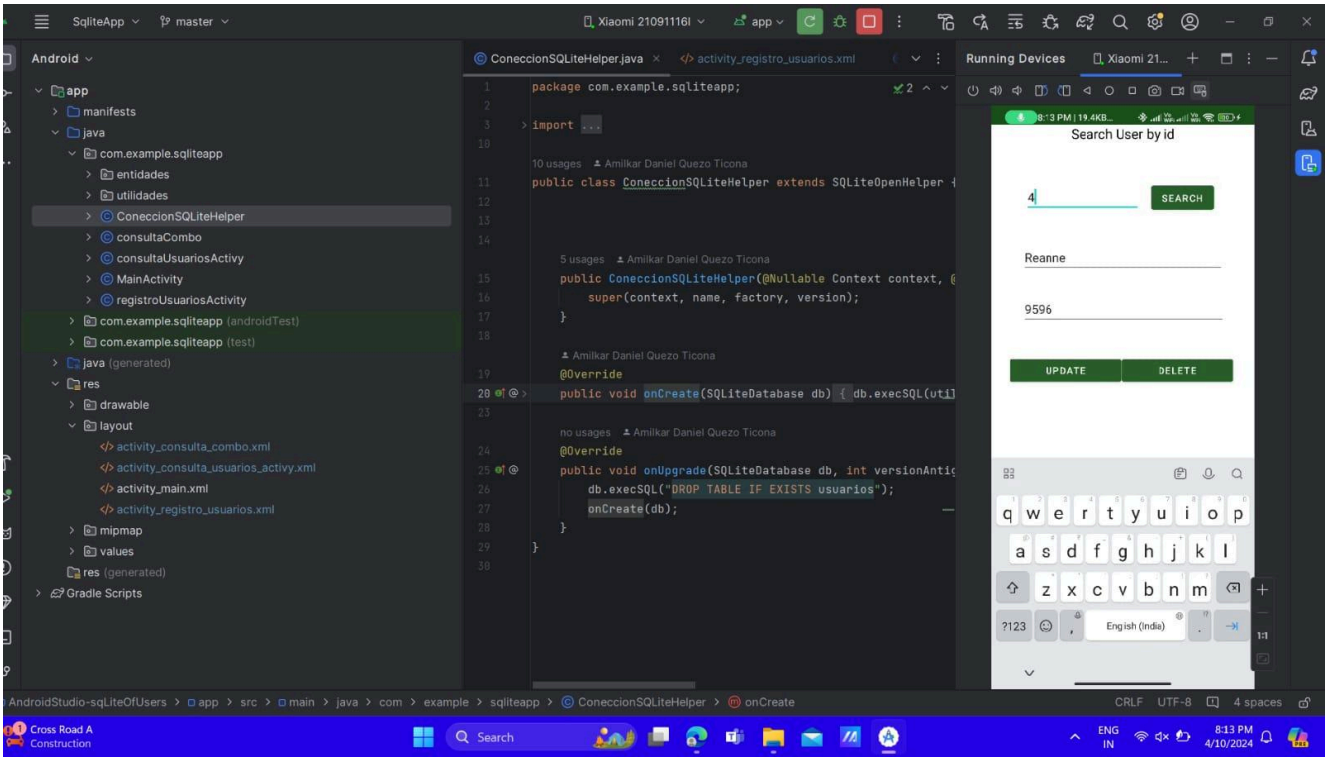
```
     */
    fun isEntryValid(itemName: String, itemPrice: String, itemCount: String): Boolean {
        if (itemName.isBlank() || itemPrice.isBlank() || itemCount.isBlank()) {
            return false
        }
        return true
    }

    /**
     * Returns an instance of the [Item] entity class with the item info entered by the user.
     * This will be used to add a new entry to the Inventory database.
     */
    private fun getNewItemEntry(itemName: String, itemPrice: String, itemCount: String): Item {
        return Item(
            itemName = itemName,
            itemPrice = itemPrice.toDouble(),
            quantityInStock = itemCount.toInt()
        )
    }

    /**
     * Called to update an existing entry in the Inventory database.
     * Returns an instance of the [Item] entity class with the item info updated by the user.
     */
    private fun getUpdatedItemEntry(
        itemId: Int,
        itemName: String,
        itemPrice: String,
        itemCount: String
    ): Item {
        return Item(
            id = itemId,
            itemName = itemName,
            itemPrice = itemPrice.toDouble(),
            quantityInStock = itemCount.toInt()
        )
    }
}

/**
 * Factory class to instantiate the [ViewModel] instance.
 */
class InventoryViewModelFactory(private val itemDao: ItemDao) : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(InventoryViewModel::class.java)) {
            @Suppress("UNCHECKED_CAST")
```

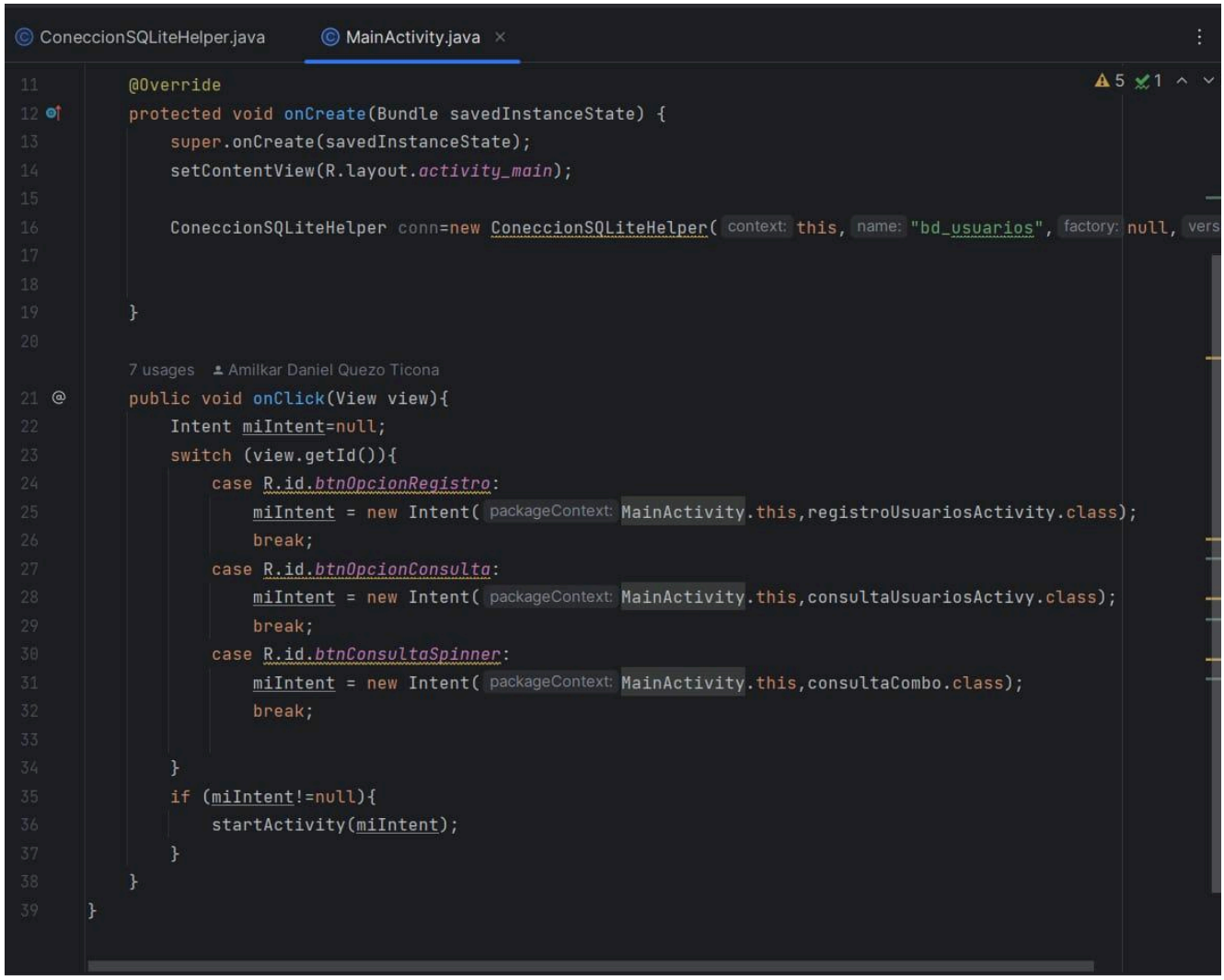
```
            return InventoryViewModel(itemDao) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

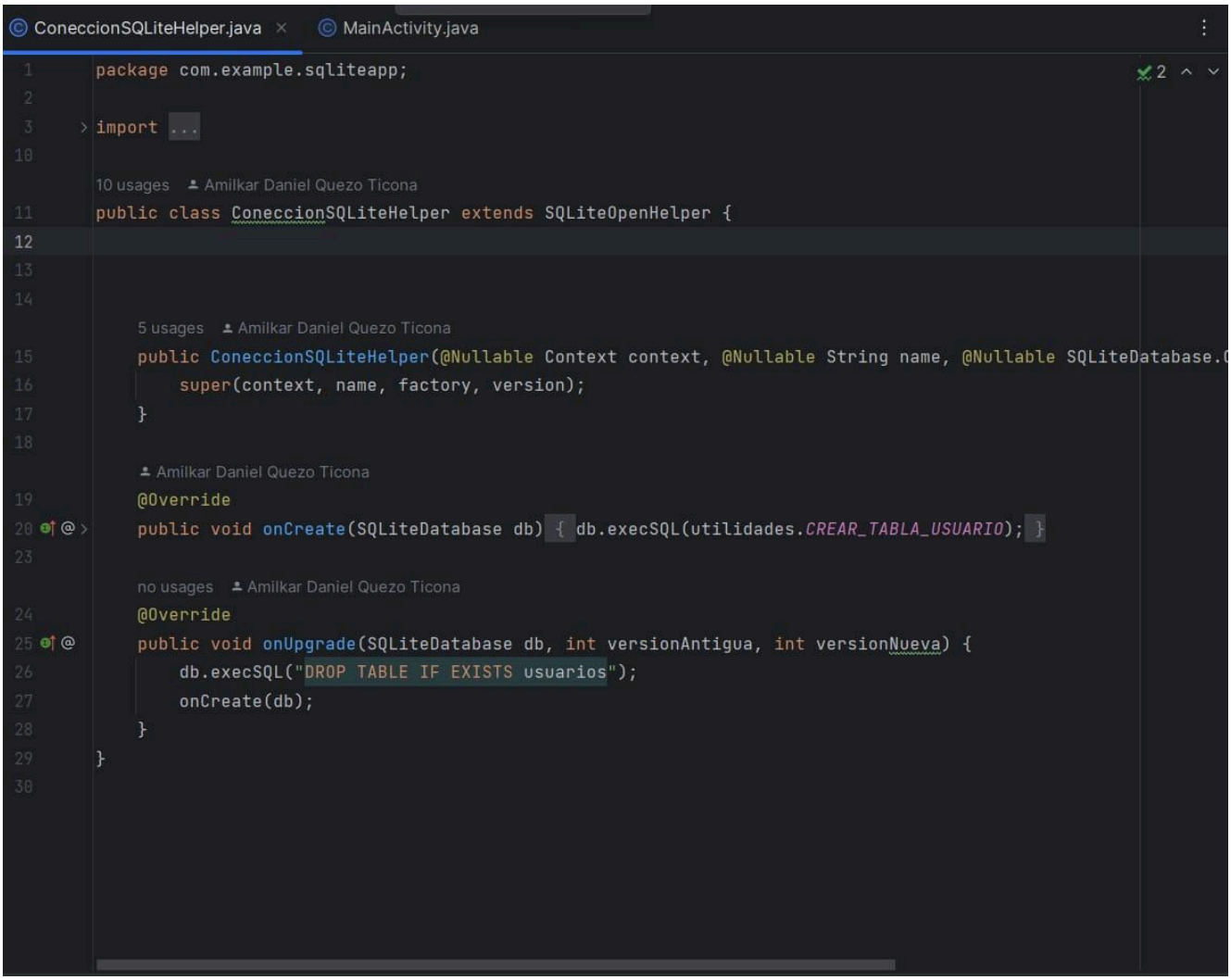
Output:




SqliteApp
master
Xiaomi 21091116I
app
Android
app
manifests
java
com.example.sqliteapp
entidades
utilidades
ConeccionSQLiteHelper
consultaCombo
consultaUsuariosActivy
MainActivity
registroUsuariosActivity
com.example.sqliteapp (androidTest)
com.example.sqliteapp (test)
java (generated)
res
drawable
layout
activity_consulta_combo.xml
activity_consulta_usuarios_activy.xml
activity_main.xml
activity_registro_usuarios.xml
mipmap
values
res (generated)
Gradle Scripts
ConeccionSQLiteHelper.java
activity_registro_usuarios.xml
package com.example.sqliteapp;
import ...
10 usages Amilkar Daniel Quezo Ticona
public class ConeccionSQLiteHelper extends SQLiteOpenHelper {
5 usages Amilkar Daniel Quezo Ticona
public ConeccionSQLiteHelper(@Nullable Context context, @
super(context, name, factory, version);
}
Amilkar Daniel Quezo Ticona
@Override
public void onCreate(SQLiteDatabase db) { db.execSQL(util
no usages Amilkar Daniel Quezo Ticona
@Override
public void onUpgrade(SQLiteDatabase db, int versionAntig
db.execSQL("DROP TABLE IF EXISTS usuarios");
onCreate(db);
}
}
Running Devices
Xiaomi 21...
Search User by id
10
SEARCH
Slayde Sequeira
9638
UPDATE
DELETE
AndroidStudio-sqLiteOfUsers
app
src
main
java
com
example
sqliteapp
ConeccionSQLiteHelper
onCreate
CRLF
UTF-8
4 spaces
Cross Road A
Construction
Search
ENG
IN
8:12 PM
4/10/2024


SqliteApp
master
Xiaomi 21091116I
app
Android
app
manifests
java
com.example.sqliteapp
entidades
utilidades
ConeccionSQLiteHelper
consultaCombo
consultaUsuariosActivy
MainActivity
registroUsuariosActivity
com.example.sqliteapp (androidTest)
com.example.sqliteapp (test)
java (generated)
res
drawable
layout
activity_consulta_combo.xml
activity_consulta_usuarios_activy.xml
activity_main.xml
activity_registro_usuarios.xml
mipmap
values
res (generated)
Gradle Scripts
ConeccionSQLiteHelper.java
activity_registro_usuarios.xml
package com.example.sqliteapp;
import ...
10 usages Amilkar Daniel Quezo Ticona
public class ConeccionSQLiteHelper extends SQLiteOpenHelper {
5 usages Amilkar Daniel Quezo Ticona
public ConeccionSQLiteHelper(@Nullable Context context, @
super(context, name, factory, version);
}
Amilkar Daniel Quezo Ticona
@Override
public void onCreate(SQLiteDatabase db) { db.execSQL(util
no usages Amilkar Daniel Quezo Ticona
@Override
public void onUpgrade(SQLiteDatabase db, int versionAntig
db.execSQL("DROP TABLE IF EXISTS usuarios");
onCreate(db);
}
}
Running Devices
Xiaomi 21...
Search User by id
4
SEARCH
Reanne
9596
UPDATE
DELETE
AndroidStudio-sqLiteOfUsers
app
src
main
java
com
example
sqliteapp
ConeccionSQLiteHelper
onCreate
CRLF
UTF-8
4 spaces
Cross Road A
Construction
Search
ENG
IN
8:13 PM
4/10/2024


ConeccionSQLiteHelper.java
MainActivity.java
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
ConeccionSQLiteHelper conn=new ConeccionSQLiteHelper(context: this, name: "bd_usuarios", factory: null, vers
}
7 usages Amilkar Daniel Quezo Ticona
public void onClick(View view){
Intent miIntent=null;
switch (view.getId()){
case R.id.btnOpcionRegistro:
miIntent = new Intent(packageContext: MainActivity.this,registroUsuariosActivity.class);
break;
case R.id.btnOpcionConsulta:
miIntent = new Intent(packageContext: MainActivity.this,consultaUsuariosActivy.class);
break;
case R.id.btnConsultaSpinner:
miIntent = new Intent(packageContext: MainActivity.this,consultaCombo.class);
break;
}
if (miIntent!=null){
startActivity(miIntent);
}
}
}



Conclusion:

Thus, we have performed the experiment and made an app that uses SQLite database over here takes the student name and roll number and input and stores that in database which can be viewed later. We have successfully executed the experiment.

Experiment No.: 10

Aim: Implement an application that creates an alert upon receiving a message.

Theory:

A notification is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.

Android **Toast** class provides a handy way to show users alerts but problem is that these alerts are not persistent which means alert flashes on the screen for a few seconds and then disappears.

To see the details of the notification, you will have to select the icon which will display notification drawer having detail about the notification. While working with emulator with virtual device, you will have to click and drag down the status bar to expand it which will give you detail as follows. This will be just **64 dp** tall and called normal view.

Create and Send Notifications

You have simple way to create a notification. Follow the following steps in your application to create a notification –

Step 1 - Create Notification Builder

As a first step is to create a notification builder using *NotificationCompat.Builder.build()*. You will use Notification Builder to set various Notification properties like its small and large icons, title, priority etc.

Step 2 - Setting Notification Properties

Once you have **Builder** object, you can set its Notification properties using Builder object as per your requirement. But this is mandatory to set at least following −

- A small icon, set by **setSmallIcon()**
- A title, set by **setContentTitle()**
- Detail text, set by

setContentText() Step 3 - Attach Actions

This is an optional part and required if you want to attach an action with the notification. An action allows users to go directly from the notification to an **Activity** in your application, where they can look at one or more events or do further work.

Step 4 - Issue the notification

Finally, you pass the Notification object to the system by calling NotificationManager.notify()

to send your notification. Make sure you call **NotificationCompat.Builder.build()** method on

builder object before notifying it. This method combines all of the options that have been set and return a new **Notification** object.

Code:

```
package com.fernando.basicnotification

import android.app.Notification
import  android.app.NotificationChannel
import android.app.NotificationManager
import android.content.Context
import android.os.Build
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.core.app.NotificationCompat
import androidx.core.app.NotificationManagerCompat
import
com.fernando.basicnotification.databinding.ActivityMainBinding
import java.nio.channels.Channels

class MainActivity : AppCompatActivity() {
    private lateinit var bindViews: ActivityMainBinding
    private lateinit var notificationManager: NotificationManagerCompat
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        bindViews = ActivityMainBinding.inflate(layoutInflater)
        setContentView(bindViews.root)
        notificationManager = NotificationManagerCompat.from(this)

        bindViews.btnChannel1.setOnClickListener {
            val title = bindViews.edtTitle.text.toString()
            val message = bindViews.edtMessage.text.toString()

            val notification = NotificationCompat.Builder(this, App.CHANNEL_ID_01)
                .setSmallIcon(R.drawable.ic_baseline_ring_volume_24)
                .setContentTitle(title)
                .setContentText(message)
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setCategory(NotificationCompat.CATEGORY_MESSAGE)
                .build()
            notificationManager.apply {
                notify(1, notification)
            }
        }

        bindViews.btnChannel2.setOnClickListener {
            val title = bindViews.edtTitle.text.toString()
            val message = bindViews.edtMessage.text.toString()

            val notification = NotificationCompat.Builder(this, App.CHANNEL_ID_02)
                .setSmallIcon(R.drawable.ic_baseline_ring_volume_24)
                .setContentTitle(title)
                .setContentText(message)
```

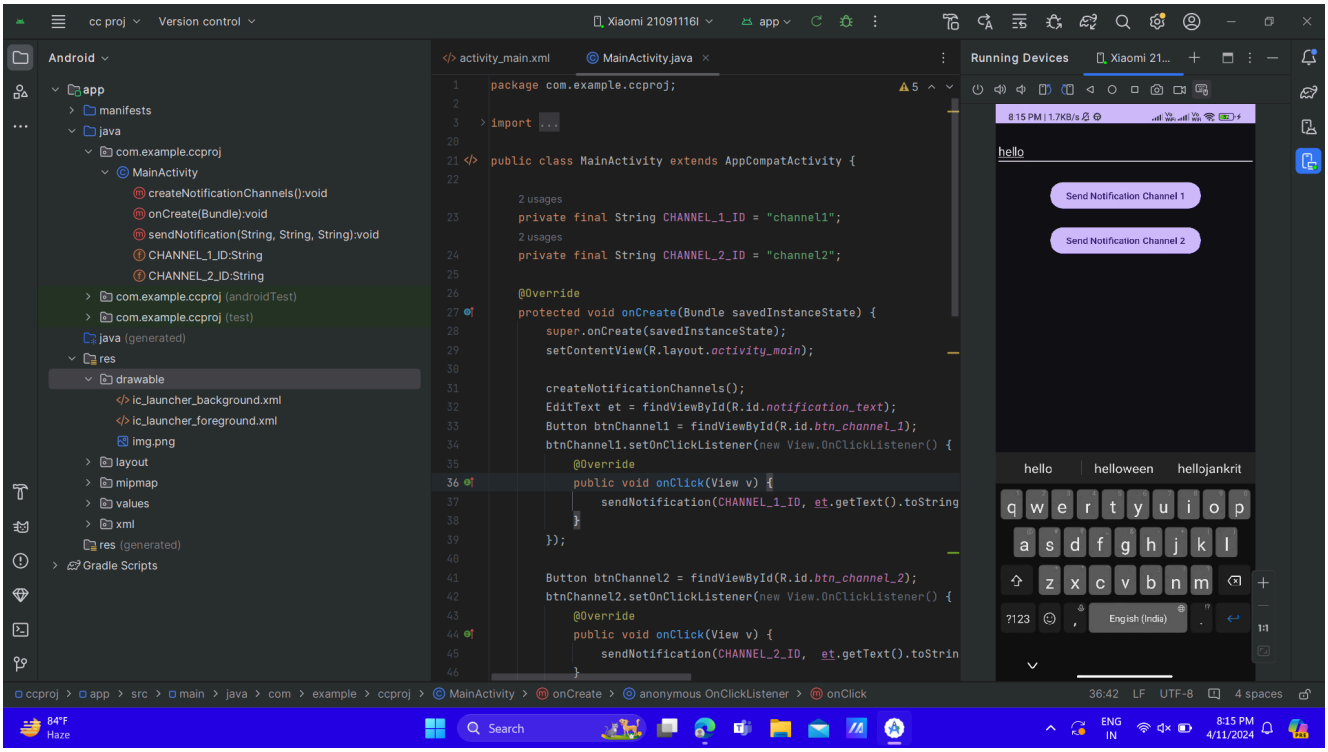
```
.setPriority(NotificationCompat.PRIORITY_LOW)
```

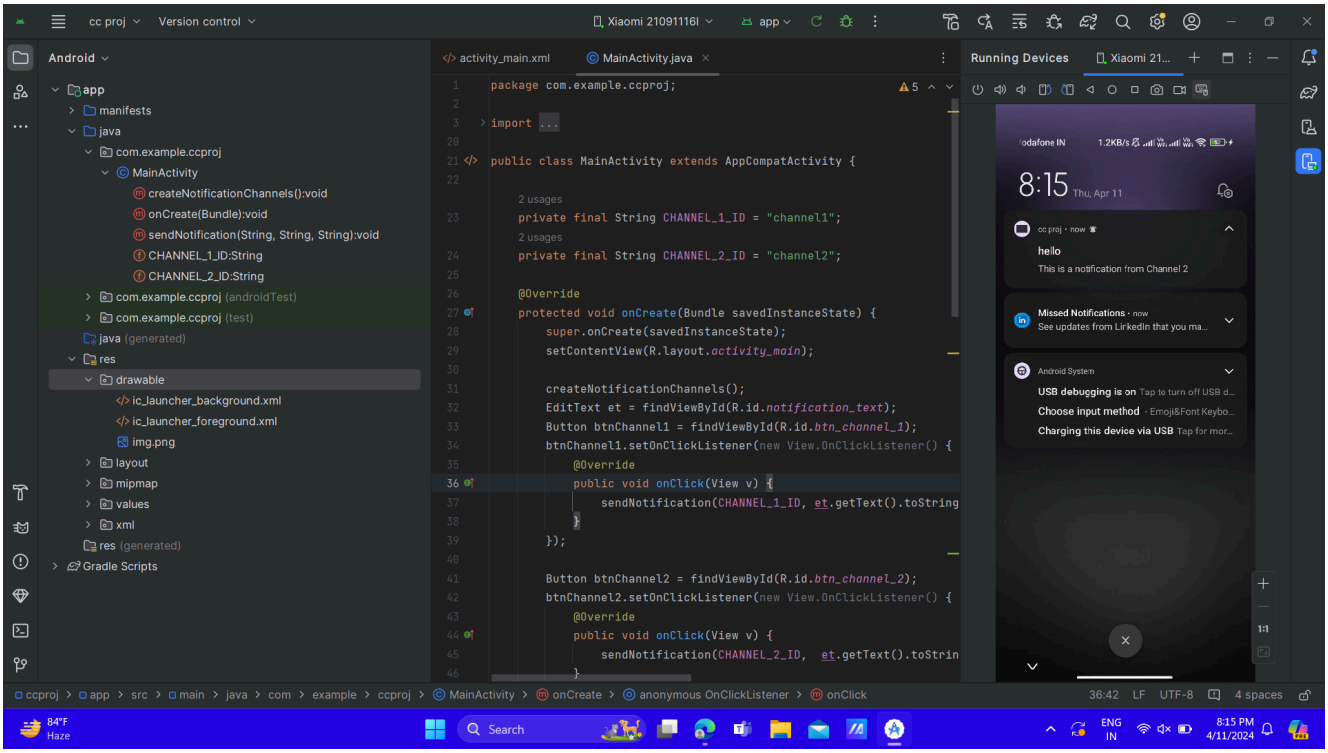
```
            .build()
        notificationManager.apply {
            notify(2, notification)
        }

    }
  }
}
package com.fernando.basicnotification

import android.app.Application
import  android.app.NotificationChannel
import android.app.NotificationManager
import android.content.Context
import     android.os.Build

class App: Application() {

    override fun onCreate() {
        super.onCreate()
        createNotificationChannel()
    }https://github.com/vinaynpp/mcc

    private fun createNotificationChannel() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val channel1 = NotificationChannel(
                CHANNEL_ID_01,
                "Canal 1",
                NotificationManager.IMPORTANCE_DEFAULT).apply {
                description = DESCRIPTION_CHANNEL1 }

            val channel2 = NotificationChannel(
                CHANNEL_ID_02,
                "Canal 2",
                NotificationManager.IMPORTANCE_LOW).apply {
                    description = DESCRIPTION_CHANNEL2
            }
            val manager = getSystemService(NotificationManager::class.java)
            manager.createNotificationChannels(listOf(channel1, channel2))
        }
    }
    companion object{
        const val CHANNEL_ID_01 = "channel1"
        const val CHANNEL_ID_02 = "channel2"
        const val CHANNEL_FERNANDO = "fernando"
        const val DESCRIPTION_CHANNEL1 = "Descrição do canal do Fernando 1"
        const val DESCRIPTION_CHANNEL2 = "Descrição do canal do Fernando 2"
    }
}


package com.fernando.basicnotification

import org.junit.Test
```

```
import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}
```

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

    <application
        android:name=".App"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.BasicNotification"
        >
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Output


cc proj
Version control
Xiaomi 21091116I
app
Android
app
manifests
java
com.example.ccproj
MainActivity
createNotificationChannels():void
onCreate(Bundle):void
sendNotification(String, String, String):void
CHANNEL_1_ID:String
CHANNEL_2_ID:String
com.example.ccproj (androidTest)
com.example.ccproj (test)
java (generated)
res
drawable
ic_launcher_background.xml
ic_launcher_foreground.xml
img.png
layout
mipmap
values
xml
res (generated)
Gradle Scripts
activity_main.xml
MainActivity.java
package com.example.ccproj;
import ...
public class MainActivity extends AppCompatActivity {
2 usages
private final String CHANNEL_1_ID = "channel1";
2 usages
private final String CHANNEL_2_ID = "channel2";
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
createNotificationChannels();
EditText et = findViewById(R.id.notification_text);
Button btnChannel1 = findViewById(R.id.btn_channel_1);
btnChannel1.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
sendNotification(CHANNEL_1_ID, et.getText().toString
}
});
Button btnChannel2 = findViewById(R.id.btn_channel_2);
btnChannel2.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
sendNotification(CHANNEL_2_ID, et.getText().toStrin
Running Devices
Xiaomi 21...
8:15 PM | 1.7KB/s
hello
Send Notification Channel 1
Send Notification Channel 2
hello
helloween
hellojankrit
English (India)
ccproj › app › src › main › java › com › example › ccproj › MainActivity › onCreate › anonymous OnClickListener › onClick
36:42
LF
UTF-8
4 spaces
84°F
Haze
Search
ENG
IN
8:15 PM
4/11/2024


cc proj
Version control
Xiaomi 21091116I
app
Android
app
manifests
java
com.example.ccproj
MainActivity
createNotificationChannels():void
onCreate(Bundle):void
sendNotification(String, String, String):void
CHANNEL_1_ID:String
CHANNEL_2_ID:String
com.example.ccproj (androidTest)
com.example.ccproj (test)
java (generated)
res
drawable
ic_launcher_background.xml
ic_launcher_foreground.xml
img.png
layout
mipmap
values
xml
res (generated)
Gradle Scripts
activity_main.xml
MainActivity.java
package com.example.ccproj;
import ...
public class MainActivity extends AppCompatActivity {
2 usages
private final String CHANNEL_1_ID = "channel1";
2 usages
private final String CHANNEL_2_ID = "channel2";
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
createNotificationChannels();
EditText et = findViewById(R.id.notification_text);
Button btnChannel1 = findViewById(R.id.btn_channel_1);
btnChannel1.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
sendNotification(CHANNEL_1_ID, et.getText().toString
}
});
Button btnChannel2 = findViewById(R.id.btn_channel_2);
btnChannel2.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
sendNotification(CHANNEL_2_ID, et.getText().toStrin
Running Devices
Xiaomi 21...
odafone IN
1.2KB/s
8:15
Thu, Apr 11
cc proj • now
hello
This is a notification from Channel 2
Missed Notifications • now
See updates from LinkedIn that you ma...
Android System
USB debugging is on Tap to turn off USB d...
Choose input method - Emoji&Font Keybo...
Charging this device via USB Tap for mor...
ccproj › app › src › main › java › com › example › ccproj › MainActivity › onCreate › anonymous OnClickListener › onClick
36:42
LF
UTF-8
4 spaces
84°F
Haze
Search
ENG
IN
8:15 PM
4/11/2024

Conclusion:

Thus, we have performed the experiment in which we have developed an app where a alert is created when a message is sent and we have executed it properly.

MC Assignment 1

Q1) Mobile communication & application environments are tailored to the unique demands of diverse sectors

1) Business:
- Enterprise Mobility sol: Facilitate remote work and access to business info

2) Location based services:
- Geolocation Applications: Utilize GPS for location-specific services
- Mapping & Nav: Provide real time direction & maps.

3) Banking services
- Mobile Banking Apps: Enable transactions, balance checks and fund transfers
- Mobile wallets: Support contactless payments and peer to peer transactions

Q2) 1g (1st gen): Analog Tech
features: * voice only: primarily support voice calls.
* low capacity
* limited international roaming capabilities
2g (Second gen): Digital Tech
features: * Digital voice Transmission
* moderate mobile data service
* 2g WLAN provided a high data rate
* Supported all 4 sectors
* Speed: 64 kbps

3g (3rd gen) : high speed digital tech

features * internet improved
* better system & capacity
* offers high speed
* connection used UMTS and HCMFI
* Spd 2mbps

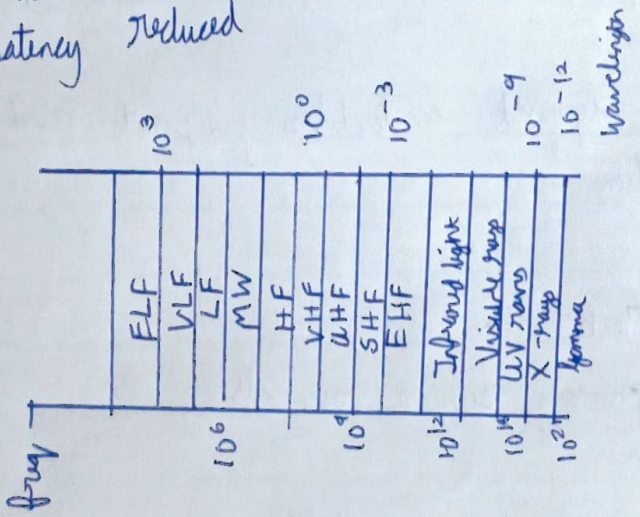
4g (4th gen): LTE (Long Term Evolution tech)

* High speed
* HD Quality
* IP based protocol
* High usability

5g (5th gen)

* Higher data rates
* connectivity will be fast
* 80 times faster
* massive network
* latency reduced

9.3)



① The electromagnetic spectrum defined as radio waves and microwaves is divided into 8 ranges, called band, each regulated by govt authorities. These bands are rated from very low frid to extremely high frequency. Radio transmission can take place using many diff freq bands. Each freq band exhibits certain adv & disadv

② Directly coupled to the freq is the wavelength λ via the fol $\lambda = c/f$

③ Components of the electromagnetic ~~field~~ spectrum

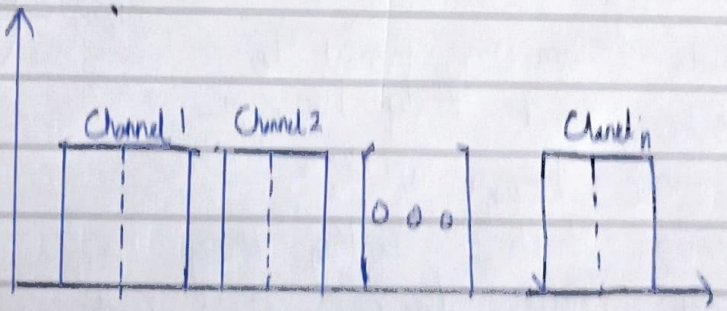
Radio waves, microwaves, infrared visible light, UV, X-ray, gamma ray

② Application in communication, medical imaging, remote sensing, tech applications application

Q.2) There are 3 types of multiplexing

a) i) Frequency division multiplexing: It is defined as the type where the bandwidth of a single physical medium is divided into a number of smaller, independent, frequency channel.

ii) One can observe a lot of inter-channel cross-talk due to the fact that in this type of multiplexing the bandwidth is divided into freq channels. In order to prevent the inter channel cross talk, unused strips of bandwidth must be placed between each channel are known as guard bands.



b) Time Division multiplexing: It is defined as the type where in FDM instead of sharing a portion of the bandwidth in the form of channels in TDM, time is shared.

① Synchronous TDM: assigns fixed synchronized time slots

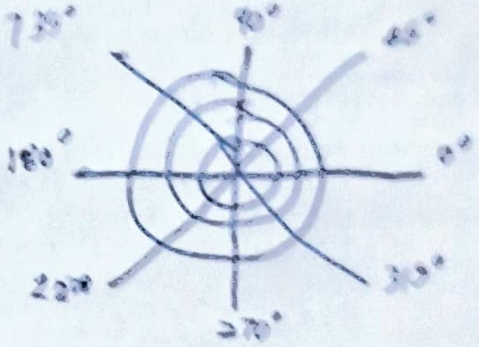
② Asynchronous TDM: Statistical time slots to signals for predictable transmission

c) Wavelength Division multiplexing - used on fiber optics to increase the capacity of a single fibre. It is an analog multiplexing technique ~~Optical~~ optical signals from the left source are combined to form a wider band of light

5.5) The types of antenna are as follows:

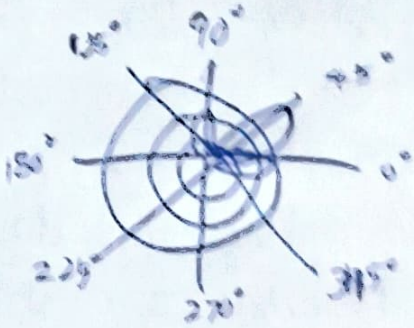
a) Omni Directional Antenna

These are provide considerable power signal in all direction covering 360° at an equal radial distance Simple design



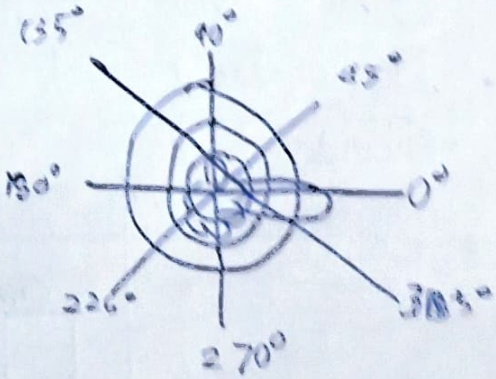
b) Semi Directional Antenna :-

These radiate the signal to a single, particular direction for comm. from 1 point to another. They can connect both the indoor and outdoor comm.



c) Highly Directional Antenna :-

These are used for directing signals from one point to another but they have a narrow beam with a high focus which is why they can reach much larger dist and are used in outdoor areas.



c) Highly Directed Antenna: These are used for directing signals from one point to another but they have a narrow beam with a high gain which is why they can reach much longer distances and are used in outdoor areas.

6) i) Spread spectrum is a communication technique that spreads the transmission of a single signal over a wide frequency band.

ii) This method enhances security, reduces interference and provides robustness against signal fading. It is commonly used in wireless communication systems, including WiFi and Bluetooth, to improve reliability and efficiency.

iii) Spread spectrum can be categorized into 2 main types: frequency hopping spread spectrum (FHSS) and direct sequence spread spectrum.

7) FHSS: In frequency hopping spread spectrum (FHSS), At one moment signal modulate ~~by~~ ~~as~~ one carrier frequency and at the subsequent moments, modulates after carrier frequencies.

DHSS: In DHSS, the bandwidth of the original signal is also expanded by a different technique The chip rate is n times the bit rate of the original signal. In wireless LAN, the sequence with n = 11 is used.

(8) Co-channel interference occurs when multiple transmitters share the same radio frequency channel simultaneously.

Causes:

- Overlap: When cells use the same frequency overlap

Effects on Communication:

- Increased noise: makes it challenging to distinguish the intended signal from unwanted signals.

Impact on Network Capacity

- Effective management: maximizes notebook capacity.

Frequency Reuse Patterns

- Different layout minimize inference

MC Assignment 2

Q1.

→ network and switching sub-system & GSM architechture

NSS stands for network and switching subsystem.

NSS core network of GSM.

carries out call and mobility management function for mobile phone in present network.

NSS have different component like VLR, HLR and EIR.

VLR

Visiter location Register.

database which contains exact location of all mobile subscriber currently present in area.

If you are going from one state to another then your entry marked in VLR.

HLR.

Home location register

Database containing date regarding subscribers authorised to use GSM network.

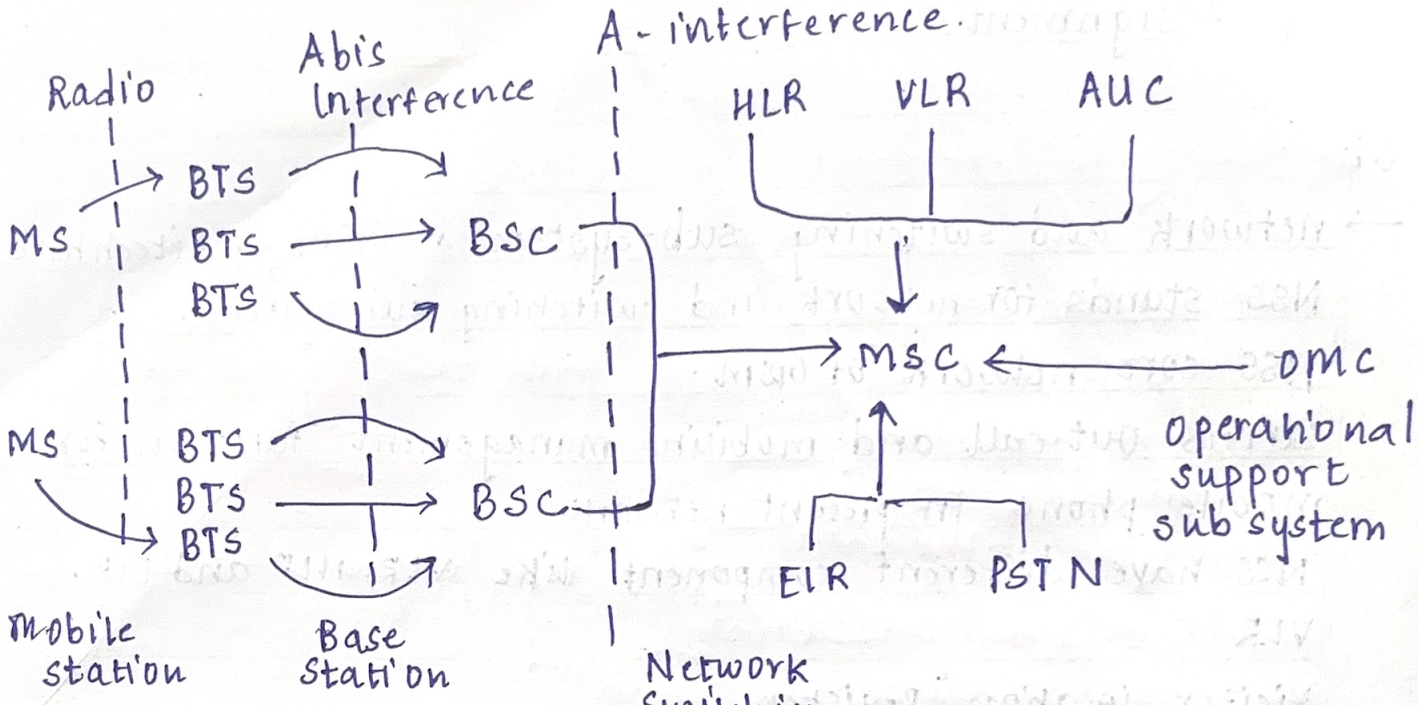
Its like a home which contains all data like your ID proof, which plan you are taking, which caller tune you are.

EIR

Equipment identity register

database that keeps record of all allowed or banned subscriber in the network.

If you are banned then you cannot enter the network nor make calls.



Q2] mobile originating call work.

→ subscriber initiates a call by dialing a number on the mobile.

→ it directly sends a request to the BTS which he comes under.

→ BTS then sends that request to BSC to which it's connected

→ from BSC request is made to the MSC to which it's connected.

→ Then MSC send it to HLR to check the info about the caller (account balance, area of caller etc)

→ if MSC is OK caller is allowed to make calls.

→ at the next instance call is disconnected.

→ if the MSC is not OK then MSC send it to BSC which then send it to BTS and BTS to mobile.

Q3.

→ when station calls a mobile station we can term it as MTC

→ User dials mobile no.

→ call is identified as Gsm and forwaed to GMSC

→ GMSC identifies HLR based on subscriber code.

→ HLR check for validity and subscription, request MSRN

→ HLR obtains MSRN from subscriber's current VLR.

→ MSRN is a temp no. for call routing and identity protection.

→ GMSC forwards call setup to msc responsible for mobile station.

→ MSC represent mobile station status from VLR.

→ Paging in all cells within location area to find ms.

→ MSC connects to ms using MSRN.

MSRN

Mobile station Roaming no.
Telephone no. used to route telephone calls in a mobile network.
also defined as temporary for no calling routing.
Assign for every mobile terminated call.

IMSI

International Mobile subscriber identity
Used to identify the user of a cellular network.

→ sent by phone to network.
→ used for network communication and analysis.
→ Also used for acquiring other details of the mobile in HLR.

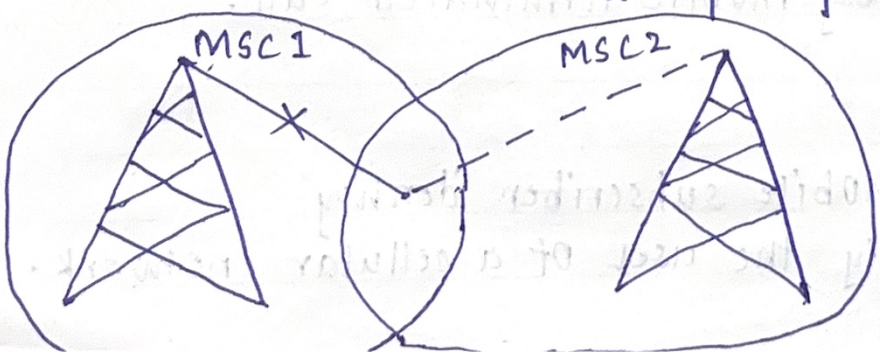
TMSI

→ Temporary mobile subscriber identity
→ Prohibits tracing of identity of mobile subscriber.
→ Assigned by the AUC (Authentication centre) with service area.
→ used as an alias for IMSI for security & connecting concerns.

Q4]
Handoff in GSM network.

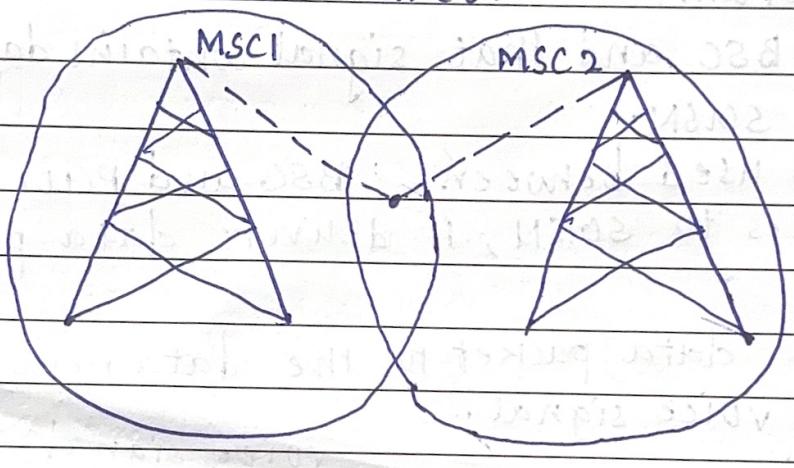
2 types: Hard & Soft:

- Hard Handoff
Occurs during the switch from one Base station & MSC
User hardly notice the break due to the speed of the switching process.
- ~~Soft handoff~~
- Connection quality is not that good
Adopted the break before make policy.



Soft Handoff.

(1) Atleast one of the links is kept when radio signals are added or removed to the base station.
(2) Adopted make before break policy
(3) more costly than Hard Handoff
(4) used in UMTS to improve signal quality
(5) its more seamless handover



Q5] Subscriber Authentication in GSM

→ GSM authenticates the identity of the subscriber through the use of challenge response architecture.
→ A 12 bit RAND (Random number) is sent to MS.
→ The MS computes the 32 bit SRES based on the encryption of RAND with the authentication algorithm using individual subscriber authentication.
→ GSM network repeats SRES calculation to verify subscriber identity upon receipt.
→ If recieved SRES matches calculated value, subscriber successfully authenticated.
→ signed response calculation occurs within the SIM

Q6]

→ GPRS architechture

& GPRS

General Packet Radio Service.

Modified version of GSM architechture service.

GSM has BSC but GPRS has added PUC to BSC

PCU → Packet control unit.

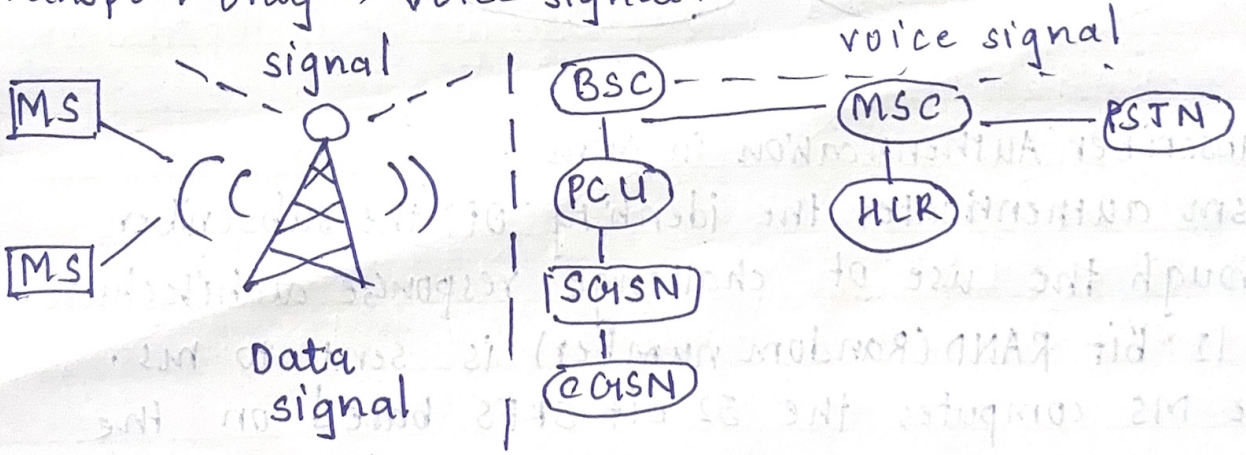
signal comes to BSC and that signal contains data, the PCU routes it to SGSN.

FRI interface is used between BSC and PCU.

After signal comes to SGSN, it delivers data packet to GGSN

GSN then routes data packet to the data network.

transport only ⇒ voice signal.



b) comparison

Aspects	GPRS	GSM
communication focus	Data centric communication	Voice centric communication
Network type	Packet-switched network	Circuit switched network

Data service	Optimized for packet switching data	Limited data capabilities.
Mobility	Efficient mobility for data services	Handover for voice call
components.	shared: MS, BSS, HLR, VLR	shared: MS, BSS, HLR, VLR.

7] GPRS channel.

a) PDCH

→ Packet Data channel.

→ Packet channel for transmitting packet-switched data in GPRS.

function: carrier user data such as inter traffic or application data.

b) PCH.

→ Paging channel

used for paging the mobile device when there is incoming data or a request for data transfer.

Alters mobile device to establish device connection.

ALC

→ Access link control.

→ messages the link between the mobile device & network.

Q6]

→ GPRS architechture

a GPRS

General Packet Radio Service.

Modified version of GSM architechture service.

GSM has BSC but GPRS has added PUC to BSC

PCU → Packet Control unit.

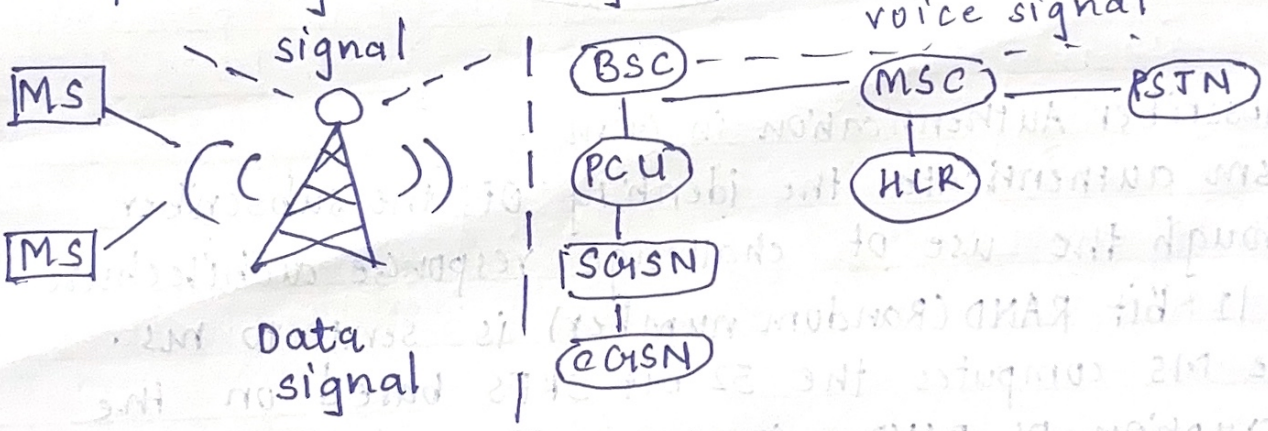
signal comes to BSC and that signal contains data, the PCU routes it to SGSN.

FRI interface is used between BSC and PCU.

After signal comes to SGSN, it delivers data packet to GGSN

GSN then routes data packet to the data network.

transport only ⇒ voice signal.



b) comparison

Aspects	GPRS	GSM
communication focus	Data centric communication	Voice centric communication
Network type	Packet-switched network	Circuit switched network

Data service	Optimized for packet switching data	Limited data capabilities.
Mobility	Efficient mobility for data services	Handover for voice call
components.	shared: MS, BSS, HLR, VLR	shared: MS, BSS, HLR, VLR.

7] GPRS channel.

a) PDCH

→ Packet Data channel.

→ Packet channel for transmitting packet-switched data in GPRS.

function: carrier user data such as inter traffic or application data.

b) PCH.

→ Paging channel

used for paging the mobile device when there is incoming data or a request for data transfer.

Alters mobile device to establish device connection.

ALC

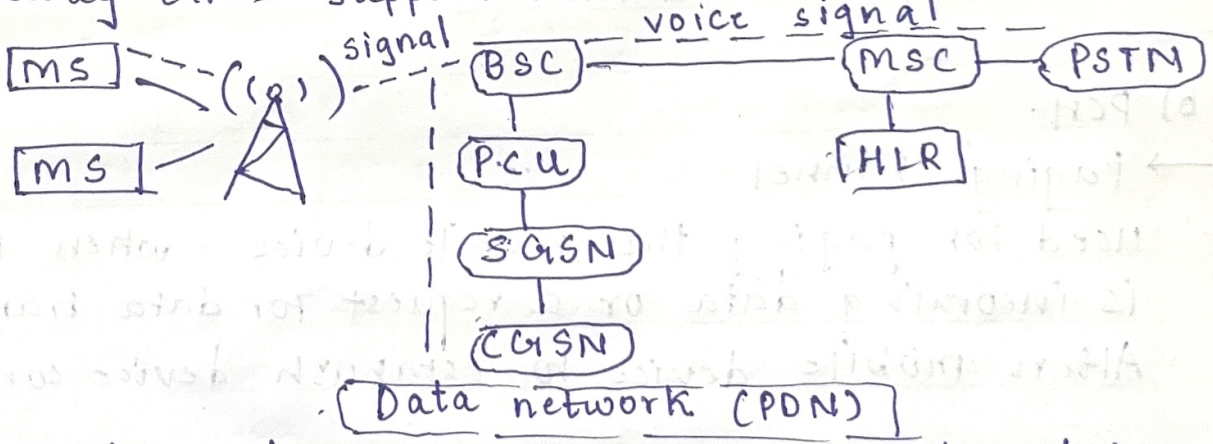
→ Access link control.

→ messages the link between the mobile device & network.

→ control access rights & permission for data transmission

d) CBCH
→ Cell broadcast channel
→ Used for delivering cell broadcast msg to multiple mobile device within a specific cell.
→ Broadcast info, such as news, emergency affairs etc

Q8] GPRS protocol architecture.
→ GPRS tries to make maximum use of GSM infrastructure has a new entity named GPRS support node (GSN) whose responsibility is to route and deliver data packets.
→ It has 2 types.
a) Securing GPRS support node (SGSN)
b) Gateway GPRS support node (GGSN)



→ GPRS requires enhanced mobile station for handling data packets.
→ These mobile stations are also capable of handling the GSM architecture to make call
→ BSC in GSM includes PCU in GPRS for data routing to SGSN via FR1 interface

SGSN handles packet delivery, mobility management authentication etc.

→ it also handles billing, logical link control management

→ GGSN mediates between GPRS and external data network handles participant data.

→ Internal backbone Network support GPRS with IP based packet exchange.

→ mobility support, includes attachment procedure location management and routing area.

→ Routing area are smaller than location area

→ Gsm introduced short messaging service (SMS) similar to peer to peer instant message.

09 UMTS

→ Universal mobile Telecommunication system.

→ 3G wireless communication technology

→ successor to Gsm family including GPRS & EDGE

→ utilize 5MHz wide channel for its radio interface

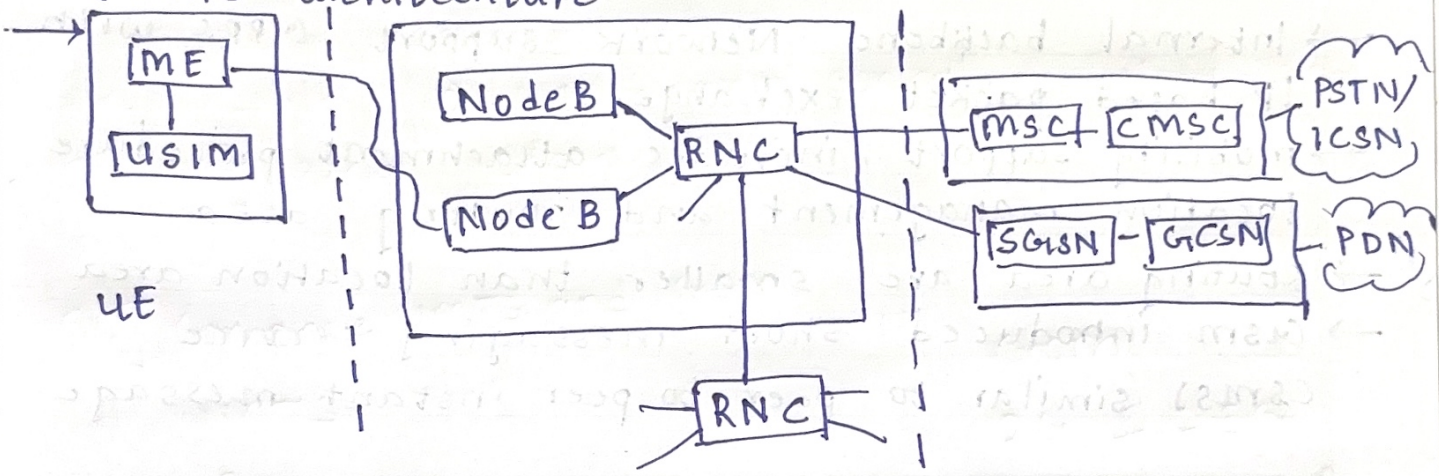
→ provide higher data rates & multimedia capability

→ Application: streaming
video conference
Fast Internet
Remote login
Email
FTP access

Advantages .support data rates upto 2 mbps allows smooth migration from 2G to 3G flexible for operators to introduce new service to attract customer.

Disadvantages- High cost compared to Gsm poor video experience not yet considered broadband.

Q10] UMTS architechture



→ UTMS is designed to support high speed data & technologies multimedia service, offering improvement over earlier

→ UE (user Equipment)
Mobile device like smartphone, table, etc, including the physical device & sim card.

→ UTMS terrestorial Radio Access Network (UTRAN)
manages radio resources & connect mobile device to the core network.

→ UTMS core network (CN)
handles network functionality and subscriber service.

ii) components: SGSN (pocket switching service) GGSN gateway to external networks), MSC, HLR, AUC.

GSM interworking
facilitates interoperability with GSM network for seamless network
Multimedia domain
Manages multimedia session like video calling & multimedia messaging.
Dynamic resource allocation.
Allows flexible management of radio resources used on traffic load and user demand.

Q11.

→ UMTS air interface.

a) UE

→ User equipment

→ refers to mobile device or handset used by end user.

b) Utran

→ comprises of 2 main components.
NodeB: base station that communicate directly with the mobile device
responsible for radio transmission & reception.
Radio Network Controller - manage multimedia Node Bs, control handover between them
responsible for radio source allocation.

c) AAA

→ Authentication, Authorization & accounting. responsible for verifying users identity, authorizing network access, managing build info.

d) MME

→ mobile management entity responsible for significant control, mobility and tracking area updates.

Q12] Compare UMTS and GSM

UMTS	GSM
3G	2G & 2.5G
mainly CSMA based	Typically based on TDMA.
frame duration → 10ms	frame duration → 4.165ms
its new	its old technology
modulation efficiency → 1.0 bs/Hz	1.35 b/s/Hz

- MC - ASSIGNMENT 3

Summary Of IEEE 802.11 : Wireless LANs From A To N

* Historical Development

- originated to meet the need for wireless LAN connectivity
- initial focus on developing MAC protocol and defining physical layer
- challenges led to adoption of CSMA/CA over token-passing protocol
- formation of IEEE 802.11 working group in 1990 was a milestone.

* Protocol Architecture

- LLC Layer : interface b/w higher layers and MAC. Perform error control, flow control and frame delimiting. Ensures correct formatting & transmission of data frames
- MAC Layer : Controls access to wireless medium, defines frame structures and governs transmission rules. Includes DCF and PCF sublayers for contention-based & free access
- Physical Layer : Defines radio transmission characteristics, modulation schemes and data rates. Specifies how data is modulated onto carrier signals for wireless transmission

* Physical Standards (Layer)

- DSSS : operates at 2.4 GHz, provides 1Mbps & 2Mbps data rates, spreads signals over wider bandwidth for interference reduction
- FHSS : operates at 2.4 GHz, hops b/w freq. rapidly to remove interference, channels vary by regulatory domain
- Infrared : operates at 1-2 Mbps, requires line-of-sight communication, limited by data rates and line-of-sight requirements.

* IEEE 802.11 STANDARDS :

- 802.11a : uses 5 GHz band, OFDM modulation, data rates up to 54 Mbps per channel
- 802.11b : extends DSSS, offers 5.5 and 11 Mbps data rates, backward compatible with existing networks.
- 802.11g : extends 802.11b with data rates from 12 to 54 Mbps, supports DSSS and OFDM.
- 802.11c : defines bridge operation for LAN interconnection
- 802.11d : addresses regulatory differences for WLAN operation in various countries
- 802.11e : enhances MAC layer for QoS improvement
- 802.11f : ensures interoperability among access points from different vendors.
- 802.11h : enhances 802.11a for compliance with European regulations.
- 802.11i : enhances MAC layer security with stronger encryption and authentication

* WLAN SECURITY

- Wired Equivalent Privacy (WEP) : originally part of the 802.11 standard, WEP employs RC4 encryption with 40-bit or 104-bit keys but suffers from vulnerbilities like key reuse and weak initialization vectors.
- Wi-Fi Protected Access (WPA) : introduced as an interim solution based on 802.11i draft standards, WPA offers enhancements over WEP, including TKIP encryption and stronger key management.
- IEEE 802.11i : defines robust security mechanisms such as AES encryption, robust authentication protocols and secure key distribution, addressing WEP weaknesses and ensuring comprehensive security for WLANs.