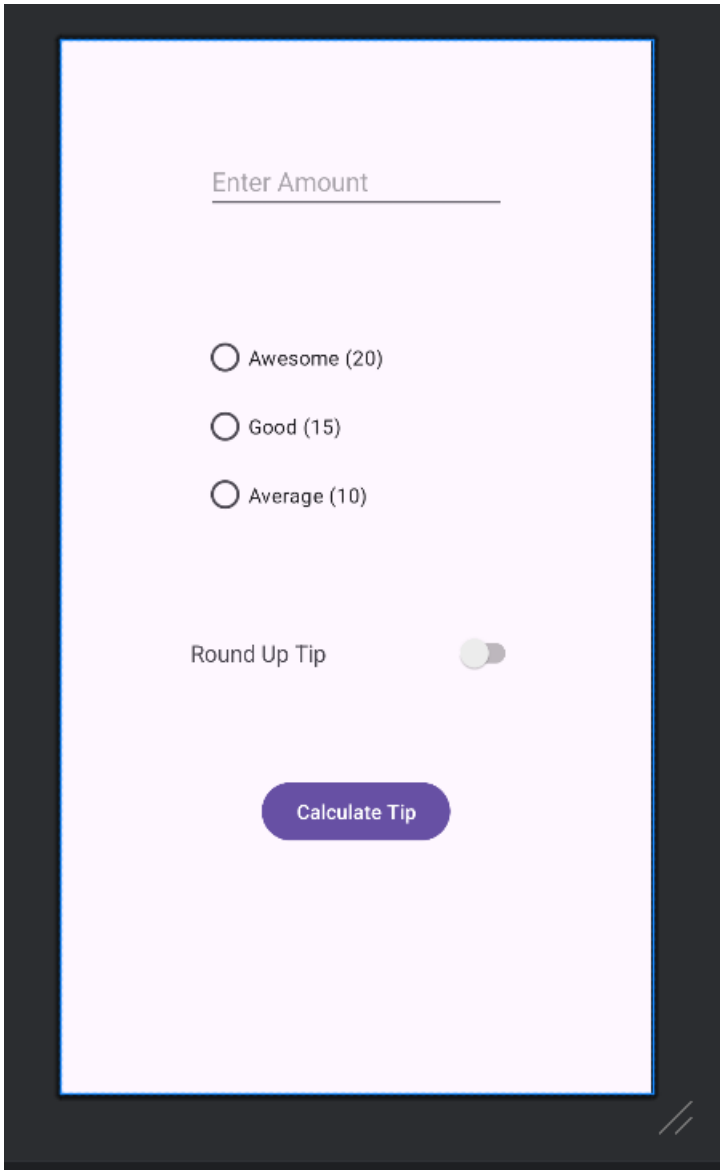
TIP CALCULATOR

```
package com.example.toastmsg

import android.annotation.SuppressLint
import android.os.Bundle
import android.widget.Button
import android.widget.ImageView
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import com.example.toastmsg.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {
    lateinit var binding: ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.tipBtn.setOnClickListener() {
            calculateTip()
        }
    }
    fun calculateTip()
    {
        val cost=(binding.editTextText.text.toString()).toDouble()
        val selected=binding.radioGroup.checkedRadioButtonId
        val tipPercent=when(selected)
        {
            R.id.radioButton2 -> 0.15
            R.id.radioButton3 -> 0.10
            else -> 0.20
        }
        var tip=tipPercent*cost
        if(binding.switch1.isChecked)
        {
            tip=kotlin.math.ceil(tip)
        }
        binding.textView3.text=tip.toString()
    }
}
```

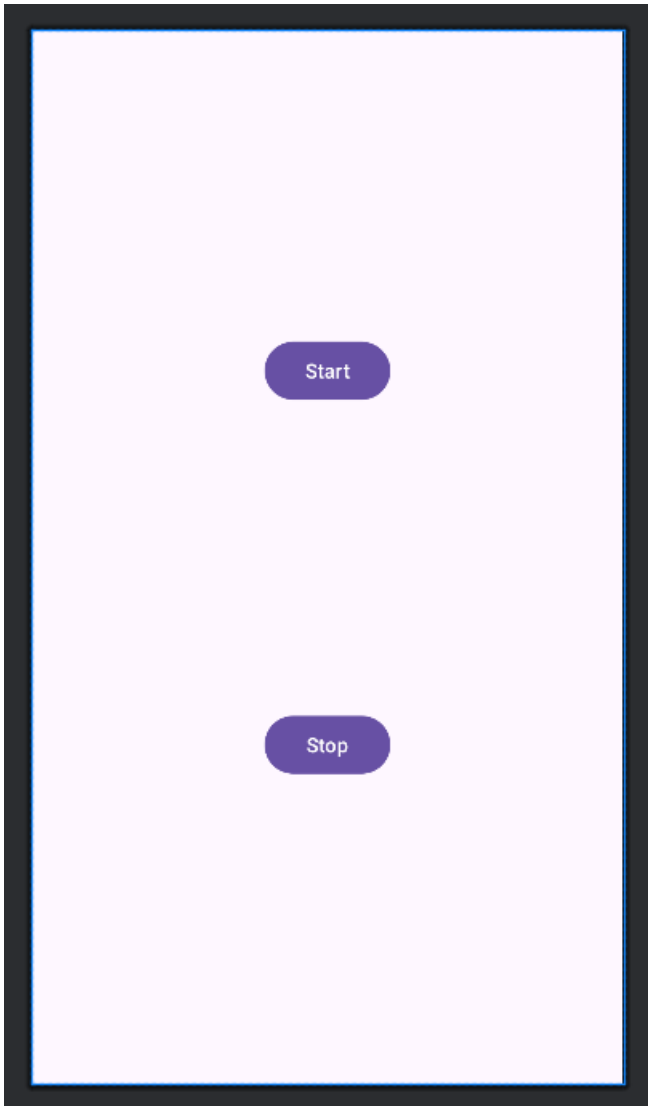
In build.gradle (module)

```
buildFeatures{
    viewBinding=true
}
```



START STOP TOAST

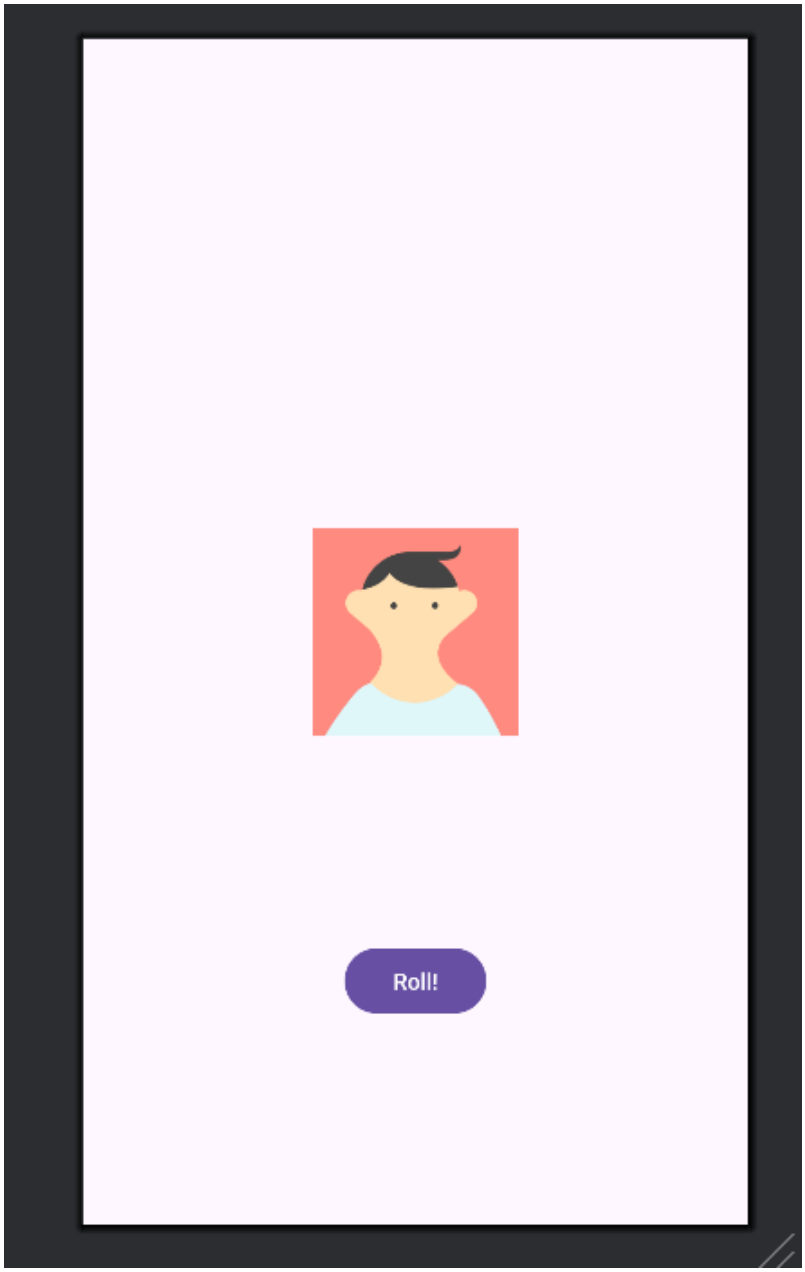
```
package com.example.toastmsg

import android.annotation.SuppressLint
import android.os.Bundle
import android.widget.Button
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class MainActivity : AppCompatActivity() {
    @SuppressLint("MissingInflatedId")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        var b1: Button = findViewById(R.id.button)
        var b2: Button = findViewById(R.id.button2)
        b1.setOnClickListener(){
            Toast.makeText(this, "Start", Toast.LENGTH_SHORT).show()
        }
        b2.setOnClickListener(){
            Toast.makeText(this, "Stop!!", Toast.LENGTH_SHORT).show()
        }
    }
}
```


Start
Stop

DICE ROLLER

```
package com.example.app

import android.os.Bundle
import android.widget.Button
import android.widget.ImageView
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        val imgv: ImageView = findViewById(R.id.imageView2)
        imgv.setImageResource(R.drawable.dice)
        var btn: Button = findViewById(R.id.button4)
        btn.setOnClickListener{
            diceRoller()
        }
    }
    private fun diceRoller(){
        val diceIns = Dice(6)
        val diceRollNumber = diceIns.roll()
        val imgv: ImageView = findViewById(R.id.imageView2)

        val res = when(diceRollNumber){
            1 -> R.drawable.one
            2 -> R.drawable.two
            3 -> R.drawable.three
            4 -> R.drawable.four
            5 -> R.drawable.five
            6 -> R.drawable.six
            else -> R.drawable.one
        }
        imgv.setImageResource(res)
    }
    class Dice(private val n:Int){
        fun roll() : Int{
            return (1..n).random();
        }
```

```
    }
}
```


Roll!

WELCOME TOAST

```
package com.example.toastmsg

import android.annotation.SuppressLint
import android.os.Bundle
import android.widget.Button
import android.widget.ImageView
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class MainActivity : AppCompatActivity() {
    @SuppressLint("MissingInflatedId")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        var btn:Button = findViewById(R.id.button5)
        btn.setOnClickListener{
            Toast.makeText(this, "Welcome Message", Toast.LENGTH_LONG).show()
        }
    }
}
```

ACTIVITY LIFE CYCLE

```
package com.example.madpractice

import android.os.Bundle
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        Toast.makeText(this, "onCreate()", Toast.LENGTH_SHORT).show()
    }

    override fun onStart() {
        super.onStart()
        Toast.makeText(this, "onStart()", Toast.LENGTH_SHORT).show()
    }

    override fun onRestart() {
        super.onRestart()
        Toast.makeText(this, "onRestart()", Toast.LENGTH_SHORT).show()
    }

    override fun onPause() {
        super.onPause()
        Toast.makeText(this, "onPause()", Toast.LENGTH_SHORT).show()
    }

    override fun onStop() {
        super.onStop()
        Toast.makeText(this, "onStop()", Toast.LENGTH_SHORT).show()
    }

    override fun onDestroy() {
        super.onDestroy()
        Toast.makeText(this, "onDestroy()", Toast.LENGTH_SHORT).show()
    }
}
```

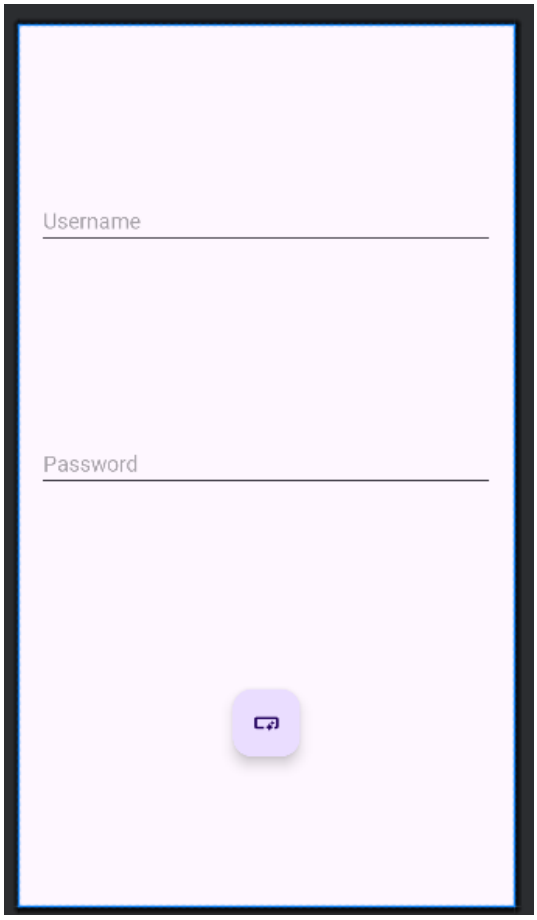
Navigation using intent (explicit intent)

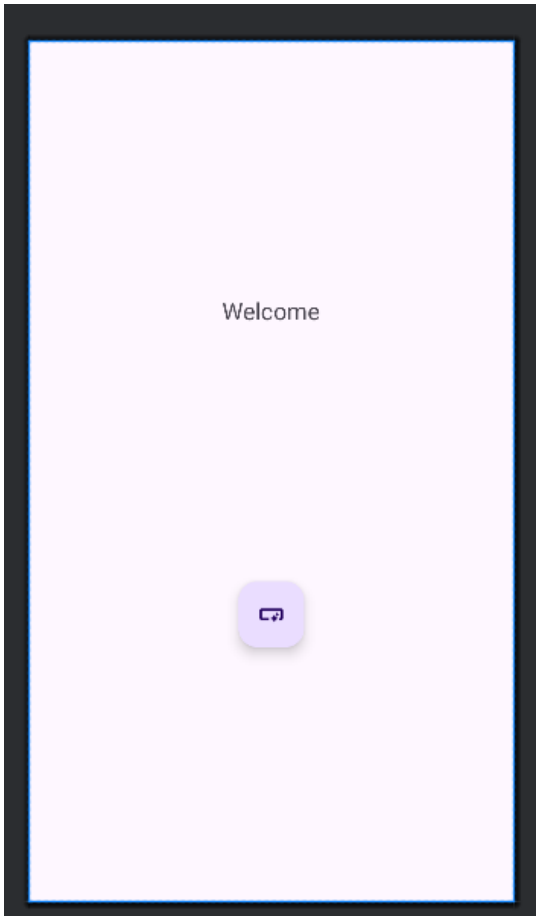
ActivityMain.kt

```
package com.example.madpractice

import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import com.google.android.material.floatingactionbutton.FloatingActionButton

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        var e1:EditText = findViewById(R.id.editText1)
        var f:FloatingActionButton = findViewById(R.id.floatingActionButton)
        f.setOnClickListener{
            var i = Intent(this, MainActivity2::class.java)
            startActivity(i)
        }
    }
}
```

ActivityMain2.kt

```
class MainActivity2 : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main2)
        var t:TextView = findViewById(R.id.textView)
        var f: FloatingActionButton = findViewById(R.id.floatingActionButton2)
        f.setOnClickListener{
            var i = Intent(this, MainActivity::class.java)
            startActivity(i)
        }
    }
}
```


Username
Password


Welcome

LOGIN USING EXPLICIT INTENT

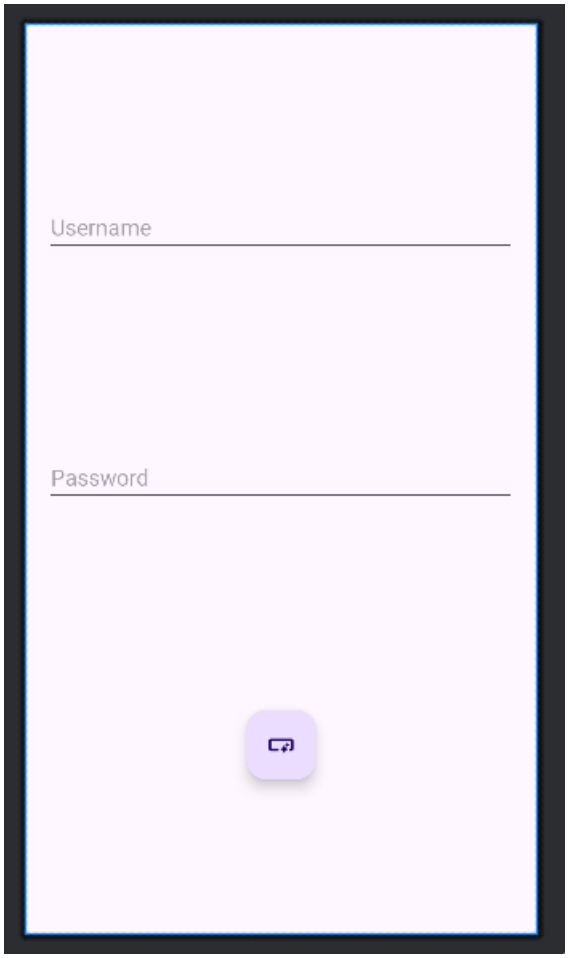
MainActivity.kt

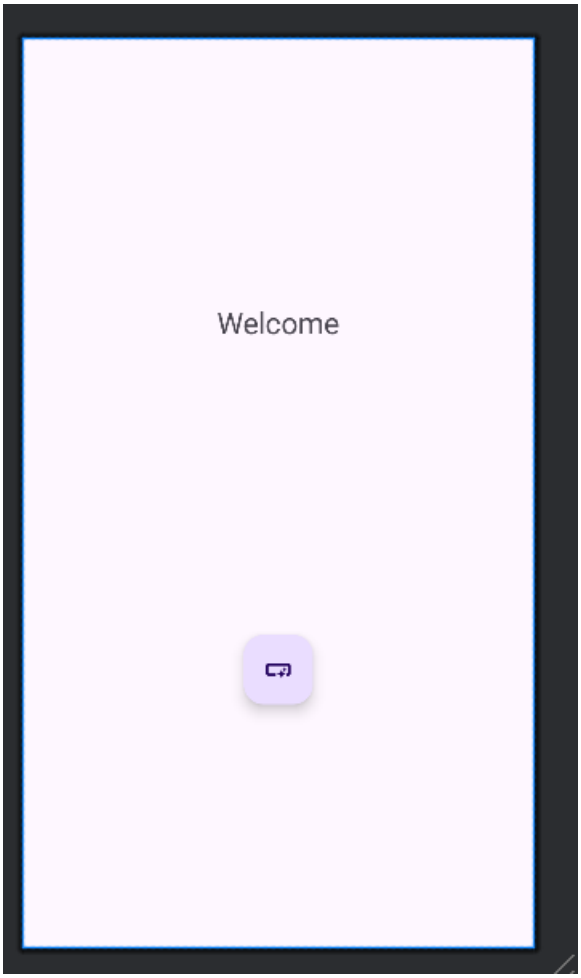
```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        var un:EditText = findViewById(R.id.editText1)
        var pw:EditText = findViewById(R.id.editText2)
        var b:FloatingActionButton = findViewById(R.id.floatingActionButton)
        b.setOnClickListener{
            val user=un.text.toString()
            val pass=pw.text.toString()
            if(user=="user" && pass=="password") {
                var i = Intent(this, MainActivity2::class.java)
                i.putExtra("USERNAME", user)
                startActivity(i)
            } else{
                Toast.makeText(this, "Invalid Login", Toast.LENGTH_SHORT). show()
            }
        }
    }
}
```

MainActivity2.kt

```
class MainActivity2 : AppCompatActivity() {
    @SuppressLint("MissingInflatedId")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main2)
        var t:TextView = findViewById(R.id.textView)
        var f: FloatingActionButton = findViewById(R.id.floatingActionButton2)
        val un=intent.getStringExtra("USERNAME")
        t.text="Welcome, $un !"
        f.setOnClickListener{
            var i = Intent(this, MainActivity::class.java)
            startActivity(i)
        }
    }
}
```

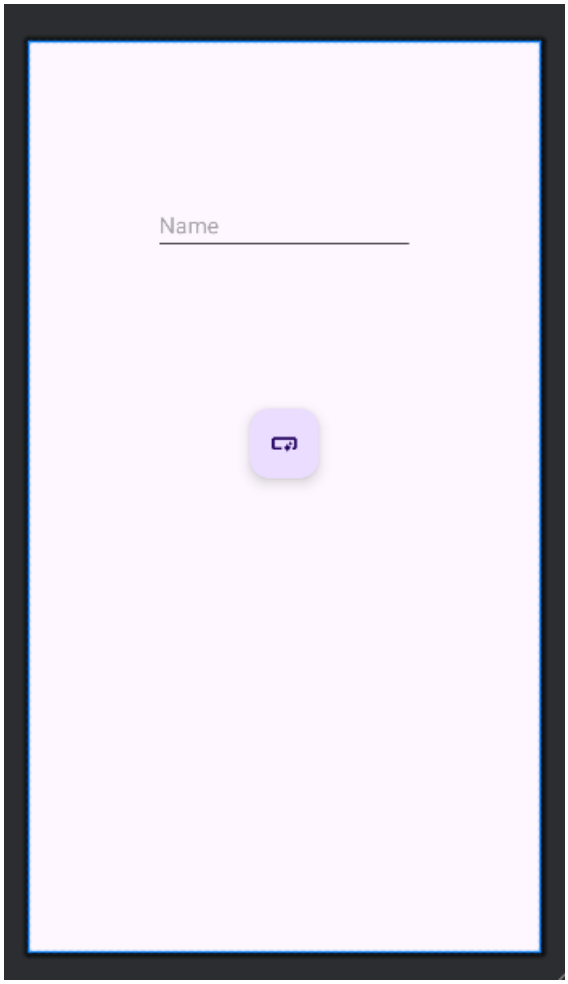
android:inputType="textPassword" → in activity_main.xml of password edit text


Username
Password


Welcome

Implicit Intent app

```
class MainActivity : AppCompatActivity() {
    @SuppressLint("MissingInflatedId")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        var e:EditText = findViewById(R.id.editTextText4)
        var b:FloatingActionButton = findViewById(R.id.floatingActionButton)
        b.setOnClickListener{
            var url:String = e.getText().toString()
            var i = Intent(Intent.ACTION_VIEW, Uri.parse(url))
            startActivity(i)
        }
    }
}
```



KOTLIN PROGRAMS

Write a Kotlin program that takes a nullable integer as input and print its Square if it is not null, or “Input is null” otherwise

```
fun main() {
    val number: Int? = readLine()?.toIntOrNull()
    if (number != null) {
        println("Square of $number is ${number * number}")
    } else {
        println("Input is null")
    }
}
```

Implement a number guessing game in which the user is prompted to enter a number between 1 and 100 until he or she guesses correctly. After every wrong guess, the user is told whether the guess was too high or too low.

```
import kotlin.random.Random
fun main() {
    val secretNumber = Random.nextInt(1, 11)
    println("Welcome to the Number Guessing Game!")
    println("Guess a number between 1 and 10:")
    do {
        var guess: Int? = readLine()?.toIntOrNull()
        when{
            guess == null -> println("Please enter a valid number.")
            guess < secretNumber -> println("Too low! Try again:")
            guess > secretNumber -> println("Too high! Try again:")
            else -> println("Congratulations! You guessed the correct number:$secretNumber")
        }
    } while (guess != secretNumber)
}
```

Create a function in Kotlin that takes a name as input and prints the greeting message. Make the message customizable and provide a default message if no custom message is provided.

```
fun main(){
    greetUser("renu")
    greetUser("renu", "Welcome to MAD Lab")
}
fun greetUser(name: String, msg: String = "Hello"){
    println("$msg, $name")
}
```

Hello, renu

Welcome to MAD Lab, renu

getter and setter properties

```
fun main() {
    val c = CSE()
    c.name = "WELCOME TO CSE-B" // access setter
    println(c.name)             // access getter
}
class CSE {
    var name: String = " "
    get() = field
    set(value) {
        field = value
    }
}
```

WELCOME TO CSE-B

companion object

```
class CSE {
```

```
    companion object Test  {
    //companion object name Test
            fun section_b() = println("WELCOME TO CSE-B MAD LAB")
            }
}
fun main( ) {
            CSE.section_b()   // method accessing using class name
}
```

WELCOME TO CSE-B MAD LAB

Kotlin program that defines a companion object with two variables and two methods, which can all be accessed using the class name

```
class Calculator {
    companion object {
        // Variables inside the companion object
        val pi = 3.14159
        val euler = 2.71828
        // Methods inside the companion object
        fun add(a: Int, b: Int): Int {
            return a + b
        }
        fun subtract(a: Int, b: Int): Int {
            return a - b
        }
    }
}
fun main() {
    // Accessing variables using the class name
    println("Pi: ${Calculator.pi}")
    println("Euler's Number: ${Calculator.euler}")
```

```
    // Accessing methods using the class name
    val sum = Calculator.add(10, 5)
    val difference = Calculator.subtract(10, 5)
    println("Sum: $sum")
    println("Difference: $difference")
}
```

```
Pi: 3.14159
Euler's Number: 2.71828
Sum: 15
Difference: 5
```

DICE ROLLER PROGRAM USING CLASSES

```
import kotlin.random.Random
class Dice {
    var sides = 6
    fun roll(): Int {
        val randomNumber = (1..sides).random()
        return randomNumber
    }
}
fun main() {
    val d = Dice()
    val diceRoll = d.roll()
    println("Your ${d.sides} sided dice rolled ${diceRoll}!")
    d.sides = 20
    println("Your ${d.sides} sided dice rolled ${d.roll()}!")
}
```

```
Your 6 sided dice rolled 3!
Your 20 sided dice rolled 10!
```

primary constructor

```
fun main() {
    val add = Add(5, 6)
    println("The Sum of numbers 5 and 6 is: ${add.c}")
}
class Add constructor(a: Int,b:Int) {
        var c = a+b;
}
```

The Sum of numbers 5 and 6 is: 11

secondary constructor

```
fun main() {
        Add(5, 6)
}
class Add{
    constructor(a: Int, b:Int) {
        var c = a + b
        println("The sum of numbers 5 and 6 is: ${c}")
    }
}
```

The sum of numbers 5 and 6 is: 11

Data Types and Constructors Program

```
fun main() {
    val age: Int = 25
    val name: String = "Alice"
    val height: Float = 5.6f
    val isStudent: Boolean = true
```

```
    println("Name: $name, Age: $age, Height: $height, Is Student: $isStudent")
}
```

Name: Alice, Age: 25, Height: 5.6, Is Student: true

Loop Concept and arrayOf Keyword Usage

```
fun main() {
    val numbers = arrayOf(1, 2, 3, 4, 5)
    for (number in numbers) {
        println("Number: $number")
    }
}
```

Number: 1

Number: 2

Number: 3

Number: 4

Number: 5

Demonstrate Arithmetic Operators

```
fun main() {
    val a = 10
    val b = 5
    println("Addition: ${a + b}")      // 15
    println("Subtraction: ${a - b}")   // 5
    println("Multiplication: ${a * b}") // 50
    println("Division: ${a / b}")      // 2
    println("Modulus: ${a % b}")       // 0
}
```

Addition: 15

Subtraction: 5

Multiplication: 50

Division: 2

Modulus: 0

Companion Object

```
class Example {
    companion object {
        fun greet() {
            println("Hello from the companion object!")
        }
    }
}
fun main() {
    Example.greet()
}
```

Hello from the companion object!

val / var

```
fun main() {
    val immutableValue: Int = 10 // Read-only
    var mutableValue: Int = 20    // Mutable
    // immutableValue = 15 // This will cause an error
    mutableValue = 25 // This is allowed
    println("Immutable Value: $immutableValue")
    println("Mutable Value: $mutableValue")
}
```

Immutable Value: 10

Mutable Value: 25

Variable and Invoke Companion Object Init

```
class Person(val name: String) {
    companion object {
        val defaultName = "John Doe"
        fun createDefaultPerson(): Person {
            return Person(defaultName)
        }
    }
}
fun main() {
    val person = Person.createDefaultPerson()
    println("Person's name: ${person.name}")
}
```

Person's name: John Doe

Anonymous Function

```
fun main() {
    val sum = fun(a: Int, b: Int): Int {
        return a + b
    }
    println("Sum: ${sum(5, 3)}") // Output: 8
}
```

Sum: 8

Data Types Program

```
fun main() {
    val integer: Int = 100
    val decimal: Double = 10.5
    val character: Char = 'K'
    val boolean: Boolean = true
    println("Integer: $integer")
    println("Double: $decimal")
    println("Character: $character")
    println("Boolean: $boolean")
}
```

```
Integer: 100
Double: 10.5
Character: K
Boolean: true
```

arrayOf Function

```
fun main() {
    val fruits = arrayOf("Apple", "Banana", "Cherry")
    for (fruit in fruits) {
        println("Fruit: $fruit")
    }
}
```

```
Fruit: Apple
Fruit: Banana
Fruit: Cherry
```

Dwellings

```
import kotlin.math.*
```

```
fun main() {
    val squareCabin = SquareCabin(6, 50.0)
    val roundHut = RoundHut(3, 10.0)
    val roundTower = RoundTower(4, 15.5)

    with(squareCabin) {
        println("\nSquare Cabin\n============")
        println("Capacity: ${capacity}")
        println("Material: ${buildingMaterial}")
        println("Floor area: ${floorArea()}")
    }

    with(roundHut) {
        println("\nRound Hut\n=========")
        println("Material: ${buildingMaterial}")
        println("Capacity: ${capacity}")
        println("Floor area: ${floorArea()}")
        println("Has room? ${hasRoom()}")
        getRoom()
        println("Has room? ${hasRoom()}")
        getRoom()
        println("Carpet size: ${calculateMaxCarpetLength()}")
    }

    with(roundTower) {
        println("\nRound Tower\n==========")
        println("Material: ${buildingMaterial}")
        println("Capacity: ${capacity}")
        println("Floor area: ${floorArea()}")
        println("Carpet Length: ${calculateMaxCarpetLength()}")
```

```
    }
}
abstract class Dwelling(private var residents: Int) {
    abstract val buildingMaterial: String
    abstract val capacity: Int
    abstract fun floorArea(): Double
    fun hasRoom(): Boolean {
        return residents < capacity
    }
    fun getRoom() {
        if (capacity > residents) {
            residents++
            println("You got a room!")
        } else {
            println("Sorry, at capacity and no rooms left.")
        }
    }
}
class SquareCabin(residents: Int, val length: Double) : Dwelling(residents) {
    override val buildingMaterial = "Wood"
    override val capacity = 6
    override fun floorArea(): Double {
        return length * length
    }
}
open class RoundHut(residents: Int, val radius: Double) : Dwelling(residents) {
    override val buildingMaterial = "Straw"
    override val capacity = 4
    override fun floorArea(): Double {
        return PI * radius * radius
```

```
    }
    fun calculateMaxCarpetLength(): Double {
        return sqrt(2.0) * radius
    }
}
class RoundTower(residents: Int, radius: Double, val floors: Int = 2) : RoundHut(residents, radius) {
    override val buildingMaterial = "Stone"
    override val capacity = floors * 4
    override fun floorArea(): Double {
        return super.floorArea() * floors
    }
}
```

```
Square Cabin
============
Capacity: 6
Material: Wood
Floor area: 2500.0

Round Hut
=========
Material: Straw
Capacity: 4
Floor area: 314.1592653589793
Has room? true
You got a room!
Has room? false
Sorry, at capacity and no rooms left.
Carpet size: 14.142135623730951
```

Round Tower

==========

Material: Stone

Capacity: 8

Floor area: 1509.5352700498956

Carpet Length: 21.920310216782976

INIT BLOCK

```
class InitOrderDemo(name: String) {
    println("First property: $name")
    init {
        println("First initializer block that prints $name")
    }
    println("Second property: ${name.length}")
    init {
        println("Second initializer block that prints ${name.length}")
    }
}
fun main() {
    InitOrderDemo("hello")
}
```

```
First property: hello
First initializer block that prints hello
Second property: 5
Second initializer block that prints 5
```

this keyword and Constructor Declaration of Class

```
class employee {
```

```
    var name: String = ""
    var age: Int = 0
    var gender: Char = 'M'
    var salary: Double = 0.toDouble()
    fun insertValues(n: String, a: Int, g: Char, s: Double) {
        name = n
        age = a
        gender = g
        salary = s
        println("Name of the employee: $name")
        println("Age of the employee: $age")
        println("Gender: $gender")
        println("Salary of the employee: $salary")
    }
    fun insertName(n: String) {
            this.name = n
    }
}
fun main(args: Array<String>) {
    var obj = employee()
    var obj2 = employee()
    obj.insertValues("X1", 50, 'M', 500000.00)
    obj2.insertName("X2")
    println("Name of the new employee: ${obj2.name}")
}
```

```
Name of the employee: X1
Age of the employee: 50
Gender: M
Salary of the employee: 500000.0
```

Name of the new employee: X2