Name: Madepelli Sadhana

Student ID: N01651380

Date: 17/1/25

Subject: Android Development Lab-2

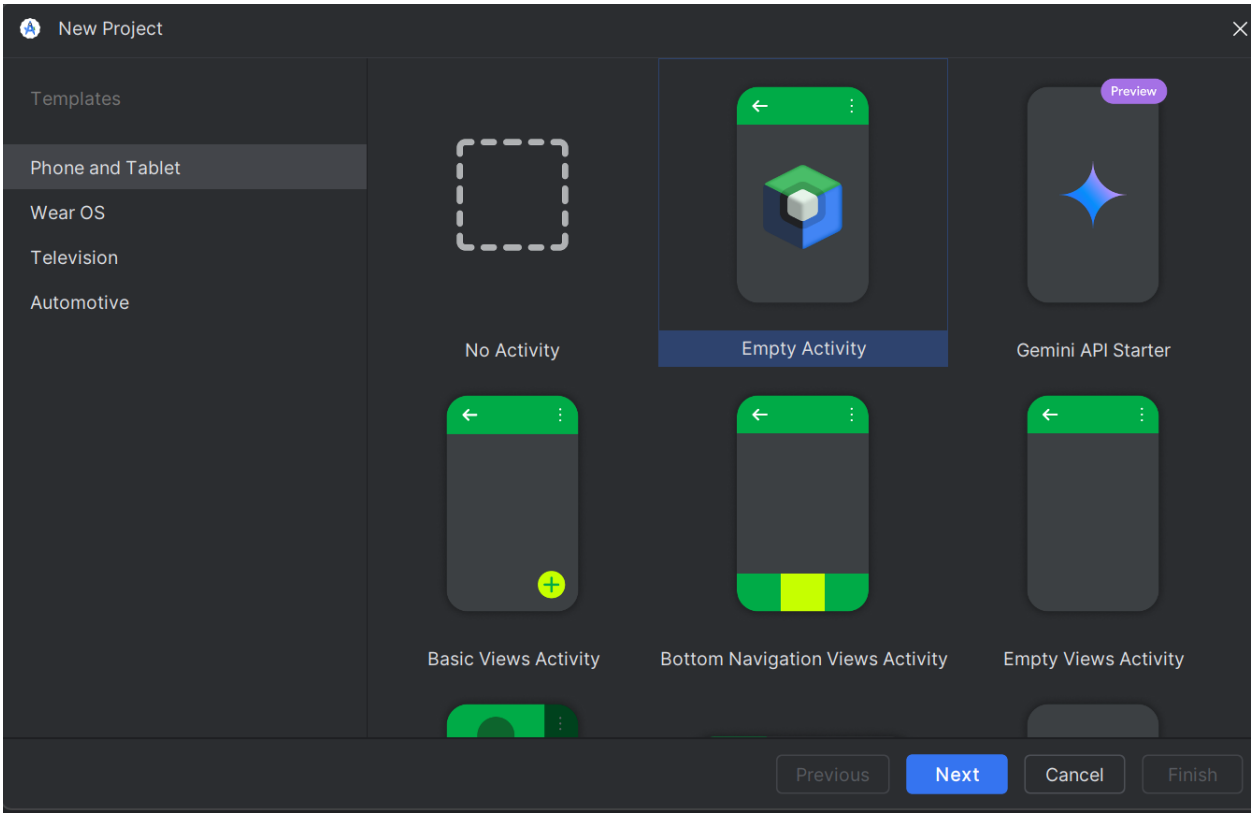
Task 1: Project Setup

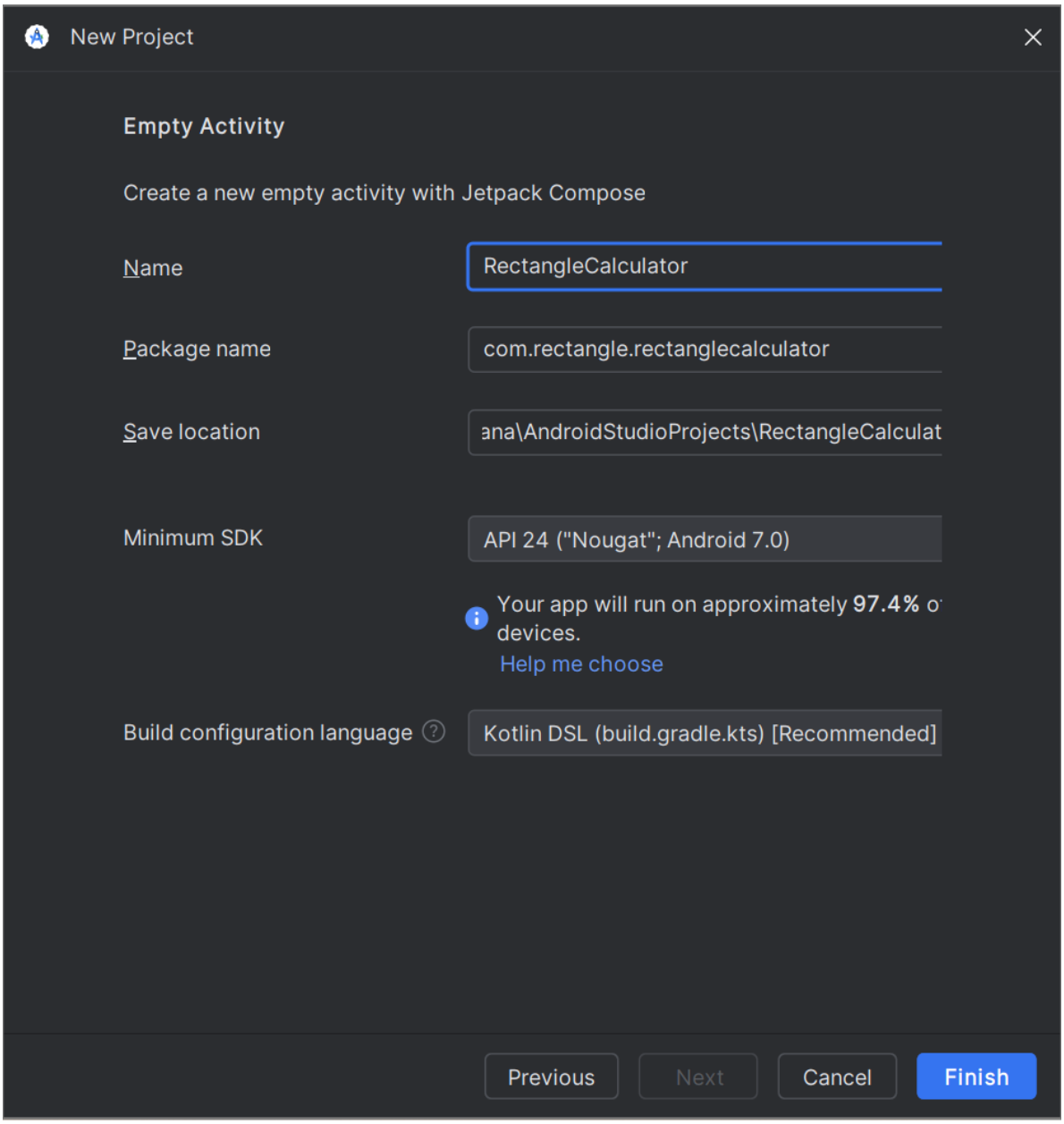
1. Install Android Studio (if not already installed).
2. Create a new Android project with the following settings:

o Project Name: RectangleCalculator

o Language: Kotlin

o Minimum SDK: API 21 (Android 5.0 Lollipop)





Task 2: User Interface Design

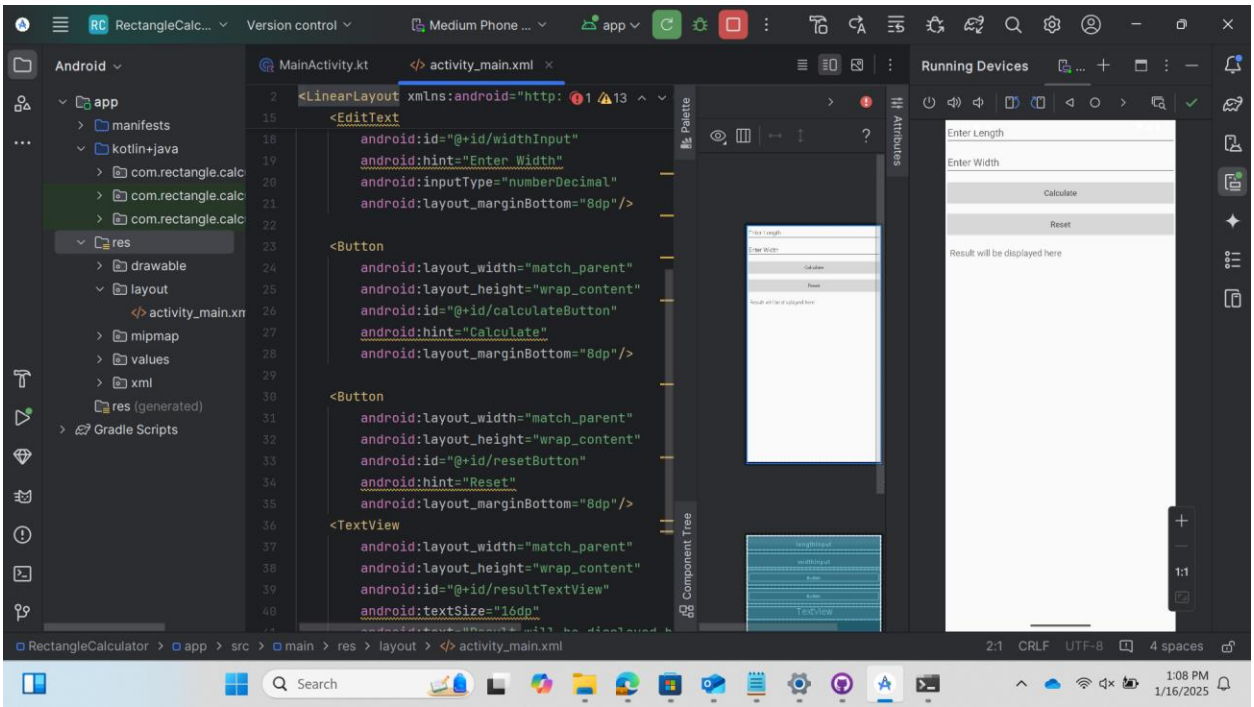
1. Design a layout in activity_main.xml with the following components:

o Two EditText fields for entering the length and width of the rectangle.

o A Button labeled "Calculate" to trigger the calculations.

o A Button labeled "Reset" to clear inputs and results.

o A TextView to display the results (perimeter and area) or error messages.



The EditText field is the place where users input the length. Match_parent will make sure to take full width of parent. Whereas wrap_content will adjust its height according to the content. @+id/lengthInput it will assign a unique identifier to the input field
which is further used by java code. android:hint="Enter Length" this acts as a place holder when the fields are empty. android:inputType="numberDecimal it ensures that only decimal numbers are entered. android:layout_marginBottom="8dp" Margin bottom adds a bottom margin between this and the next view . The same same editText is used for width as well

Calculate Button: @+id/calculateButton it triggers calculation when pressed. android:layout_marginBottom="8dp" it generates spacing below the button

Rest Button: the layout will be same as calculate button. @+id/resetButton to reset the inputs when reset button is pressed

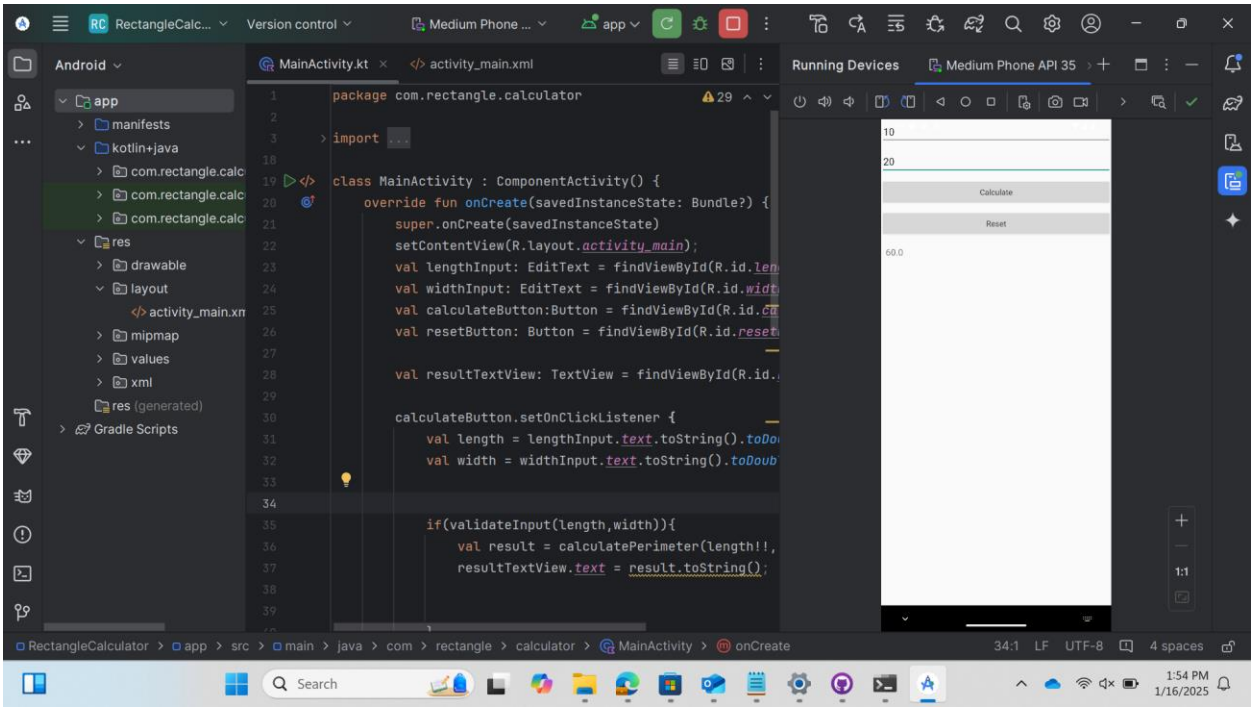
TextView: android:id="@+id/resultTextView assigns a uniquie identifier to text view. android:text="Result will be displayed here “this is the default text which is shown initially. android:textSize="16dp" size of text is 16 density-indepent pixels. android:padding="8dp" it adds spacing inside the text view

Task 3: Kotlin Code Implementation

1. Implement the logic in MainActivity.kt:

o Define dedicated functions for input validation, perimeter calculation, and area calculation.

▪ validateInput: Ensures that input values are numeric and greater than zero.
▪ calculatePerimeter: Calculates the perimeter using the formula: 2 * (length + width).
▪ calculateArea: Calculates the area using the formula: length * width.
o Use when expressions for detailed validation feedback.
o Incorporate a loop (for or while) to demonstrate iterative execution:
▪ After validating inputs, use a for loop or while loop to simulate multiple calculations. For example, calculate the perimeter and area three times to demonstrate iterative execution.
▪ Example using a for loop:
▪ Example using a while loop:
o Use setOnClickListener to:
▪ Trigger calculations when the "Calculate" button is clicked.
▪ Reset the UI when the "Reset" button is clicked.



OnCreate() this method is a lifecycle method which is used to set up the layout and event listener are configured. setContentView it sets the content view for the layout defined in activity_main.xml. findViewById() are user interact components. lengthInput, widthInput are editText fileds , `calculateButton` and `resetButton` are buttons which get triggered when pressed. resultTextView this will display the result of area and perimeter.

OnClickListener: when user clicks the calculate button this block is triggered. lengthInput.text.toString().toDoubleOrNull() if the input is valid, it will convert the input to double, otherwise it will return null.

Task 4: Debugging and Testing

1. Test the app for various scenarios:

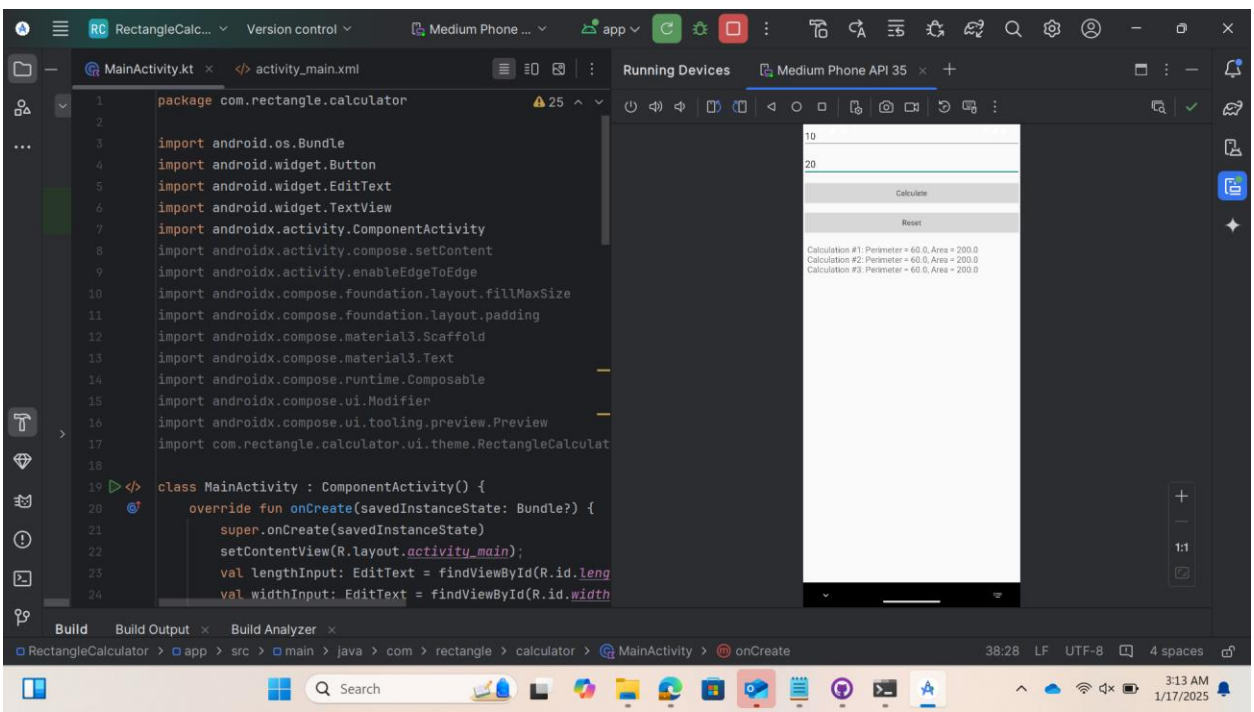
o Valid inputs (e.g., length = 10, width = 20).

o Invalid inputs (e.g., negative values, non-numeric inputs, empty fields).

2. Use Android Studio's debug tools to observe variable values and application flow.

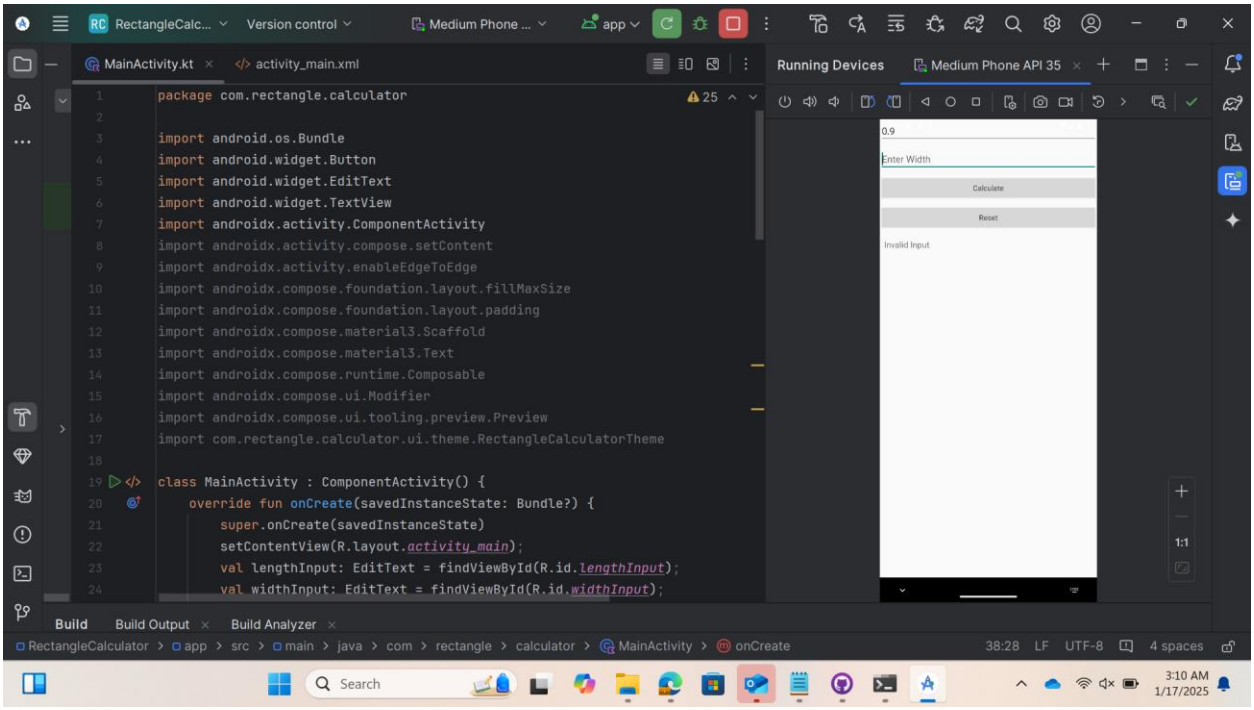
3. Take screenshots showing the application in action, including inputs and displayed results for valid and invalid scenarios

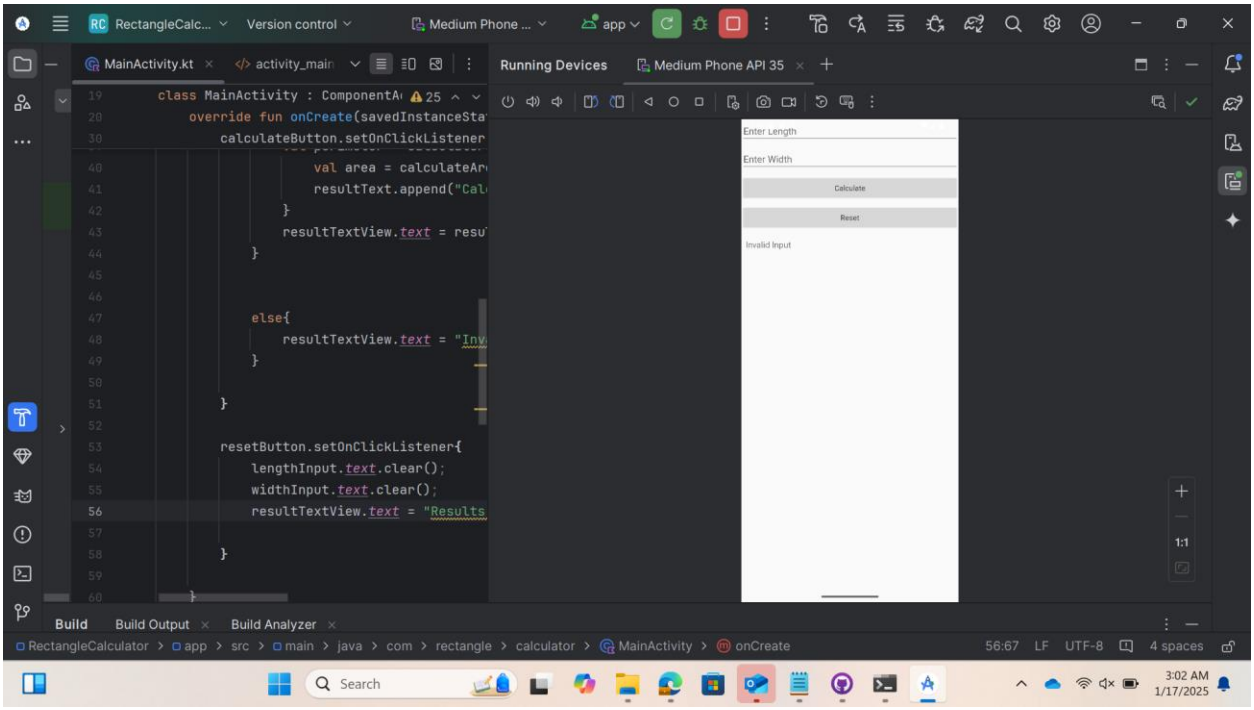
Valid scenario:

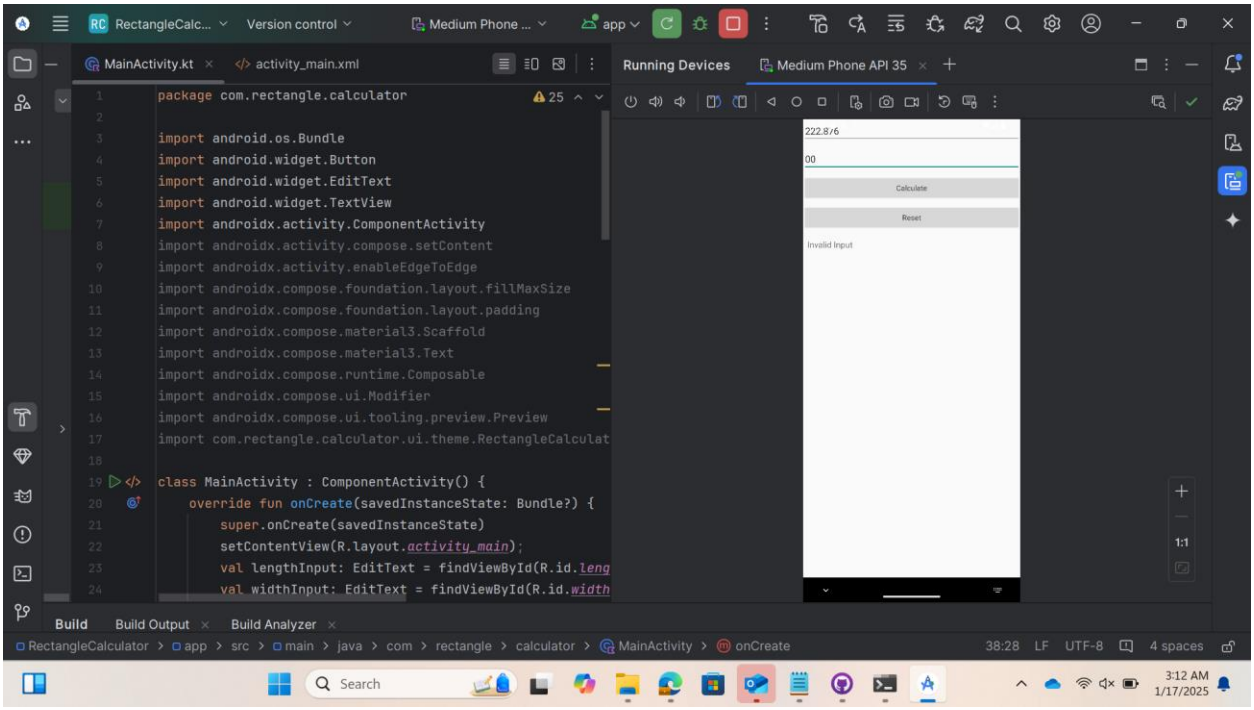


ValidateInput(length,width) it will check whether the given inputs are valid, not null and greater than zero. If the given inputs are valid then the area and perimeter gets calculated 3 times. Because it is in for loop and append the result to resultText. After calculations the results are displayed in the resultTextView.

Invalid scenario:







ValidateInput() this method checks whether the length and width values are valid. For a valid input it will return true. Else it returns false if the given value is null or less than or equal to zero.

Challenges faced:

1) **Input validation was crucial but by using toDoubleOrNull() and validateInput() methods handles the invalid inputs.**
2) **User interface design was little difficult because spacing the buttons, labels and spacing between elements.**
3) **Ensuring button behavior such as calculate, reset are performing correctly in all scenarios**
4) **Maintaining clear and understandable code structure**