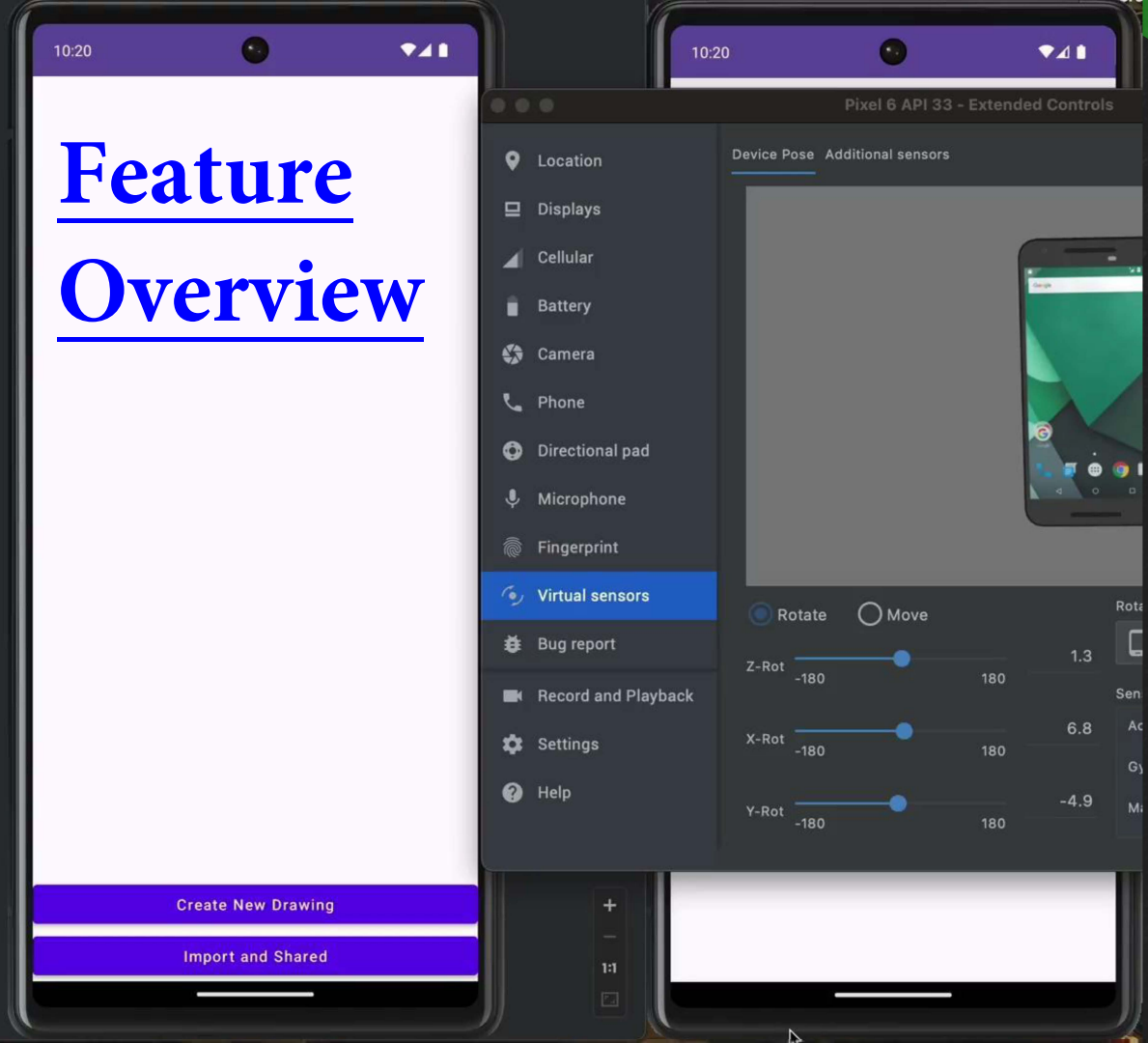
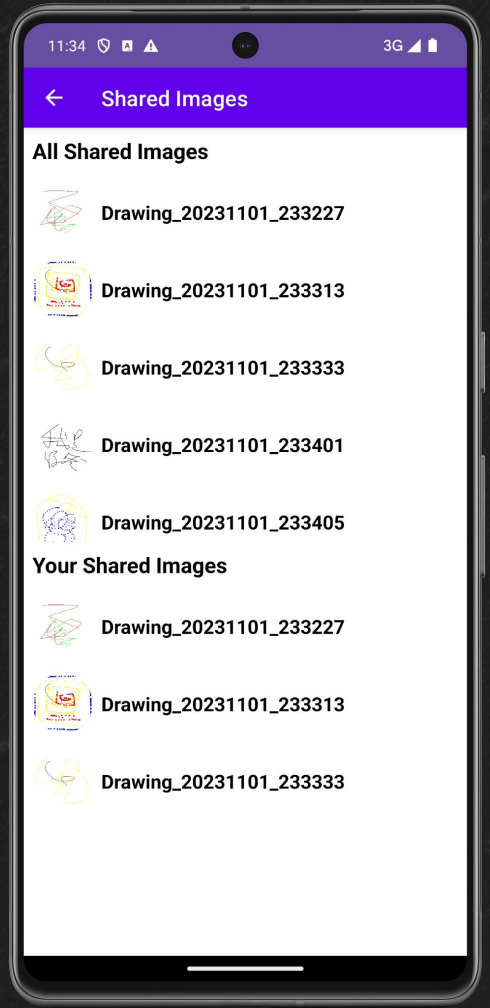
Feature Overview



Scaffold

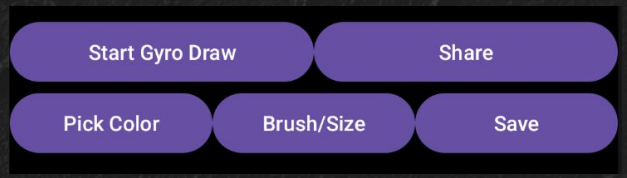
```
Yutian_Qin +1 *
@SuppressLint("UnusedMaterialScaffoldPaddingParameter")
@Composable
fun DrawingLists(
    navController: NavController,
    drawingViewModel: DrawingViewModel,
    drawingRepository: DrawingRepository
) {
    val allUsersImages by drawingViewModel.allUsersImages.collectAsState()
    val thisUsersImages by drawingViewModel.thisUsersImages.collectAsState()
    val baseURL = "http://10.0.2.2:8080/drawings/"
    val scope = rememberCoroutineScope()

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text(text = "Shared Images") },
                navigationIcon = {
                    IconButton(onClick = {
                        // Navigate back to the startFragment
                        navController.popBackStack()
                    }) {
                        Icon(Icons.Default.ArrowBack, contentDescription = null)
                    }
                }
            )
        }
    ) { it: PaddingValues
```



Gyro

- Senor based drawing using gyro sensor
- Main aspects of feature:
  - Button to start/stop feature
  - SensorEventListener
  - Handling the gyro data




```
private fun startGyroSensor(sensorManager: SensorManager, gyroscope: Sensor, customView: CustomView) {
    val gyroFlow: Flow<FloatArray> = getGyroData(gyroscope, sensorManager)

    gyroFlowJob?.cancel()
    gyroFlowJob = viewLifecycleOwner.lifecycleScope.launch { this: CoroutineScope
        gyroFlow.collect { gyroReading ->
            customView.handleGyroInput(gyroReading[0], gyroReading[1])
        }
    }
}
```

```
fun stopGyroSensor() {
    if (gyroSensorListener != null) {
        sensorManager.unregisterListener(gyroSensorListener)
    }
}
```

Gyro

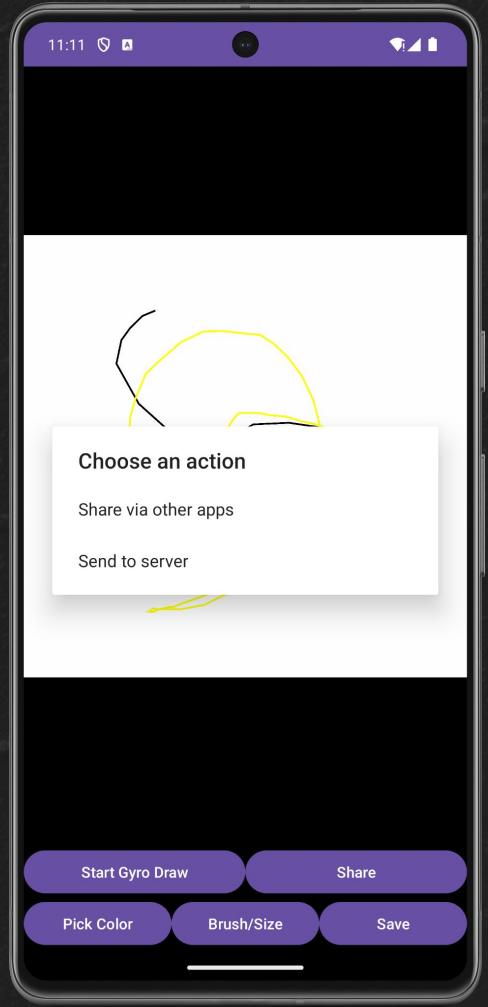
- Handling the data:
  - Keep track of the last path position
  - Grab only the x and y data
  - Clamp the data so path stays in bounds
  - Draw path to the new position
  - Can adjust sensitivity, higher the sensitivity the longer created paths

```
fun handleGyroInput(x: Float, y: Float) {
    if (isGyroDrawing) {
        lastGyroPosition?.let { lastPos ->
            // Calculate the new position based on gyroscope input
            val newX = lastPos.x + x * sensitivity
            val newY = lastPos.y + y * sensitivity
            // Clamp the values to make sure they're within the canvas bounds
            val clampedX = newX.coerceIn(0f, width.toFloat())
            val clampedY = newY.coerceIn(0f, height.toFloat())
```

AlertDialog

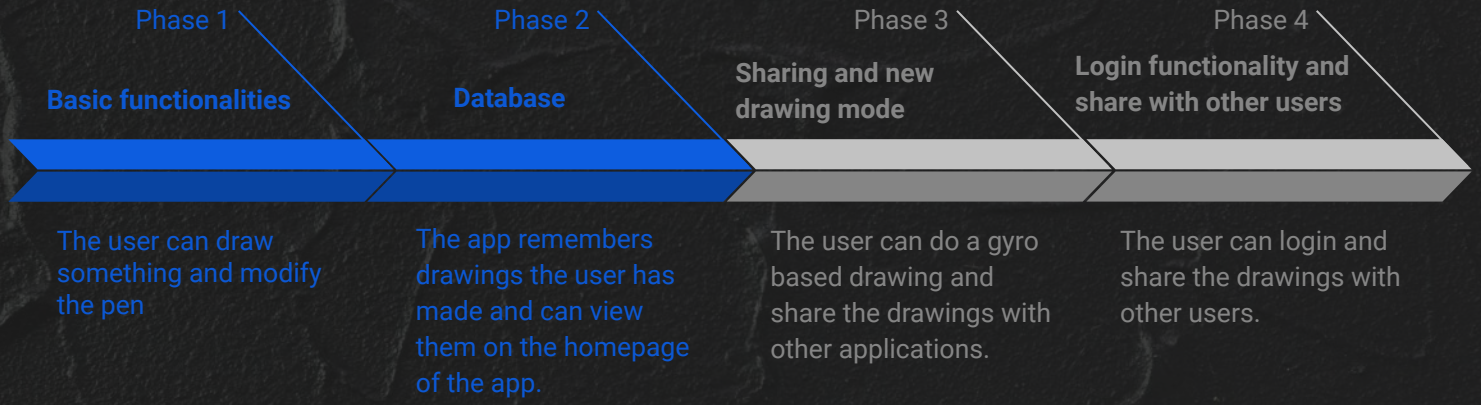
```
Yutian_Qin +1
private fun onClickShare(anchorView: View) {
    val customView = view?.findViewById<CustomView>(R.id.custom_view)
    val bitmap = customView?.getDrawingBitmap()

    bitmap?.let { it: Bitmap
        val choices = arrayOf("Share via other apps", "Send to server")
        AlertDialog.Builder(requireContext()) AlertDialog.Builder
            .setTitle("Choose an action") AlertDialog.Builder!
            .setItems(choices) { _, which ->
                when (which) {
                    0 -> shareImageAndText(it) // Share via other apps
                    1 -> sendImageToServer(it) // Send to server
                }
            }.show() ^let
    }
}
```



UI

Changes made to the UI as the client added more requirements.



-Basic layout
-2 buttons

-Drawings page
-Save button

-Share button
-Gyro button

-Login page
-Shared drawings page