
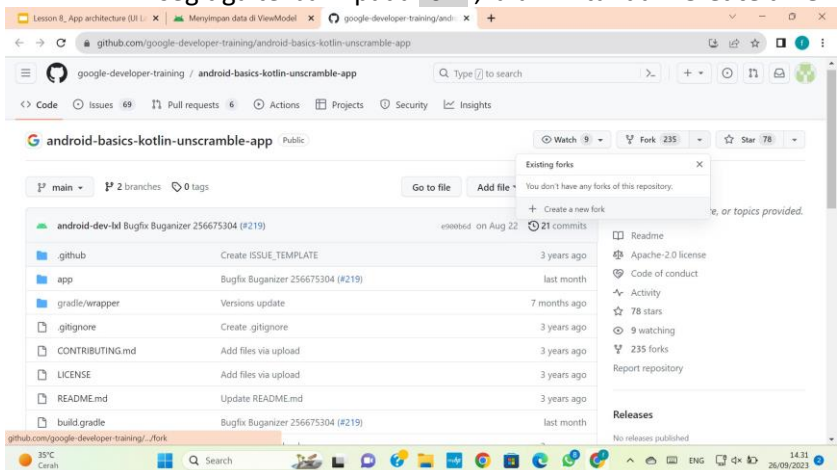
	<b>JURUSAN TEKNIK</b> <b>PROGRAM STUDI D-III TEKNOLOGI INFORMASI</b>		
	<b>Mata Kuliah</b>	:	Praktik Pemrograman Mobile 2
	<b>Dosen Pengampu</b>	:	Sigit Kariagil Bimonugroho, S.Kom., M.T.
	<b>Sub CPMK</b>	:	Arsitektur Aplikasi (Lapisan UI)

## LAPORAN PRAKTIKUM

IDENTITAS MAHASISWA	
<b>Nama Mahasiswa</b>	: Firza Aji Karya
<b>NPM</b>	: 223307007
<b>Kelas</b>	: 3A

DASAR TEORI
<p>Dalam GitHub, repository adalah tempat menyimpan file proyek yang dapat berupa file <i>source code</i>, file gambar, text, dan file lainnya. Umumnya repository memang digunakan untuk menyimpan <i>source code</i> karena berhubungan dengan pemrograman.</p> <p>Arsitektur aplikasi adalah seperangkat aturan desain. Mirip dengan cetak biru sebuah rumah, arsitektur memberikan struktur bagi aplikasi Anda. Arsitektur aplikasi yang baik dapat membuat kode Anda andal, fleksibel, skalabel, dan mudah dikelola selama bertahun-tahun. mempelajari cara menggunakan ViewModel, salah satu komponen Arsitektur untuk menyimpan data aplikasi Anda. Data yang disimpan tidak akan hilang jika framework menghancurkan dan membuat ulang aktivitas dan fragmen selama perubahan konfigurasi atau peristiwa lainnya.</p>

ALAT DAN BAHAN
<ol style="list-style-type: none"> <li>1. Laptop</li> <li>2. Android Studio</li> <li>3. GitHub</li> </ol>

KODE PROGRAM DAN HASIL
<p>Membuat Repository</p> <ul style="list-style-type: none"> <li>• Klik link berikut <a href="https://github.com/google-developer-training/android-basics-kotlin-unsramble-app">https://github.com/google-developer-training/android-basics-kotlin-unsramble-app</a></li> <li>• Pilih segitiga terbalik pada fork, lalu klik tanda + Create a new fork</li> </ul> 

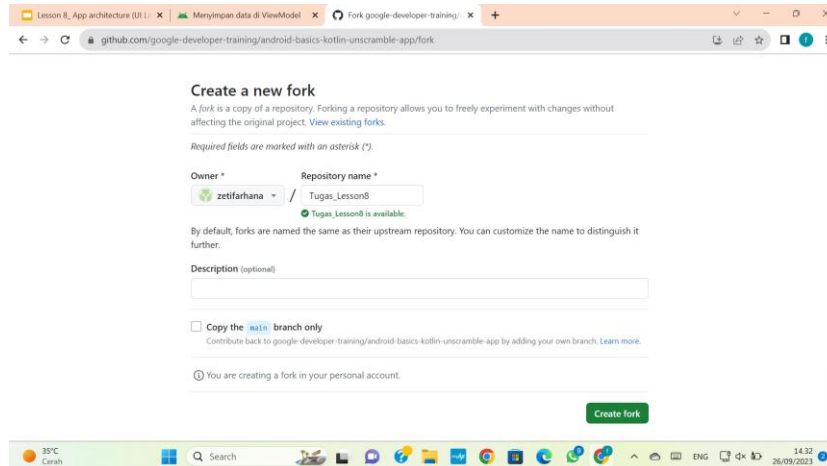


**JURUSAN TEKNIK**  
**PROGRAM STUDI D-III TEKNOLOGI INFORMASI**

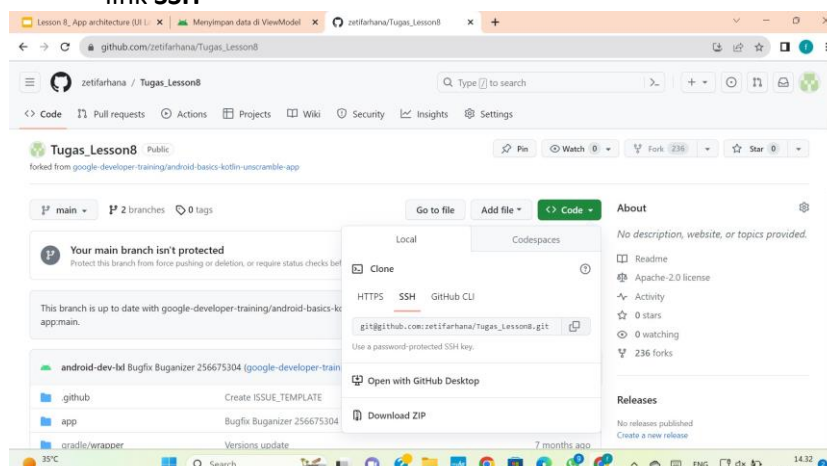
**Mata Kuliah** : Praktik Pemrograman Mobile 2  
**Dosen Pengampu** : Sigit Kariagil Bimonugroho, S.Kom., M.T.  
**Sub CPMK** : Arsitektur Aplikasi (Lapisan UI)



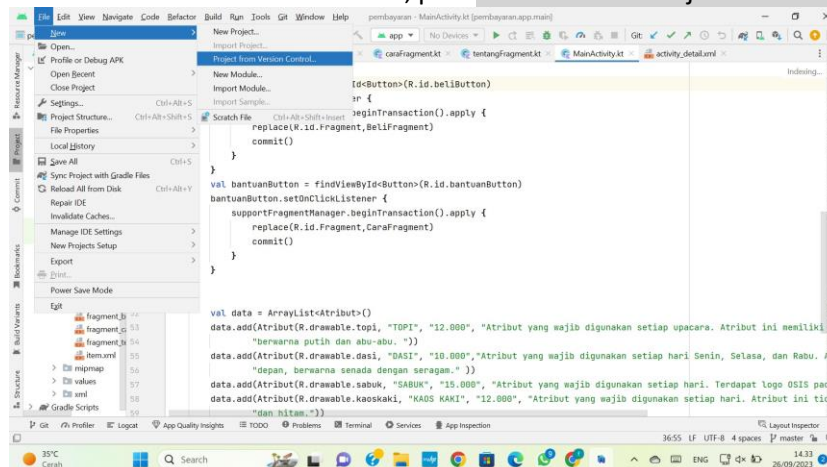
- Ganti nama Repository, jangan centang pada **copy the main branch only** lalu **create fork**



- Repository sudah berhasil dibuat, kemudian **Clone** dengan cara klik **Code** lalu copy link **SSH**



- Masuk ke Android Studio, pilih File -> New -> Project from Version Control



- Paste link SSH yang sudah di copy, kemudian **Clone**

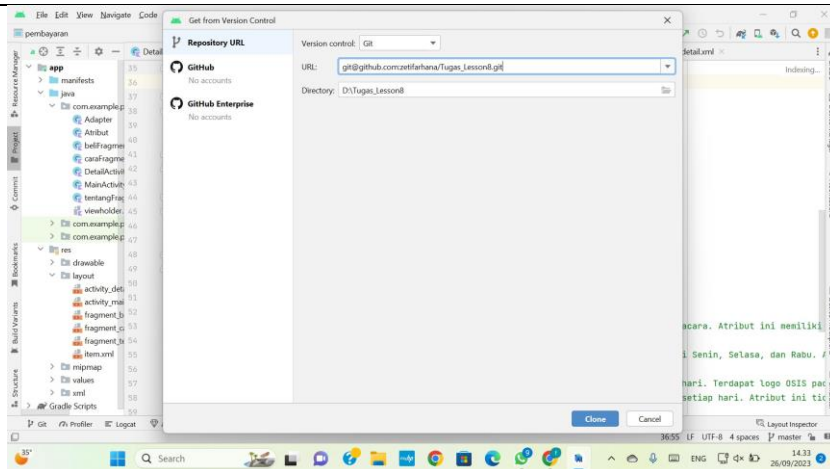


**JURUSAN TEKNIK**  
**PROGRAM STUDI D-III TEKNOLOGI INFORMASI**

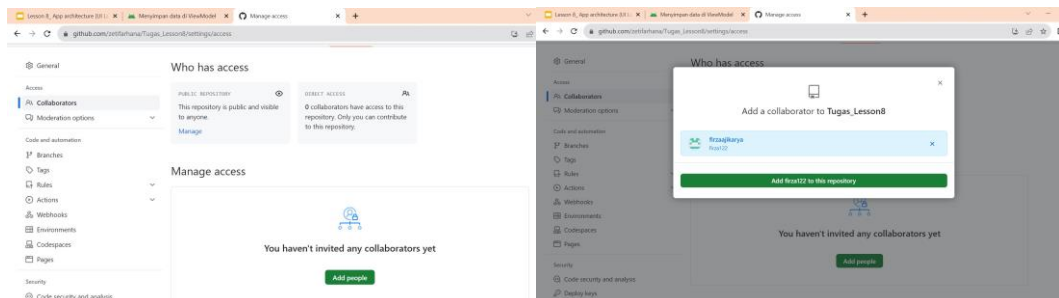
**Mata Kuliah** : Praktik Pemrograman Mobile 2

**Dosen Pengampu** : Sigit Kariagil Bimonugroho, S.Kom., M.T.

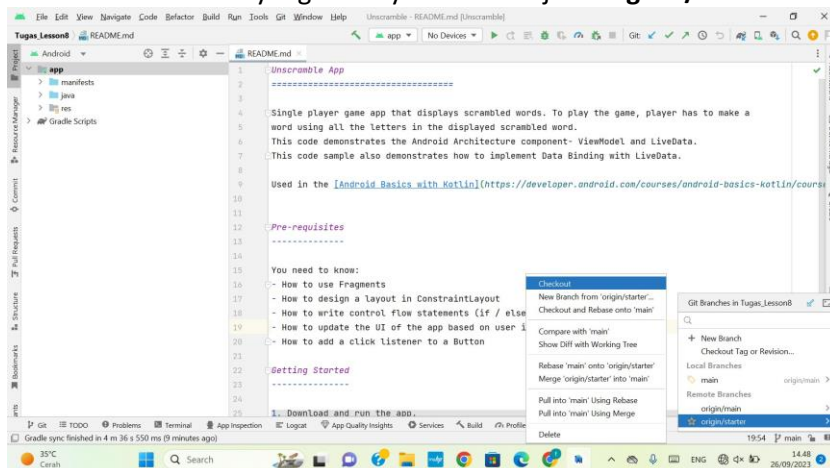
**Sub CPMK** : Arsitektur Aplikasi (Lapisan UI)



- Invite teman ke Repository yang sudah dibuat dengan cara masuk ke Setting -> Collaborators -> Add People -> Masukkan Username GitHub teman , tunggu teman menerima invite



- Buat branch yang awalnya main menjadi **original/starter** lalu Checkout



CodeLab

#### 4. Menambahkan View Model

- Buat file class Kotlin baru bernama **GameViewModel**, Di jendela **Android**, klik kanan pada folder **ui.game**. Pilih **New > Kotlin File/Class**.

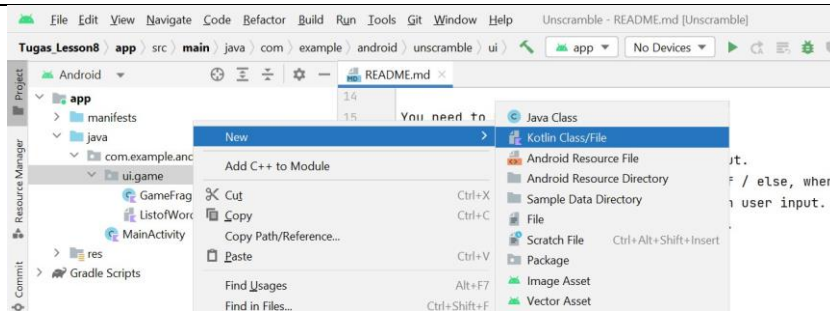


**JURUSAN TEKNIK**  
**PROGRAM STUDI D-III TEKNOLOGI INFORMASI**

**Mata Kuliah** : Praktik Pemrograman Mobile 2

**Dosen Pengampu** : Sigit Kariagil Bimonugroho, S.Kom., M.T.

**Sub CPMK** : Arsitektur Aplikasi (Lapisan UI)



- Ubah **GameViewModel** menjadi subclass dari **ViewModel**  

```
class GameViewModel : ViewModel() {
```
- Pada bagian atas class **GameFragment** tambahkan properti jenis **GameViewModel**. Lakukan inisialisasi menggunakan properti Kotlin **by viewModels()**  

```
class GameFragment : Fragment() {  
    private val viewModel : GameViewModel by viewModels()
```

5. Memindahkan data ke ViewModel

- Pindahkan variabel data **score**, **currentWordCount**, **currentScrambleWord** ke class **GameViewModel**

```
class GameViewModel : ViewModel() {  
  
    private var score = 0  
    private var currentWordCount = 0  
    private var currentScrambledWord = "test"  
}
```

- Dalam **GameViewModel**, Ubah deklarasi **currentScrambleWord** untuk menambahkan properti pendukung
- Di **Game Fragment**, perbarui metode **updateNextWordOnScreen()**  

```
private fun updateNextWordOnScreen() {  
    binding.textViewUnscrambledWord.text = viewModel.currentScrambledWord  
}
```

6. Memahami siklus proses ViewModel

- Menambahkan blok **init** di **GameViewModel.kt**  

```
init {  
    Log.d( tag: "GameFragment", msg: "GameViewModel created!")  
}
```
- Menambahkan laporan log didalam **onCleared()**  

```
override fun onCleared() {  
    super.onCleared()  
    Log.d( tag: "GameFragment", msg: "GameViewModel destroyed!")  
}
```
- Di **GameFragment** dalam **onCreateView()**, setelah Anda mendapatkan referensi ke objek binding, tambahkan laporan log untuk mencatat pembuatan fragmen.



**JURUSAN TEKNIK**  
**PROGRAM STUDI D-III TEKNOLOGI INFORMASI**

**Mata Kuliah** : Praktik Pemrograman Mobile 2  
**Dosen Pengampu** : Sigit Kariagil Bimonugroho, S.Kom., M.T.  
**Sub CPMK** : Arsitektur Aplikasi (Lapisan UI)



```
override fun onCreateView(  
    inflater: LayoutInflater, container: ViewGroup?,  
    savedInstanceState: Bundle?  
): View {  
    // Inflate the layout XML file and return a binding object instance  
    binding = GameFragmentBinding.inflate(inflater, container, attachToParent: false)  
    Log.d( tag: "GameFragment", msg: "GameFragment created/re-created!")  
    Log.d( tag: "GameFragment", msg: "Word: ${viewModel.currentScrambledWord} " +  
        "Score: ${viewModel.score} WordCount: ${viewModel.currentWordCount}")  
    return binding.root  
}
```

- Di GameFragment, ganti metode callback `onDetach()`, yang akan dipanggil saat aktivitas dan fragmen yang terkait dihancurkan.

```
override fun onDetach() {  
    super.onDetach()  
    Log.d( tag: "GameFragment", msg: "GameFragment destroyed!")  
}
```

#### 7. Mengisi ViewModel

- Pada GameViewModel, tambahkan variabel class baru jenis `MutableList<String>` yang disebut `wordsList`. Tambahkan variabel class lain yang disebut `currentWord` untuk menyimpan kata yang ingin disusun oleh pemain. Gunakan kata kunci `lateinit` karena Anda akan menginisialisasi properti ini nanti.

```
private var wordsList: MutableList<String> = mutableListOf()  
private lateinit var currentWord: String
```

- Tambahkan metode `private` baru yang disebut `getNextWord()`, di atas blok `init`, tanpa parameter yang tidak menampilkan apa-apa. Dapatkan kata acak dari `allWordsList` dan tetapkan ke `currentWord`.



```
private fun getNextWord() {  
    currentWord = allWordsList.random()  
}
```

- Di `getNextWord()`, konversikan string `currentWord` ke array karakter dan tetapkan ke `val` baru yang disebut `tempWord`.

```
val tempWord = currentWord.toCharArray()  
tempWord.shuffle()
```

- Terkadang urutan karakter yang diacak sama dengan kata aslinya. Tambahkan loop `while` berikut di sekitar panggilan untuk mengacak, untuk melanjutkan loop hingga kata yang diacak tidak sama dengan kata asli.

```
while (String(tempWord).equals(currentWord, ignoreCase: false)) {  
    tempWord.shuffle()  
}
```

	<b>JURUSAN TEKNIK</b> <b>PROGRAM STUDI D-III TEKNOLOGI INFORMASI</b>		
	<b>Mata Kuliah</b>	:	Praktik Pemrograman Mobile 2
	<b>Dosen Pengampu</b>	:	Sigit Kariagil Bimonugroho, S.Kom., M.T.
	<b>Sub CPMK</b>	:	Arsitektur Aplikasi (Lapisan UI)

- Tambahkan blok if-else untuk memeriksa apakah suatu kata telah digunakan. Jika wordsList berisi currentWord, panggil getNextWord().

```
if (wordsList.contains(currentWord)) {
    getNextWord()
} else {
    _currentScrambledWord = String(tempWord)
    ++_currentWordCount
    wordsList.add(currentWord)
}
```



- Untuk menampilkan kata yang ejaannya diacak di awal aplikasi, Anda perlu memanggil metode getNextWord(), yang selanjutnya memperbarui currentScrambledWord. Lakukan panggilan ke metode getNextWord() di dalam blok init pada GameViewModel.

```
init {
    Log.d( tag: "GameFragment", msg: "GameViewModel created!")
    getNextWord()
}
```

- Tambahkan pengubah lateinit ke properti \_currentScrambledWord. Tambahkan penyebutan eksplisit jenis data String,  
`private lateinit var _currentScrambledWord: String`
- Hasil





	<b>JURUSAN TEKNIK</b> <b>PROGRAM STUDI D-III TEKNOLOGI INFORMASI</b>		
	<b>Mata Kuliah</b>	:	Praktik Pemrograman Mobile 2
	<b>Dosen Pengampu</b>	:	Sigit Kariagil Bimonugroho, S.Kom., M.T.
	<b>Sub CPMK</b>	:	Arsitektur Aplikasi (Lapisan UI)

#### 8. Dialog

- tambahkan properti pendukung ke variabel `score`. Di `GameViewModel`  

```
private var _score = 0
val score: Int
    get() = _score
```
- Di `GameFragment`, tambahkan fungsi pribadi bernama `showFinalScoreDialog()`. Untuk membuat `MaterialAlertDialog`, gunakan class `MaterialAlertDialogBuilder` untuk membuat bagian dialog langkah demi langkah.  



```
private fun showFinalScoreDialog() {
    MaterialAlertDialogBuilder(requireContext())
```
- Tambahkan kode untuk menetapkan judul pada dialog pemberitahuan, gunakan resource string  

```
MaterialAlertDialogBuilder(requireContext())
    .setTitle("Congratulations!")
```
- Setel pesan agar menampilkan skor akhir, gunakan versi hanya baca dari variabel skor (`viewModel.score`), yang telah Anda tambahkan sebelumnya.  

```
.setMessage("You scored: {viewModel.score}")
```
- Jadikan dialog pemberitahuan tidak dapat dibatalkan saat tombol kembali ditekan, menggunakan metode `setCancelable()` dan meneruskan `false`.  

```
.setCancelable(false)
```
- Tambahkan dua tombol teks **EXIT** dan **PLAY AGAIN** menggunakan metode `setNegativeButton()` dan `setPositiveButton()`. Panggil masing-masing `exitGame()` dan `restartGame()` dari lambda.  

```
.setNegativeButton("Exit") { _, _ ->
    exitGame()
}
.setPositiveButton("Play Again") { _, _ ->
    restartGame()
}
```
- Di bagian akhir, tambahkan `show()`, yang akan membuat lalu menampilkan dialog pemberitahuan.

	<b>JURUSAN TEKNIK</b> <b>PROGRAM STUDI D-III TEKNOLOGI INFORMASI</b>		
	<b>Mata Kuliah</b>	:	Praktik Pemrograman Mobile 2
	<b>Dosen Pengampu</b>	:	Sigit Kariagil Bimonugroho, S.Kom., M.T.
	<b>Sub CPMK</b>	:	Arsitektur Aplikasi (Lapisan UI)

```

private fun showFinalScoreDialog() {
    MaterialAlertDialogBuilder(requireContext())
        .setTitle("Congratulations!")
        .setMessage("You scored: {viewModel.score}")
        .setCancelable(false)
        .setNegativeButton("Exit") { _, _ ->
            exitGame()
        }
        .setPositiveButton("Play Again") { _, _ ->
            restartGame()
        }
        .show()
}

```

#### 9. mengimplementasikan OnClickListener untuk tombol Submit

- Hapus kode di dalam `onSubmitWord()` yang akan dipanggil saat tombol **Submit** diketuk.
- Tambahkan centang pada nilai return metode `viewModel.nextWord()`. Jika `true`, kata lain tersedia, jadi perbarui kata yang ejaannya diacak di layar menggunakan `updateNextWordOnScreen()`.



```

private fun onSubmitWord() {
    val playerWord = binding.textInputEditText.text.toString()
    if (viewModel.isUserWordCorrect(playerWord)) {
        setErrorTextField(false)
        if (viewModel.nextWord()) {
            updateNextWordOnScreen()
        } else {
            showFinalScoreDialog()
        }
    }
}

```

- Mencoba



	<b>JURUSAN TEKNIK</b> <b>PROGRAM STUDI D-III TEKNOLOGI INFORMASI</b>		
	<b>Mata Kuliah</b>	:	Praktik Pemrograman Mobile 2
	<b>Dosen Pengampu</b>	:	Sigit Kariagil Bimonugroho, S.Kom., M.T.
	<b>Sub CPMK</b>	:	Arsitektur Aplikasi (Lapisan UI)

16.16

Unscramble

0 of 10 words

SCORE: 0

erfe

Unscramble the word using all the letters.

Enter your word

yoyo

SKIP

SUBMIT

≡
○
<

- Di `GameViewModel`, tambahkan metode pribadi baru yang disebut `increaseScore()` tanpa parameter dan nilai yang ditampilkan. Tingkatkan variabel `score` sebesar `SCORE_INCREASE`.

```

private fun increaseScore() {
    _score += SCORE_INCREASE
}

```
- Dalam `GameViewModel`, tambahkan metode bantuan yang disebut `isUserWordCorrect()` yang menampilkan `Boolean` dan mengambil `String`, kata dari pemain, sebagai parameter.
- Di `isUserWordCorrect()` validasikan kata dari pemain dan tingkatkan skor jika tebakannya benar. Tindakan ini akan memperbarui skor akhir dalam dialog pemberitahuan.



**JURUSAN TEKNIK  
PROGRAM STUDI D-III TEKNOLOGI INFORMASI**

**Mata Kuliah** : Praktik Pemrograman Mobile 2  
**Dosen Pengampu** : Sigit Kariagil Bimonugroho, S.Kom., M.T.  
**Sub CPMK** : Arsitektur Aplikasi (Lapisan UI)





```
fun isUserWordCorrect(playerWord: String): Boolean {  
    if (playerWord.equals(currentWord, ignoreCase: true)) {  
        increaseScore()  
        return true  
    }  
    return false  
}
```

- Pada GameFragment, di awal onSubmitWord(), buat val yang bernama playerWord
- Pada onSubmitWord(), di bawah deklarasi playerWord, validasikan kata pemain. Tambahkan pernyataan if untuk memeriksa kata pemain menggunakan metode isUserWordCorrect(), dengan meneruskan playerWord.
- Di dalam blok if, reset kolom teks, panggil setErrorTextField dengan memasukkan false.
- Pindahkan kode yang ada ke dalam blok if.

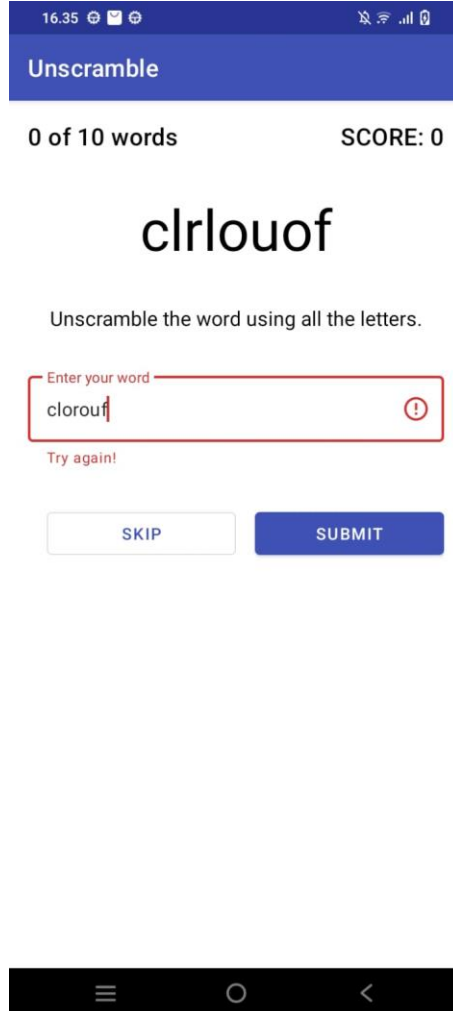
```
private fun onSubmitWord() {  
    val playerWord = binding.textInputEditText.text.toString()  
    if (viewModel.isUserWordCorrect(playerWord)) {  
        setErrorTextField(false)  
        if (viewModel.nextWord()) {  
            updateNextWordOnScreen()  
        } else {  
            showFinalScoreDialog()  
        }  
    }  
}
```

- Jika kata pengguna salah, tampilkan pesan error di kolom teks. Tambahkan blok else ke blok if di atas, lalu panggil setErrorTextField() yang meneruskan true.

```
private fun onSubmitWord() {  
    val playerWord = binding.textInputEditText.text.toString()  
    if (viewModel.isUserWordCorrect(playerWord)) {  
        setErrorTextField(false)  
        if (viewModel.nextWord()) {  
            updateNextWordOnScreen()  
        } else {  
            showFinalScoreDialog()  
        }  
    } else {  
        setErrorTextField(true)  
    }  
}
```

	<b>JURUSAN TEKNIK</b> <b>PROGRAM STUDI D-III TEKNOLOGI INFORMASI</b>		
	<b>Mata Kuliah</b>	:	Praktik Pemrograman Mobile 2
	<b>Dosen Pengampu</b>	:	Sigit Kariagil Bimonugroho, S.Kom., M.T.
	<b>Sub CPMK</b>	:	Arsitektur Aplikasi (Lapisan UI)

- Jalankan Aplikasi





#### 10. Mengimplementasikan tombol Skip

- Serupa dengan `onSubmitWord()`, tambahkan kondisi dalam metode `onSkipWord()`. Jika `true`, tampilkan kata di layar dan reset kolom teks. Jika `false` dan tidak ada lagi kata yang tersisa di babak ini, tampilkan dialog pemberitahuan dengan skor akhir.

```
private fun onSkipWord() {
    if (viewModel.nextWord()) {
        setErrorTextField(false)
        updateNextWordOnScreen()
    } else {
        showFinalScoreDialog()
    }
}
```

#### 11. Memverifikasi ViewModel mempertahankan data

	<b>JURUSAN TEKNIK</b>		
	<b>PROGRAM STUDI D-III TEKNOLOGI INFORMASI</b>		
	<b>Mata Kuliah</b>	: Praktik Pemrograman Mobile 2	
	<b>Dosen Pengampu</b>	: Sigit Kariagil Bimonugroho,S.Kom., M.T.	
	<b>Sub CPMK</b>	: Arsitektur Aplikasi (Lapisan UI)	

- Untuk tugas ini, tambahkan logging di GameFragment untuk mengamati bahwa data aplikasi Anda disimpan di ViewModel, selama perubahan konfigurasi.

#### 12. Memperbarui logika restart game

- Untuk mereset data aplikasi, di GameViewModel, tambahkan metode yang disebut `reinitializeData()`. Tetapkan skor dan jumlah kata menjadi 0. Hapus daftar kata dan panggil metode `getNextWord()`.

```
fun reinitializeData() {
    _score = 0
    _currentWordCount = 0
    wordsList.clear()
    getNextWord()
}
```

- Pada GameFragment di bagian atas metode `restartGame()`, lakukan panggilan ke metode yang baru dibuat, `reinitializeData()`.

```
private fun restartGame() {
    viewModel.reinitializeData()
    setErrorTextField(false)
    updateNextWordOnScreen()
}
```

- Jalankan

### KESIMPULAN

Pada praktikum ini kita dapat mengetahui bagaimana cara membuat Repository dan menginvite teman di GitHub dan mengClone di Android Studio. Kita juga dapat mempelajari codelab tentang arsitektur aplikasi bagaimana cara menambahkan ViewModel, Memindahkan data ke ViewModel, dan mengisi ViewModel.