PROSIDING Seminar Nasional SISFOTEK ISSN 2597-3584 (Media Online)
Sistem Informasi dan Teknologi
website : www.seminar.iaii.or.id | email : seminar@iaii.or.id

Penerapan Pola Arsitektur Model-View-ViewModel pada Aplikasi Pembelajaran Shorof Berbasis Mobile

Havidz Muhammad Iqbal[1*], Oddy Virgantara Putra[2], Dihin Muriyatmoko[3]
[1,2,3]Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Darussalam Gontor Ponorogo
havidziqbal42032@mhs.unida.gontor.ac.id

Abstract

This research aims to develop an Android-based Basic Shorof learning application using Model-View-ViewModel (MVVM) architecture pattern. The background of this research focuses on the difficulties faced by Non KMI students at Darussalam Gontor University in understanding Shorof material, which is a major challenge in Arabic language learning. Many students have difficulty in understanding the basic concepts of Shorof, which is an important component in the overall mastery of the Arabic language. This research method includes data collection through questionnaires distributed to students. The purpose of this data collection was to identify the difficulties faced by students in learning Shorof. Analysis of the questionnaires showed that many students have difficulties in understanding the basic concepts of Shorof. This finding became the basis for the development of a learning application that is easy for students to use. The developed application uses MVVM approach to separate the business logic from the user interface. This approach facilitates future development, testing, and maintenance of the application. The features of this application include learning modules, quizzes, and Arabic vocabulary designed to strengthen students' understanding of Shorof material. Application testing involved a number of students as respondents to measure the effectiveness and efficiency of the application in improving their understanding of Shorof material. The test results showed that students experienced an increase in understanding after using this application. In conclusion, the application of MVVM architecture pattern in the development of Basic Shorof learning application can improve students' understanding effectively. This research is expected to have a positive impact on the development of educational applications in the future. Thus, this application is useful not only for students of Darussalam Gontor University, but also for the development of Arabic learning methods more broadly in various educational institutions.

Keywords: MVVM, mobile applications, instructional Media, shorof, educational application

Abstrak

Penelitian ini bertujuan untuk mengembangkan aplikasi pembelajaran Shorof Dasar berbasis Android dengan menggunakan pola arsitektur Model-View-ViewModel (MVVM). Latar belakang penelitian ini berfokus pada kesulitan yang dihadapi mahasiswa Non KMI di Universitas Darussalam Gontor dalam memahami materi Shorof, yang merupakan tantangan utama dalam pembelajaran bahasa Arab. Banyak mahasiswa mengalami kesulitan dalam memahami konsep dasar Shorof, yang merupakan komponen penting dalam penguasaan bahasa Arab secara keseluruhan. Metode penelitian ini mencakup pengumpulan data melalui kuesioner yang disebarkan kepada mahasiswa. Tujuan pengumpulan data ini adalah untuk mengidentifikasi kesulitan yang dihadapi mahasiswa dalam mempelajari Shorof. Analisis kuesioner menunjukkan bahwa banyak mahasiswa mengalami kesulitan dalam memahami konsep dasar Shorof. Temuan ini menjadi dasar bagi pengembangan aplikasi pembelajaran yang mudah digunakan oleh mahasiswa. Aplikasi yang dikembangkan menggunakan pendekatan MVVM untuk memisahkan logika bisnis dari antarmuka pengguna. Pendekatan ini memudahkan pengembangan, pengujian, dan pemeliharaan aplikasi di masa depan. Fitur-fitur aplikasi ini meliputi modul pembelajaran, kuis, dan kosakata bahasa Arab yang dirancang untuk memperkuat pemahaman mahasiswa terhadap materi Shorof. Pengujian aplikasi melibatkan sejumlah mahasiswa sebagai responden untuk mengukur efektivitas dan efisiensi aplikasi dalam meningkatkan pemahaman mereka terhadap materi Shorof. Hasil pengujian menunjukkan bahwa mahasiswa mengalami peningkatan pemahaman setelah menggunakan aplikasi ini. Kesimpulannya, penerapan pola arsitektur MVVM dalam pengembangan aplikasi pembelajaran Shorof Dasar dapat meningkatkan pemahaman mahasiswa secara efektif. Penelitian ini diharapkan dapat memberikan dampak positif bagi pengembangan aplikasi pendidikan di masa depan. Dengan demikian, aplikasi ini bermanfaat tidak hanya bagi mahasiswa Universitas Darussalam Gontor, tetapi juga bagi pengembangan metode pembelajaran bahasa Arab secara lebih luas di berbagai institusi pendidikan.

Kata kunci: MVVM, aplikasi mobile, media pembelajaran, shorof, aplikasi pendidikan

1. Pendahuluan

Pembelajaran bahasa Arab merupakan bagian penting dalam pendidikan Islam. Bahasa Arab digunakan sebagai bahasa Al-Qur'an, hadis, dan literatur Islam lainnya.[1] Oleh karena itu, pemahaman yang baik terhadap bahasa Arab sangat diperlukan untuk

memahami dan mengaplikasikan ajaran-ajaran agama dengan benar. Namun, dalam proses pembelajaran bahasa Arab, materi *Shorof* seringkali dianggap sulit bagi mahasiswa.[2] Materi *Shorof* membahas tentang bentuk dan pola kata-kata dalam bahasa Arab, yang meliputi perubahan kata kerja. Pemahaman yang baik terhadap materi *Shorof* sangat penting karena akan mempengaruhi kemampuan mahasiswa dalam membaca, menulis, dan berbicara bahasa Arab.[3]

Menurut pandangan Fuad Ni'mah, ilmu *Shorof* merupakan fondasi yang tak terpisahkan dalam pembelajaran bahasa Arab karena memiliki peran sentral dalam membongkar struktur kalimat dan tata bahasa Arab secara menyeluruh. Kehadirannya dipandang sebagai langkah yang sangat dianjurkan dalam Islam, karena kemampuannya dalam mengoptimalkan pemahaman dan penggunaan bahasa Arab dengan kesempurnaan dan kefasihan yang sesuai dengan tuntutan syariat.[4]

Selain itu, perkembangan teknologi informasi, terutama perangkat mobile berbasis Android, telah memberikan potensi yang besar dalam meningkatkan proses pembelajaran. Teknologi memungkinkan penyajian materi pembelajaran yang lebih interaktif melalui penggunaan multimedia, video, gambar, animasi, dan audio. Hal ini membuat perolehan pengetahuan menjadi lebih menarik dan efektif.[5]

Bedasarkan hasil penelitian yang dilakukan Arief Rahman Fajri, peneliti menyatakan penerapan design pattern MVVM dalam pengembangan aplikasi Android menjadi penting karena memberikan manfaat yang berdampak besar dalam memisahkan logika bisnis dengan logika tampilan. Oleh karena itu, penerapan MVVM memudahkan akses data sehingga dapat berpengaruh dalam peningkatan kualitas dan pemeliharaan aplikasi Android secara keseluruha[6]

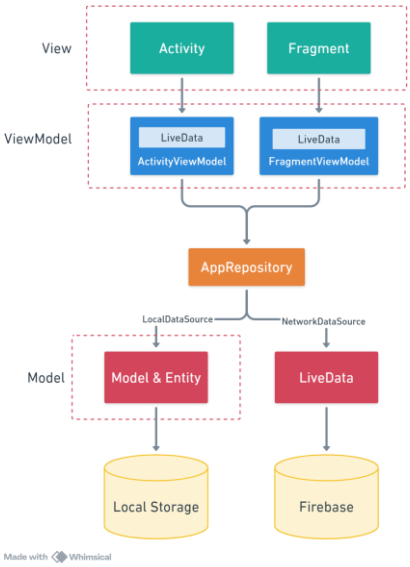
Universitas Darussalam Gontor Ponorogo memiliki sistem asrama yang digunakan sebagai pusat pembelajaran yang efektif dan efisien. Namun, terdapat tantangan bagi mahasiswa Non KMI karena banyak dari mereka yang belum mengerti bagaimana berbahasa Arab dengan baik dan benar. Oleh karena itu, penting untuk menyediakan alat bantu yang dapat membantu mereka dalam memahami materi *Shorof* dengan lebih baik.

Penelitian ini bertujuan untuk mengembangkan aplikasi pembelajaran Shorof berbasis mobile Android dengan pola arsitektur Model-View-View Model (MVVM) untuk membantu mahasiswa dalam mempelajari ilmu *Shorof* dasar. Aplikasi ini diharapkan dapat meningkatkan pemahaman mahasiswa dalam mempelajari ilmu Shorof, sehingga dapat meningkatkan keterampilan khususnya dalam berbahasa Arab. Dengan adanya aplikasi ini, diharapkan proses pembelajaran menjadi lebih efektif dan fleksibel.

2. Metode Penelitian

Metode pada penelitian ini menggunakan *Model-View-ViewModel* (MVVM) sebagai pendekatan dalam pengembangannya. MVVM adalah arsitektur perangkat lunak yang memisahkan logika bisnis dan antarmuka pengguna, memungkinkan pengembangan yang lebih terstruktur dan terorganisir.[7]

MVVM memisahkan kode ke dalam tiga komponen utama: Model, View, dan ViewModel. Model berisi data dan logika bisnis aplikasi, memastikan bahwa semua data yang dibutuhkan aplikasi dikelola dengan baik. View merupakan antarmuka pengguna yang menampilkan data dari ViewModel, sehingga pengguna dapat berinteraksi dengan aplikasi secara visual. ViewModel berfungsi sebagai penghubung antara Model dan View, menangani logika aplikasi dan interaksi pengguna, serta memastikan bahwa data dari Model dapat diakses dan ditampilkan oleh View dengan benar.[8] Desain implementasi MVVM dapat dilihat pada Gambar 1.



Gambar 1. Desain Implementasi MVVM

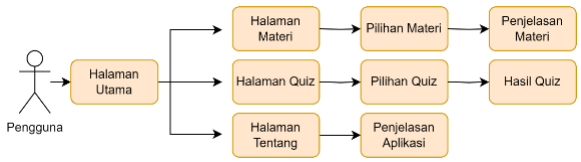
Kategori *view* meliputi *Activity* dan *fragment.* Didalam kategori *ViewModel* mencakup *ActivityViewModel* dan *FragmentViewModel*, kedua *ViewModel* ini mengimplementasikan *LiveData* yang bertujuan untuk mengatur data yang akan ditampilkan nantinya. Pada *AppRepository* bertujuan untuk menangani koneksi ke database aplikasi yang bersumber dari data local maupun jaringan. Kategori model mencakup bagian yang mengelola data local dan pada bagian *NetworkDataSource* menangani pengambilan data dari *firebase*[9]

Pada penelitian ini penulis memulai dengan menganalisis pemahaman mahasiswa Universitas Darussalam Gontor Ponorogo tentang Shorof. Hasil kuesioner menunjukkan banyak minat dalam

pembelajaran Shorof, namun kurangnya media pembelajaran memadai. Pengguna mendukung adanya media pembelajaran Shorof dasar berbasis mobile untuk membantu pemahaman dasar.

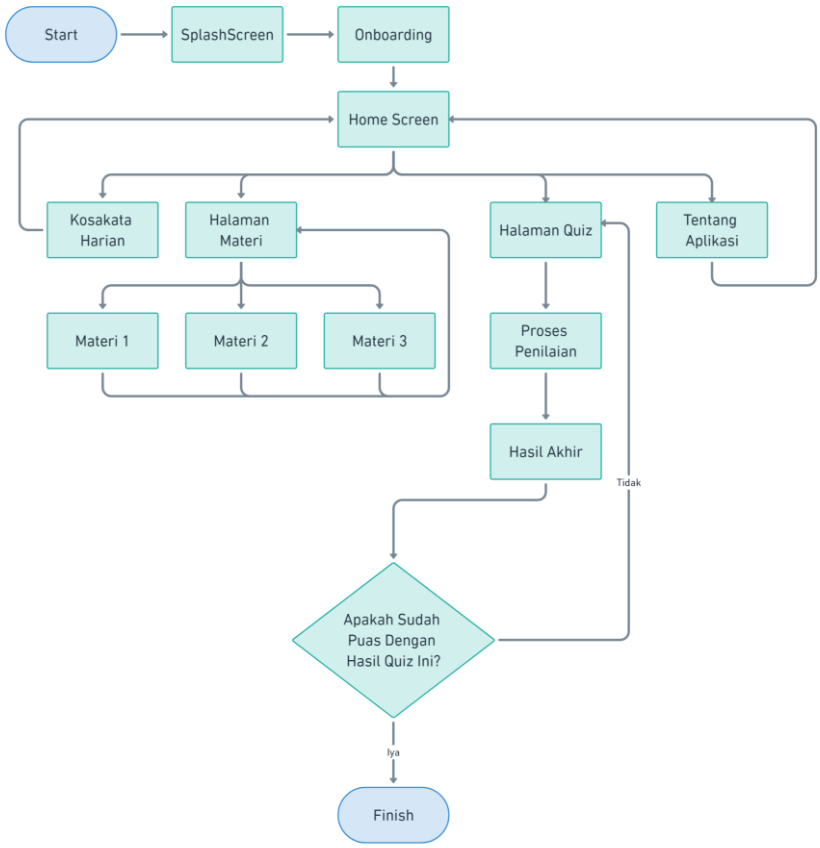
Pada tahapan selanjutnya, penulis merancang tahapan desain aplikasi meliputi perancangan sistem, dan desain aplikasi, seperti flowchart dan use case diagram, proses ini berguna untuk membantu mempermudah dalam tahapan pembuatan aplikasi.[10]

Use case diagram menunjukkan tiga menu utama: Materi, Kuis, dan Tentang Aplikasi. Untuk *use case diagram* dapat dilihat pada Gambar 2.



Gambar 2. *Use Case Diagram* Aplikasi

Flowchart menggambarkan alur aplikasi dari splash screen ke halaman utama, dengan fitur kosakata harian, materi, kuis, dan tentang aplikasi. Untuk *flowchart* dapat dilihat pada Gambar 3.



Gambar 3. *Flowchart* Aplikasi

Pada penelitian ini penulis menggunakan android studio dalam pembuatan aplikasi dan menggunakan bahasa pemrograman kotlin. Dalam pengembangan aplikasi ini menggunakan bahan ajar buku "Ilmu Shorof Dasar Untuk Pemula" sebagai materi yang akan dipelajari.

Pada proses pengujian, penulis melakukan pengujian terhadap pengguna melalui kuesioner sebagai penilaian. pengguna aplikasi ini merupakan mahasiswa Universitas Darussalam Gontor non-KMI. Pada tahapan pengujian penulis menggunakan *blackbox testing* dan *hardware testing*. Pada *blackbox testing* ini bertujuan untuk memastikan fungsionalitas tanpa memeriksa stuktur internal,[11] sedangkan pada *hardware testing* bertujuan untuk memastikan aplikasi dapat berjalan baik di berbagai perangkat Pada tahapan terakhir peneliti melakukan tahapan pemeliharaan, tahapan ini melibatkan evaluasi ulang dan perbaikan bedasarkan kesalahan dan *bug* yang terdeteksi

3. Hasil dan Pembahasan

3.1 Implementasi MVVM Pada Halaman Utama

Pada kelas `HomeViewModel`, yang tertera pada Gambar 4 menjelaskan kelas ini mengatur data kosakata menggunakan *Firebase Firestore*. Kelas ini memiliki dua *LiveData*: `_kosakataList` untuk menyimpan daftar kosakata yang dapat diperbarui secara internal dan kosakataList yang dapat diakses dari luar. Fungsi

`fetchKosakata` dipanggil untuk mengambil data dari koleksi "Kosakata" di *Firestore*.

Fungsi `fetchKosakata` mendengarkan perubahan pada koleksi "Kosakata", mengurutkan berdasarkan "Terjemahan", dan memperbarui `_kosakataList` dengan data yang diambil.

```
class HomeViewModel : ViewModel() {

    private val db = FirebaseFirestore.getInstance()
    private val _kosakataList = MutableLiveData<List<Kosakata>>()
    val kosakataList: LiveData<List<Kosakata>> = _kosakataList

    init {
        fetchKosakata()
    }

    private fun fetchKosakata() {

 db.collection("Kosakata").orderBy("Terjemahan",Query.Direction.ASCENDI
NG)
//            .limit(5) // Batasi jumlah data yang diambil menjadi 4
            .addSnapshotListener { value, error ->
                if (error != null) {
                    // Handle error
                    return@addSnapshotListener
                }
                val list = mutableListOf<Kosakata>()
                value?.documents?.forEach { document ->
                    val kosakata =
document.toObject(Kosakata::class.java)
                    kosakata?.let { list.add(it) }
                }
                _kosakataList.value = list
            }
    }
}
```

Gambar 4. *Impelementasi MVVM Pada Halaman Utama*

Berikut untuk tampilan di halaman utama yang tertera pada Gambar 5.

Gambar 5. Tampilan Halaman Utama

3.2 Implementasi MVVM Pada Halaman Materi

Pada kelas `MateriViewModel`, yang tertera pada Gambar 6 menjelaskan kelas ini mengelola data materi menggunakan *Firebase Firestore*. Kelas ini memiliki dua *LiveData*: `_materiList` untuk menyimpan daftar materi yang dapat diperbarui secara internal dan `materiList` yang dapat diakses dari luar. Saat inisialisasi, fungsi `fetchMateriData` dipanggil untuk mengambil data dari koleksi "Materi" di *Firestore*.

Pada fungsi `fetchMateriData` menggunakan `addSnapshotListener` untuk mengetahui perubahan pada koleksi "Materi", mengurutkan berdasarkan "totalPembahasan", dan memperbarui `_materiList` dengan data yang diambil. Jika terjadi kesalahan, fungsi ini akan mengabaikannya. Data yang diambil diubah menjadi objek Materi dan ditambahkan ke dalam daftar `materiArrayList`, yang kemudian diatur sebagai nilai *Live*

```
class MateriViewModel : ViewModel() {

    private val _materiList = MutableLiveData<List<Materi>>()
    val materiList: LiveData<List<Materi>> get() = _materiList
    private val db = FirebaseFirestore.getInstance()

    init {
        fetchMateriData()
    }

    private fun fetchMateriData() {

 db.collection("Materi").orderBy("totalPembahasan",Query.Direction.ASCENDING)
            .addSnapshotListener(object :EventListener<QuerySnapshot>{
                override fun onEvent(value: QuerySnapshot?, error:
FirebaseFirestoreException?) {
                    if (error !=null){
                        return
                    }
                    val materiArrayList = arrayListOf<Materi>()
                    for (dc:DocumentChange in value?.documentChanges!!){
                        if (dc.type == DocumentChange.Type.ADDED){

 materiArrayList.add(dc.document.toObject(Materi::class.java))
                        }
                    }
                    _materiList.value = materiArrayList
                }

            })
    }
}
```

Gambar 6. Impelementasi MVVM Pada Halaman Materi

Berikut untuk tampilan di halaman materi yang tertera pada Gambar 7.

Gambar 7. Halaman Materi

3.3 Implementasi MVVM Pada Halaman Detail Materi

Pada kelas `DetailViewModel` yang tertera pada Gambar 8 menjelaskan data sub-materi dikelola dengan menggunakan *Firebase Firestore*. Fungsi

loadSubMateri berfungsi untuk mengambil dokumen dari koleksi "subMateri" di *Firestore* berdasarkan judul yang diberikan. Dokumen tersebut kemudian diubah menjadi objek MateriDetail dan hasilnya disimpan dalam *LiveData* *_subMateriList*. Apabila terjadi kesalahan selama proses pengambilan data, fungsi ini akan mengabaikannya dan tidak melakukan perubahan pada data yang ada.

```
class DetailViewModel : ViewModel() {

    private val _subMateriList = MutableLiveData<List<MateriDetail>>()
    val subMateriList: LiveData<List<MateriDetail>> get() = _subMateriList

    private val db = FirebaseFirestore.getInstance()

    fun loadSubMateri(judul: String?) {

        db.collection("subMateri").whereEqualTo("judul", judul).get()
            .addOnSuccessListener { documents ->
                val subMateriArrayList = arrayListOf<MateriDetail>()
                for (document in documents) {
                    subMateriArrayList.add(document.toObject(MateriDetail::class.java))
                }
                _subMateriList.value = subMateriArrayList
            }
            .addOnFailureListener { exception ->
                // Handle error
            }
    }
}
```

Gambar 8. Impelementasi MVVM Pada Halaman Detail Materi

Berikut untuk tampilan di halaman detail materi yang tertera pada Gambar 9 yang menampilkan pembagian dari setiap bab.

Gambar 9. Isi Materi

3.4 Implementasi MVVM Pada Halaman Latihan Soal

Pada kelas *QuizViewModel* yang tertera pada Gambar 10 menjelaskan data kuis dikelola menggunakan *Firebase Realtime Database*. Fungsi *getDataFromFirebase* mengambil data dari koleksi "quizzes" di *Firebase*, data yang diambil lalu diubah menjadi objek *QuizModel*, dan memperbarui *LiveData* *_quizModelList*. Status pemuatan data dilacak dengan *LiveData* *_loading*, lalu kesalahan ditangani serta disimpan dalam *LiveData* *_error*.

```
class QuizViewModel:ViewModel() {
    private val _quizModelList = MutableLiveData<List<QuizModel>>()
    val quizModelList: LiveData<List<QuizModel>> get() = _quizModelList

    private val _loading = MutableLiveData<Boolean>()
    val loading:LiveData<Boolean>get() = _loading

    private val _error = MutableLiveData<String>()
    val error : LiveData<String>get() = _error

    init {
        getDataFromFirebase()
    }

    private fun getDataFromFirebase() {
        _loading.value = true
        FirebaseDatabase.getInstance().reference.child("quizzes")
            .get()
            .addOnSuccessListener { dataSpapshot->
                if (dataSpapshot.exists()){
                    val tempList = mutableListOf<QuizModel>()
                    for (snapshot in dataSpapshot.children){
                        try {
                            val quizModel = snapshot.getValue(QuizModel::class.java)
                            if (quizModel != null){
                                tempList.add(quizModel)
                            }
                        }catch (e:DatabaseException){
                            _error.value = "Failed To Convert Value"
                        }
                    }
                    _quizModelList.value = tempList
                }
                _loading.value = false
            }
            .addOnFailureListener{exception->
                _error.value = "error getting data"
                _loading.value = false
            }
    }
}
```

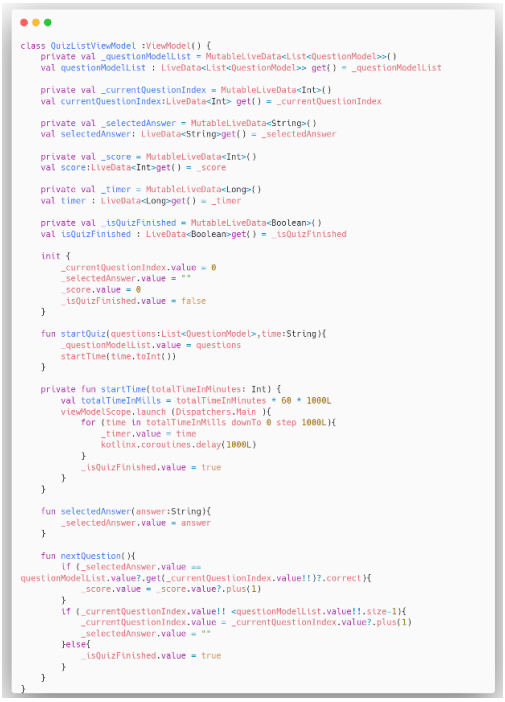
Gambar 10. Impelementasi MVVM Pada Halaman Latihan Soal

Berikut untuk tampilan di halaman Latihan soal yang tertera pada Gambar 11 yang menampilkan pembagian dari soal-soal.

Gambar 11. Halaman Daftar Soal

3.5 Implementasi MVVM Pada Halaman Latihan Soal

Pada kelas QuizListViewModel yang tertera di Gambar 12 menjelaskan data kuis dikelola secara efektif untuk memberikan pengalaman belajar yang optimal. Fungsi-fungsi utama mencakup pengaturan daftar pertanyaan, pemantauan jawaban yang dipilih, penghitungan skor, pengaturan timer, dan menentukan apakah kuis telah selesai.



```
class QuizListViewModel :ViewModel() {
    private val _questionModelList = MutableLiveData<List<QuestionModel>>()
    val questionModelList : LiveData<List<QuestionModel>> get() = _questionModelList

    private val _currentQuestionIndex = MutableLiveData<Int>()
    val currentQuestionIndex:LiveData<Int> get() = _currentQuestionIndex

    private val _selectedAnswer = MutableLiveData<String>()
    val selectedAnswer: LiveData<String>get() = _selectedAnswer

    private val _score = MutableLiveData<Int>()
    val score:LiveData<Int>get() = _score

    private val _timer = MutableLiveData<Long>()
    val timer : LiveData<Long>get() = _timer

    private val _isQuizFinished = MutableLiveData<Boolean>()
    val isQuizFinished : LiveData<Boolean>get() = _isQuizFinished

    init {
        _currentQuestionIndex.value = 0
        _selectedAnswer.value = ""
        _score.value = 0
        _isQuizFinished.value = false
    }

    fun startQuiz(questions:List<QuestionModel>,time:String){
        _questionModelList.value = questions
        startTime(time.toInt())
    }

    private fun startTime(totalTimeInMinutes: Int) {
        val totalTimeInMills = totalTimeInMinutes * 60 * 1000L
        viewModelScope.launch (Dispatchers.Main ){
            for (time in totalTimeInMills downTo 0 step 1000L){
                _timer.value = time
                kotlinx.coroutines.delay(1000L)
            }
            _isQuizFinished.value = true
        }
    }

    fun selectedAnswer(answer:String){
        _selectedAnswer.value = answer
    }

    fun nextQuestion(){
        if (_selectedAnswer.value ==
questionModelList.value?.get(_currentQuestionIndex.value!!)?.correct){
            _score.value = _score.value?.plus(1)
        }
        if (_currentQuestionIndex.value!! <questionModelList.value!!.size-1){
            _currentQuestionIndex.value = _currentQuestionIndex.value?.plus(1)
            _selectedAnswer.value = ""
        }else{
            _isQuizFinished.value = true
        }
    }
}
```

Gambar 12. Impelementasi MVVM Pada Halaman Daftar Soal

Berikut untuk tampilan di halaman Latihan soal yang tertera pada Gambar 13 yang menampilkan pembagian dari soal-soal.

Gambar 13. Halaman Latihan Soal

3.6 Imlementasi MVVM Pada Halaman Kosakata

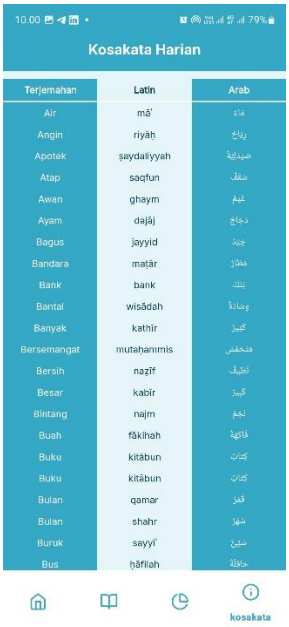
Pada kelas `KosakataViewModel` yang tertera di Gambar 14 menjelaskan data kosakata dikelola menggunakan *Firebase Firestore* untuk memberikan akses real-time terhadap data kosakata yang diurutkan berdasarkan terjemahan. Firebase Firestore memungkinkan aplikasi untuk melakukan perubahan data secara langsung.

```
class KosakataViewModel : ViewModel() {
    private val dbKosakata = FirebaseFirestore.getInstance()
    private val _kosakataList = MutableLiveData<List<Kosakata>>()
    val kosakataList: LiveData<List<Kosakata>> = _kosakataList
    init {
        fetchKosakata()
    }

    private fun fetchKosakata() {
        dbKosakata.collection("Kosakata").orderBy("Terjemahan",Query.Direction.ASCENDING)
            .addSnapshotListener{value,error ->
                if (error != null){
                    return@addSnapshotListener
                }
                val list = mutableListOf<Kosakata>()
                value?.documents?.forEach{document ->
                    val kosakata = document.toObject(Kosakata::class.java)
                    kosakata?.let { list.add(it) }
                }
                _kosakataList.value = list
            }
    }
}
```

Gambar 14. Impelementasi MVVM Pada Halaman Kosakata

Berikut untuk tampilan di halaman kosakata yang tertera pada Gambar 15 yang menampilkan kosakata Bahasa Arab beserta terjemahan dan latinnya.



Gambar 15. Halaman Kosakata Harian

3.6 Pengujian *BlackBox*

Pada tahap ini penulis melakukan pengujian *blackbox,* pengujian ini dirancang untuk pengujian pada setiap proses. Pada tahap ini pengujian dilakukan dengan menguji langsung proses ketika pengguna menggunakan aplikasi, hasil pengujian dapat dilihat di Tabel 1.

Tabel 1. Pengujian Black Box

Halaman	Proses	Hasil
Splash Screen	Menampilkan splash screen	Berhasil
Menu Utama	Menampilkan menu utama	Berhasil
Materi Shorof	Menampilkan materi shorof	Berhasil
Latihan Soal	Menampilkan latihan soal shorof	Berhasil
Hasil Latihan	Menampilkan hasil akhir latihan soal	Berhasil
Kosakata	Menampilkan kosakata	Berhasil
Tentang Aplikasi	Menampilkan penjelasan singkat aplikasi	Berhasil

Bedasarkan hasil yang dilihat di Tabel 1 menunjukan bahwa semua fungsi pada aplikasi dapat berjalan dengan lancar, hasil ini pengujian *blackbox* ini menunjukan bahwa aplikasi yang dikembangkan berjalan dengan baik dan siap untuk dipakai oleh pengguna.

3.6 Pengujian *Hardware*

Pada tahap ini penulis melukan pengujian di beberapa perangkat smartphone berbasis android yang berbeda, pengujian ini bertujuan untuk memastikan bahwa aplikasi dapat berjalan lancar di berbagai jenis perangkat, hasil pengujian dapat dilihat di tabel 2.

Tabel 2. Pengujian Hardware

Versi Android	Ukuran Layar	Hasil
13	6,44 inci	Lancar
13	6,6 inci	Lancar
12	6,67 inci	Lancar
11	6,5 inci	Lancar

Bedasarkan hasil dari Tabel 2 menunjukan bahwa pengujian *hardware* ini berjalan dengan lancar, aplikasi ini dapat berjalan di berbagai perangkat android dari berbagai versi android hingga beberapa versi ukuran layar.

4. Kesimpulan

Aplikasi media pembelajaran *Shorof* ini merupakan aplikasi yang dapat membantu mahasiswa non-KMI Universitas Darussalam Gontor dalam pembelajaran *shorof* dasar. Aplikasi ini memiliki tiga fitur utama yaitu: pembelajaran *Shorof*, Latihan soal, dan kosakata bahasa Arab. Dari segi pengujian aplikasi ini telah diuji secara baik secara fungsi melalui pengujian *blackbox* maupun pengujian *hardware* guna memastikan aplikasi ini layak digunakan nantinya oleh pengguna.

Peneliti menyadari bahwa penelitian ini masih perlu peningkatan harapannya untuk penelitian selanjutnya dapat menambah lebih banyak lagi materi pembelajaran *shorof* dan pengembangan model Latihan soal selain pilihan ganda.

Daftar Rujukan

[1] N. Zakiah, "Problematika Pembelajaran Bahasa Arab Siswa Madrasah Tsanawiyah Al Islamiyah Kotabumi Lampung Utara," *Indones. J. Instr. Technol.*, vol. 2, no. 1, pp. 52–66, 2021.

[2] N. N. U. Tarigan and Zulkarnein, "Strategi Guru Bahasa Arab Dalam Mengatasi Kesulitan," *J. Pendidik. dan Teknol.*, vol. 3, no. 2, pp. 105–112, 2023.

[3] A. Salida and Zulpina, "Keistimewaan Bahasa Arab sebagai Bahasa Al-Quran dan Ijtihadiyyah," *Sathar J. Pendidik. Bhs. dan Sastra Arab*, vol. 1, no. 1, pp. 23–33, 2023, doi: 10.59548/js.v1i1.40.

[4] D. Ratnasari and E. M. Putra, "Pengambilan Dalil Dari Al-Qur'an Dalam Ushul Nahwu," *Pendidik. Bhs. Arab*, vol. 7, no. 1, pp. 10–21, 2023.

[5] R. P. Saputri and M. Fransisca, "Analisis Kebutuhan Siswa Terhadap Media Pembelajaran Berbasis Android Mata Pelajaran Simulasi Digital," *Semin. Nas. Terap. Ris. Inov. Ke-6*, vol. 6, no. 1, pp. 902–909, 2020.

[6] A. R. Fajri, "Penerapan Design Pattern MVVM dan Clean Architecture pada Pengembangan Aplikasi Android (Studi Kasus: Aplikasi Agree Partner)," 2022.

[7] Y. Mose, A. C. Andaria, P. Y. Damongi, H. Hendrawan, P. Egeten, and I. Rindorindo, "Pengembangan Aplikasi Sistem Informasi Kecerdasan Moral (SICEMOR) Berbasis Android," *MALCOM Indones. J. Mach. Learn. Comput. Sci.*, vol. 4, no. 1, pp. 130–139, 2024, doi: 10.57152/malcom.v4i1.1013.

[8] A. H. Zakaria and I. K. D. Nuryana, "MVVM Perangkat Lunak Android dan Analisis Arsitektur MVP dengan Studi Kasus Aplikasi iTourism," *J. Informatics Comput. Sci.*, vol. 4, no. 4, pp. 351–357, 2023.

[9] M. S. Arif, A. Musthafa, and D. Muriyatmoko, "Implementation of Model-View-ViewModel (MVVM) Architecture Pattern in the Sistem Informasi Akademik UNIDA Gontor Mobile Application," in *Proceeding International Conference on Science and Engineering*, 2020, vol. 3, pp. 283–289. doi: 10.14421/icse.v3.514.

[10] R. Hafsari, E. Aribe, and N. Maulana, "Perancangan Sistem Informasi Manajemen Inventori Dan Penjualan Pada Perusahaan Pt.Inhutani V," *PROSISKO J. Pengemb. Ris. dan Obs. Sist. Komput.*, vol. 10, no. 2, pp. 109–116, 2023, doi: 10.30656/prosisko.v10i2.7001.

[11] N. Fitriani, Apriansyah, and H. Dedi, "Analisis Manfaat Penggunaan Aplikasi PLN Mobile Menggunakan Pengujian Sistem Black Box," *J. Multidisiplin Saintek*, vol. 3, no. 1, pp. 21–26, 2024.