LAMPIRAN

Lampiran 1 : Listning Program

1. Service API

```
interface MyApiService {
    companion object {
        operator fun invoke(
            networkConnectionInterceptor:
NetworkConnectionInterceptor
        ): MyApiService {
            val interceptor = HttpLoggingInterceptor()

 interceptor.setLevel(HttpLoggingInterceptor.Level.BODY)
            val okkHttpclient = OkHttpClient.Builder()
                .connectTimeout(1, TimeUnit.MINUTES)
                .readTimeout(1, TimeUnit.MINUTES)
                .writeTimeout(1, TimeUnit.MINUTES)
                .addInterceptor(interceptor)
                .addInterceptor(networkConnectionInterceptor)
                .build()


            return Retrofit.Builder()
                .client(okkHttpclient)

.baseUrl("https://redmangaonline.000webhostapp.com/api/public/"
)

.addConverterFactory(GsonConverterFactory.create())
                .build()
                .create(MyApiService::class.java)
        }
    }

    //PENGUNJUNG
    @GET("manga")
    suspend fun getManga(): Response<ResultManga>

    @GET("chapter_by_manga/{id}")
    suspend fun getChapterMangaByID(@Path("id") id: Int):
Response<ResultChapter>

    @GET("detail/{id}")
    suspend fun getReadMangaByID(@Path("id") id: Int):
Response<ResultChapter>

    @POST("manga_filter")
    suspend fun inpSearching(
        @Body params: MutableMap<String, String>
    ): Response<ResultPostSearching>

    //ADMIN
    @POST("login")
```

```
    suspend fun loginRequest(
        @Body params: MutableMap<String, String>
    ): Response<ResultLogin>

    @GET("a_kategori")
    suspend fun getKategori(@Query("token") token: String):
Response<ResultKategori>

    @POST("a_kategori/insert")
    suspend fun addKategori(
        @Body params: MutableMap<String, String>,
        @Query("token") token: String
    ): Response<ResultPostKategori>

    @PATCH("a_kategori/update")
    suspend fun updateKategori(
        @Body params: MutableMap<String, String>,
        @Query("token") token: String
    ): Response<ResultPostKategori>

    @DELETE("a_kategori/delete/{id}")
    suspend fun deleteKategori(
        @Path("id") id: Int,
        @Query("token") token: String
    ): Response<ResultPostKategori>

    @GET("a_manga")
    suspend fun getManga(@Query("token") token: String):
Response<ResultManga>

    @POST("a_manga/insert")
    suspend fun addManga(
        @Query("token") token: String,
        @Body manga: RequestBody
    ): Response<ResultPostManga>

    @PATCH("a_manga/update")
    suspend fun updateManga(
        @Body params: MutableMap<String, String>,
        @Query("token") token: String
    ): Response<ResultPostManga>

    @DELETE("a_manga/delete/{id}")
    suspend fun deleteManga(
        @Path("id") id: Int,
        @Query("token") token: String
    ): Response<ResultPostManga>

    @GET("a_chapter_by_manga/{id}")
    suspend fun getChapter(@Path("id") id_manga: String):
Response<ResultChapter>

    @POST("a_chapter/insert")
    suspend fun addChapter(
```

```
        @Body params: MutableMap<String, String>,
        @Query("token") token: String
    ): Response<ResultPostChapter>

    @PATCH("a_chapter/update")
    suspend fun updateChapter(
        @Body params: MutableMap<String, String>,
        @Query("token") token: String
    ): Response<ResultPostChapter>

    @DELETE("a_chapter/delete/{id}")
    suspend fun deleteChapter(
        @Path("id") id: Int,
        @Query("token") token: String
    ): Response<ResultPostChapter>

    @GET("a_detail/{id}")
    suspend fun getKomik(
        @Path("id") id: Int,
        @Query("token") token: String
    ): Response<ResultChapter>

    @POST("a_detail/upload")
    suspend fun addKomik(
        @Body komik: RequestBody,
        @Query("token") token: String
    ): Response<ResultPostKomik>

    @DELETE("a_detail/delete/{id}")
    suspend fun deleteKomik(
        @Path("id") id: Int,
        @Query("token") token: String
    ): Response<ResultPostKomik>
}
```

2. Halaman Depan

```
class MangaViewModel(
    repository: MangaRepository,
    context: Context
) : ViewModel() {
    var dao: MangaDao =
AppDatabase.invoke(context).getMangaDao()
    val repo = repository

    val manga by lazyDeferred {
        repository.getManga()
    }

    val pagedListLiveDataNewManga: LiveData<PagedList<Manga>>
by lazy {
        val dataSourceFactory = dao.getNewManga()
        val config =
PagedList.Config.Builder().setPageSize(10).build()
```

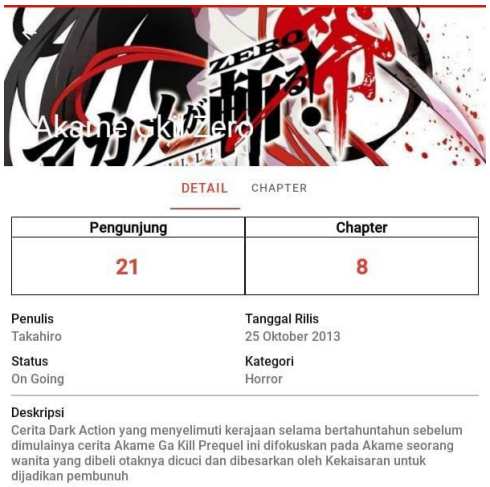
```
        LivePagedListBuilder(dataSourceFactory, config).build()
    }

    val pagedListLiveDataMostViewManga:
LiveData<PagedList<Manga>> by lazy {
        val dataSourceFactory = dao.getMostViewManga()
        val config =
PagedList.Config.Builder().setPageSize(10).build()
        LivePagedListBuilder(dataSourceFactory, config).build()
    }

    val pagedListLiveDataLastRealeaseManga:
LiveData<PagedList<Manga>> by lazy {
        val dataSourceFactory = dao.getLastestRealeasManga()
        val config =
PagedList.Config.Builder().setPageSize(10).build()
        LivePagedListBuilder(dataSourceFactory, config).build()
    }

    suspend fun loginRequest(username: String, password:
String): ResultLogin {
        return repo.fetchLogin(username, password)
    }

    fun deleteAllManga(): Int {
        return dao.deleteAllManga()
    }

    val s = MutableLiveData<PagedList<Manga>>()

    fun pencarianManga(judul: String, kategori: String) {
        GlobalScope.launch(Dispatchers.IO) {
            val dataSourceFactory = dao.getResManga(judul,
kategori)
            val config =
PagedList.Config.Builder().setPageSize(10).build()

            val result = PagedList.Builder(dataSourceFactory,
config)

.setFetchExecutor(Executors.newSingleThreadExecutor()) // Untuk
memastikan operasi data berjalan di thread terpisah
                .setNotifyExecutor(MainThreadExecutor()) //
Pastikan notifikasi perubahan berjalan di thread utama
                .build()

            // Kirim hasil ke thread utama
            withContext(Dispatchers.Main) {
                s.value = result
            }
        }
    }

}
```

Output :

3. Detail Manga

```
class DetailViewModel(
    repository: MangaRepository
) : ViewModel() {
    val repo = repository
    var data = MutableLiveData<Manga>()
    var read = repo.allRead

    val chapter by lazyDeferred {
        repository.getChapter(data.value!!.id_manga)
    }

    val reader by lazyDeferred {
        repository.getReader(data.value!!.id_manga)
    }

    val listChapter = repository.chapter

    fun saveRead(reader: Reader) {
        repo.saveReader(reader)
    }
}
```

Output :

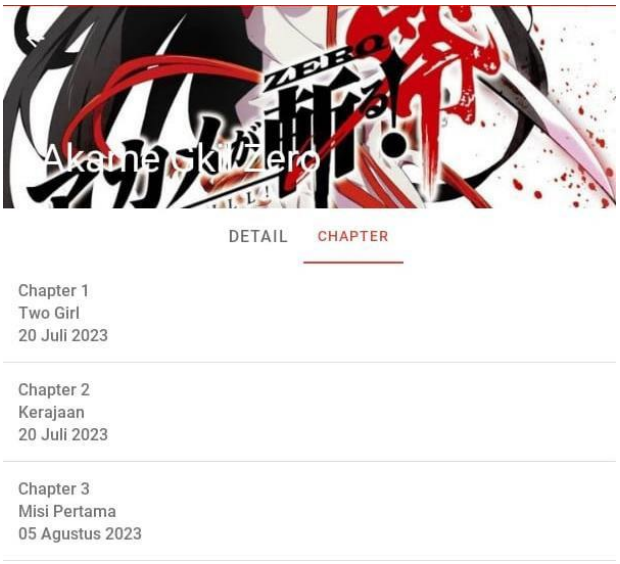


4. Detail Chapter

```
class ChapterAdapter(private val data: List<Chapter>, private
val read: List<Int>) :
    RecyclerView.Adapter<ChapterAdapter.ViewHolder>() {
    private var onItemClick: OnItemClick? = null

    fun ItemClick(onItemClick: OnItemClick?) {
        this.onItemClick = onItemClick
    }

    override fun onCreateViewHolder(
        parent: ViewGroup,
        viewType: Int
    ): ViewHolder {
        val view: View =

LayoutInflater.from(parent.context).inflate(R.layout.item_rv_ch
apter, parent, false)
        return ViewHolder(view)
    }

    override fun onBindViewHolder(
        holder: ViewHolder,
        position: Int
    ) {
        holder.bind(data[position])
    }

    override fun getItemCount(): Int {
        return data.size
    }

    inner class ViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        private var tvChappter: TextView = itemView.chapter
        private var tvJudul: TextView = itemView.judul_chapter
        private var tvTanggal: TextView =
itemView.tanggal_release
        fun bind(item: Chapter) {
            if(read.contains(item.id_chapter)){

tvChappter.setTextColor(ContextCompat.getColor(itemView.context
,R.color.grey_600))
                tvChappter.text =

itemView.resources.getString(R.string.chapter,
item.chapter.toString())
                tvJudul.text = item.judul_chapter

tvJudul.setTextColor(ContextCompat.getColor(itemView.context,R.
color.grey_600))
                tvTanggal.text = convertDate(item.tanggal, 2)

tvTanggal.setTextColor(ContextCompat.getColor(itemView.context,
```

```
 R.color.grey_600))
             }else{
                 tvChappter.text =

itemView.resources.getString(R.string.chapter,
item.chapter.toString())
                 tvJudul.text = item.judul_chapter
                 tvTanggal.text = convertDate(item.tanggal, 2)
             }

             itemView.setOnClickListener {

tvChappter.setTextColor(ContextCompat.getColor(itemView.context
,R.color.grey_600))
                 tvChappter.text =

itemView.resources.getString(R.string.chapter,
item.chapter.toString())
                 tvJudul.text = item.judul_chapter

tvJudul.setTextColor(ContextCompat.getColor(itemView.context,R.
color.grey_600))
                 tvTanggal.text = convertDate(item.tanggal, 2)

tvTanggal.setTextColor(ContextCompat.getColor(itemView.context,
R.color.grey_600))
                 notifyDataSetChanged()
                 onItemClick!!.onItemClicked(item)
             }
         }

     }

     interface OnItemClick {
         fun onItemClicked(item: Chapter?)
     }

 }
```

Output :

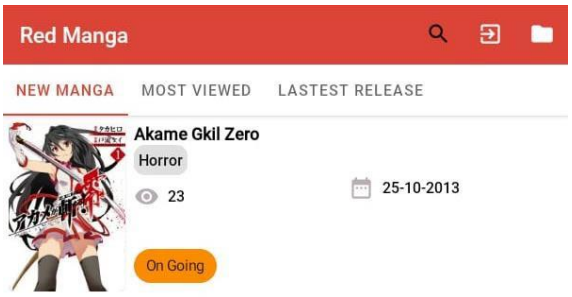


5. Dasbord Admin

```
class AdminViewModel(
    repository: AdminRepository
) : ViewModel() {
    private val repo = repository
    var kategori = MutableLiveData<List<Kategori>>()
    var manga = MutableLiveData<List<Manga>>()
    var chapter = MutableLiveData<List<Chapter>>()
    var komik = MutableLiveData<List<Chapter>>()

    suspend fun getKategori() {
        kategori.postValue(repo.getKategori())
    }

    suspend fun addKategori(nama: String): ResultPostKategori {
        return repo.addKategori(nama)
    }

    suspend fun updateKategori(id: Int, nama: String):
ResultPostKategori {
        return repo.updateKategori(id.toString(), nama)
    }

    suspend fun deleteKategori(id: Int): ResultPostKategori {
        return repo.deleteKategori(id)
    }

    suspend fun getManga() {
        manga.postValue(repo.getManga())
    }

    suspend fun addManga(
        manga: RequestBody
    ): ResultPostManga {
        return repo.addManga(manga)
```

```
    }

    suspend fun updateManga(
        id: String,
        id_kategori: String,
        judul: String,
        tanggal: String,
        deskripsi: String,
        penulis: String,
        status: String
    ): ResultPostManga {
        return repo.updateManga(id, id_kategori, judul,
tanggal, deskripsi, penulis, status)
    }

    suspend fun deleteManga(id: Int): ResultPostManga {
        return repo.deleteManga(id)
    }

    suspend fun getChapter(id_manga: String) {
        chapter.postValue(repo.getChapter(id_manga))
    }

    suspend fun addChapter(id_manga:String,chapter:
String,judul: String,tanggal: String): ResultPostChapter {
        return repo.addChapter(id_manga,chapter,judul,tanggal)
    }

    suspend fun updateChapter(id_chapter: String, id_manga:
String,chapter: String,judul: String): ResultPostChapter {
        return
repo.updateChapter(id_chapter,id_manga,chapter,judul)
    }

    suspend fun deleteChapter(id: Int): ResultPostChapter {
        return repo.deleteChapter(id)
    }

    suspend fun getKomik(id_chapter:Int) {
        komik.postValue(repo.getKomik(id_chapter))
    }

    suspend fun addKomik(komik: RequestBody): ResultPostKomik {
        return repo.addKomik(komik)
    }

    suspend fun deleteKomik(id: Int): ResultPostKomik {
        return repo.deleteKomik(id)
    }
}
```

Output :



6. Tambah Manga

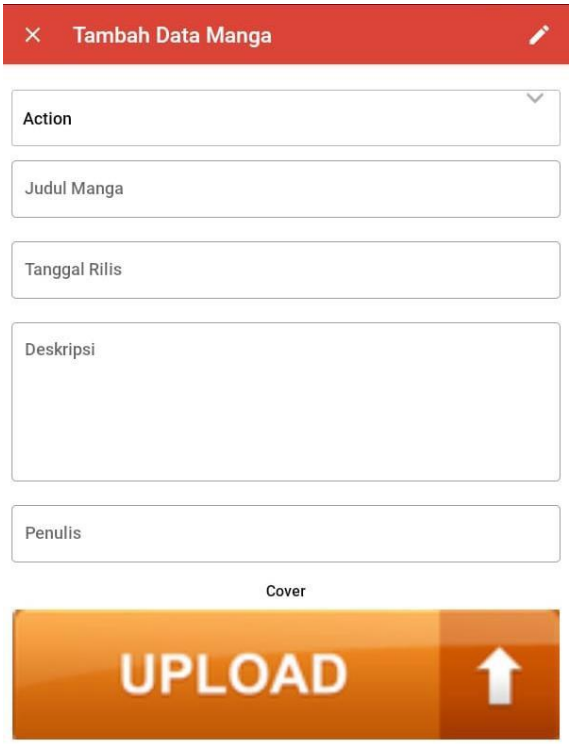
```
class MangaActivity : AppCompatActivity(), KodeinAware {

    override val kodein by kodein()

    private lateinit var adminViewModel: AdminViewModel
    private val factory: AdminViewModelFactory by instance()
    val kategoriList: MutableList<Kategori> = mutableListOf()
    lateinit var adapter: MangaAdminAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val binding: ActivityMangaBinding =
ActivityMangaBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setSupportActionBar(binding.topBar.toolbar)
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
        supportActionBar?.title = "List Manga"

        adminViewModel = ViewModelProvider(this,
factory).get(AdminViewModel::class.java)

        getManga()
        getKategori()

        adminViewModel.kategori.observe(this, Observer {
            it?.let {
                kategoriList.clear()
                kategoriList.addAll(it)
            }
        })

        val mangaList: MutableList<Manga> = mutableListOf()
        adapter = MangaAdminAdapter(mangaList)
        binding.rvManga.layoutManager =
LinearLayoutManager(this)
        val dividerItemDecoration = DividerItemDecoration(
            binding.rvManga.context,
            LinearLayoutManager.VERTICAL
        )

 binding.rvManga.addItemDecoration(dividerItemDecoration)
```

```
        binding.rvManga.setHasFixedSize(true)
        binding.rvManga.adapter = adapter

        adminViewModel.manga.observe(this, Observer {
            mangaList.clear()
            mangaList.addAll(it)
            adapter.notifyDataSetChanged()
        })

        adapter.ItemClick(object :
MangaAdminAdapter.OnItemClick {
            override fun onItemClicked(item: Manga?) {
                val intent = Intent(this@MangaActivity,
ChapterActivity::class.java)
                intent.putExtra(ChapterActivity.EXTRA_DATA,
item)
                startActivity(intent)
            }

            override fun onLongItemClicked(item: Manga?) {
                val fragment = FragmentBottomSheet(
                    kategoriList = kategoriList,
                    manga = item,
                    layout = 2,
                    type = 1
                )
                fragment.show(supportFragmentManager,
fragment.tag)

supportFragmentManager.registerFragmentLifecycleCallbacks(objec
t :

FragmentManager.FragmentLifecycleCallbacks() {
                    override fun onFragmentViewDestroyed(fm:
FragmentManager, f: Fragment) {
                        super.onFragmentViewDestroyed(fm, f)
                        getManga()

supportFragmentManager.unregisterFragmentLifecycleCallbacks(thi
s)
                    }
                }, false)
            }

            override fun onItemDeleteClicked(item: Manga?,
posisi: Int) {
                deleteDialog(posisi, item)
            }

        })
    }

    override fun onSupportNavigateUp(): Boolean {
        onBackPressed()
```

```
        return super.onSupportNavigateUp()
    }

    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menuInflater.inflate(R.menu.admin_add_data, menu)
        return true
    }

    private fun getKategori() = Coroutines.main {
        adminViewModel.getKategori()
    }

    private fun getManga() = Coroutines.main {
        adminViewModel.getManga()
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean
{
        if (item.itemId == R.id.tambah) {
            val fragment = FragmentBottomSheet(kategoriList =
kategoriList, layout = 2)
            fragment.show(supportFragmentManager, fragment.tag)

supportFragmentManager.registerFragmentLifecycleCallbacks(objec
t :
                FragmentManager.FragmentLifecycleCallbacks() {
                override fun onFragmentViewDestroyed(fm:
FragmentManager, f: Fragment) {
                    super.onFragmentViewDestroyed(fm, f)
                    getManga()

supportFragmentManager.unregisterFragmentLifecycleCallbacks(thi
s)
                }
            }, false)
        }
        return super.onOptionsItemSelected(item)
    }

    private fun deleteDialog(posisi: Int, item: Manga?) {
        val builder = AlertDialog.Builder(this)
        builder.setTitle("Informasi")
        builder.setMessage("Apakah anda ingin menghapus manga
${item?.judul}?")
        builder.setPositiveButton("YA") { _, _ ->
            Coroutines.main {
                val result =
adminViewModel.deleteManga(item!!.id_manga)
                showToast(this@MangaActivity, result.message)
                if (result.message == "berhasil_hapus_data") {
                    adapter.deleteItem(posisi)
                }
            }
        }
```

```
        builder.setNegativeButton("TIDAK") { dialogInterface, _
->
            dialogInterface.dismiss()
        }
        val dialog: AlertDialog = builder.create()
        dialog.show()
    }
}
```

Output :



7. Tambah Kategori

```
class KategoriActivity : AppCompatActivity(), KodeinAware {

    override val kodein by kodein()

    private lateinit var adminViewModel: AdminViewModel
    private val factory: AdminViewModelFactory by instance()
    lateinit var adapter: KategoriAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val binding: ActivityKategoriBinding =
ActivityKategoriBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setSupportActionBar(binding.topBar.toolbar)
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
        supportActionBar?.title = "List Kategori"
```

```
        adminViewModel = ViewModelProvider(this,
factory).get(AdminViewModel::class.java)

        getKategori()

        val kategoriList: MutableList<Kategori> =
mutableListOf()
        adapter = KategoriAdapter(kategoriList)
        binding.rvKateogri.layoutManager =
LinearLayoutManager(this)
        val dividerItemDecoration = DividerItemDecoration(
            binding.rvKateogri.context,
            LinearLayoutManager.VERTICAL
        )

binding.rvKateogri.addItemDecoration(dividerItemDecoration)
        binding.rvKateogri.setHasFixedSize(true)
        binding.rvKateogri.adapter = adapter

        adminViewModel.kategori.observe(this, Observer {
            it?.let {
                kategoriList.clear()
                kategoriList.addAll(it)
                adapter.notifyDataSetChanged()
            }
        })

        adapter.itemClick(object : KategoriAdapter.OnItemClick
{
            override fun onItemClicked(item: Kategori?) {
                val fragment = FragmentBottomSheet(kategori =
item, layout = 1, type = 1)
                fragment.show(supportFragmentManager,
fragment.tag)

supportFragmentManager.registerFragmentLifecycleCallbacks(objec
t :

FragmentManager.FragmentLifecycleCallbacks() {
                    override fun onFragmentViewDestroyed(fm:
FragmentManager, f: Fragment) {
                        super.onFragmentViewDestroyed(fm, f)
                        getKategori()

supportFragmentManager.unregisterFragmentLifecycleCallbacks(thi
s)
                    }
                }, false)
            }

            override fun onItemDeleteClicked(item: Kategori?,
posisi: Int) {
                deleteDialog(posisi, item)
```

```
            }

        })
    }

    private fun getKategori() = Coroutines.main {
        adminViewModel.getKategori()
    }

    override fun onSupportNavigateUp(): Boolean {
        onBackPressed()
        return super.onSupportNavigateUp()
    }

    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menuInflater.inflate(R.menu.admin_add_data, menu)
        return true
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean
{
        if (item.itemId == R.id.tambah) {
            val fragment = FragmentBottomSheet(layout = 1)
            fragment.show(supportFragmentManager, fragment.tag)

supportFragmentManager.registerFragmentLifecycleCallbacks(objec
t :
                FragmentManager.FragmentLifecycleCallbacks() {
                override fun onFragmentViewDestroyed(fm:
FragmentManager, f: Fragment) {
                    super.onFragmentViewDestroyed(fm, f)
                    getKategori()

supportFragmentManager.unregisterFragmentLifecycleCallbacks(thi
s)
                }
            }, false)
        }
        return super.onOptionsItemSelected(item)
    }

    private fun deleteDialog(posisi: Int, item: Kategori?) {
        val builder = AlertDialog.Builder(this)
        builder.setTitle("Informasi")
        builder.setMessage("Apakah anda ingin menghapus
kategori ${item?.nama_kategori}?")
        builder.setPositiveButton("YA") { _, _ ->
            Coroutines.main {
                val result =
adminViewModel.deleteKategori(item!!.id_kategori)
                showToast(this@KategoriActivity,
result.message)
                if (result.message == "berhasil_hapus_data") {
                    adapter.deleteItem(posisi)
```

```
                }
            }
        }
        builder.setNegativeButton("TIDAK") { dialogInterface, _
->
            dialogInterface.dismiss()
        }
        val dialog: AlertDialog = builder.create()
        dialog.show()
    }
}
```

Output :

8. Tambah Chapter

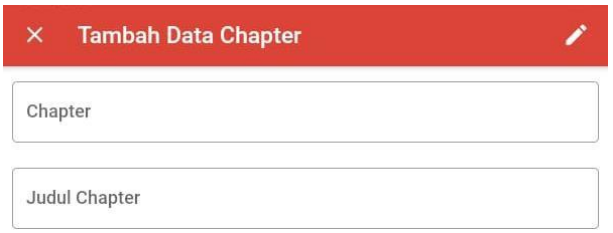
```
class ChapterActivity : AppCompatActivity(), KodeinAware {

    companion object {
        const val EXTRA_DATA = "EXTRA_DATA"
    }

    override val kodein by kodein()

    private lateinit var adminViewModel: AdminViewModel
    private val factory: AdminViewModelFactory by instance()
    lateinit var adapter: ChapterAdminAdapter
    lateinit var manga:Manga
    lateinit var idManga:String

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val binding: ActivityChapterBinding =
ActivityChapterBinding.inflate(layoutInflater)
        setContentView(binding.root)

        manga = intent.getParcelableExtra(EXTRA_DATA)!!

        setSupportActionBar(binding.topBar.toolbar)
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
        supportActionBar?.title = manga.judul

        idManga = manga.id_manga.toString()

        adminViewModel = ViewModelProvider(this,
factory).get(AdminViewModel::class.java)

        getChapter(idManga)
```

```
        val chapterList: MutableList<Chapter> = mutableListOf()
        adapter = ChapterAdminAdapter(chapterList)
        binding.rvChapter.layoutManager =
LinearLayoutManager(this)
        val dividerItemDecoration = DividerItemDecoration(
            binding.rvChapter.context,
            LinearLayoutManager.VERTICAL
        )

binding.rvChapter.addItemDecoration(dividerItemDecoration)
        binding.rvChapter.setHasFixedSize(true)
        binding.rvChapter.adapter = adapter

        adminViewModel.chapter.observe(this, Observer {
            chapterList.clear()
            chapterList.addAll(it)
            adapter.notifyDataSetChanged()
            if(it.isEmpty()){
                Toast.makeText(this,"Data
Kosong",Toast.LENGTH_SHORT).show()
            }
        })

        adapter.ItemClick(object :
ChapterAdminAdapter.OnItemClick {
            override fun onItemClicked(item: Chapter?) {
                val intent = Intent(this@ChapterActivity,
KomikActivity::class.java)
                intent.putExtra(KomikActivity.EXTRA_DATA, item)
                startActivity(intent)
            }

            override fun onLongItemClicked(item: Chapter?) {
                val fragment = FragmentBottomSheet(
                    manga = manga,
                    dataChapter = item,
                    layout = 3,
                    type = 1
                )
                fragment.show(supportFragmentManager,
fragment.tag)

supportFragmentManager.registerFragmentLifecycleCallbacks(objec
t :

FragmentManager.FragmentLifecycleCallbacks() {
                    override fun onFragmentViewDestroyed(fm:
FragmentManager, f: Fragment) {
                        super.onFragmentViewDestroyed(fm, f)
                        getChapter(idManga)

supportFragmentManager.unregisterFragmentLifecycleCallbacks(thi
s)
```

```
                    }
                }, false)
            }

            override fun onItemDeleteClicked(item: Chapter?,
posisi: Int) {
                deleteDialog(posisi, item)
            }

        })

    }

    private fun getChapter(id: String) = Coroutines.main {
        adminViewModel.getChapter(id)
    }

    override fun onSupportNavigateUp(): Boolean {
        onBackPressed()
        return super.onSupportNavigateUp()
    }

    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menuInflater.inflate(R.menu.admin_add_data, menu)
        return true
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean
{
        if (item.itemId == R.id.tambah) {
            val fragment = FragmentBottomSheet(layout = 3,
manga = manga)
            fragment.show(supportFragmentManager, fragment.tag)

supportFragmentManager.registerFragmentLifecycleCallbacks(objec
t :
                FragmentManager.FragmentLifecycleCallbacks() {
                override fun onFragmentViewDestroyed(fm:
FragmentManager, f: Fragment) {
                    super.onFragmentViewDestroyed(fm, f)
                    getChapter(idManga)

supportFragmentManager.unregisterFragmentLifecycleCallbacks(thi
s)
                }
            }, false)
        }
        return super.onOptionsItemSelected(item)
    }

    private fun deleteDialog(posisi: Int, item: Chapter?) {
        val builder = AlertDialog.Builder(this)
        builder.setTitle("Informasi")
        builder.setMessage("Apakah anda ingin menghapus chapter
```

```
${item?.judul_chapter}?")
        builder.setPositiveButton("YA") { _, _ ->
            Coroutines.main {
                val result =
adminViewModel.deleteChapter(item!!.id_chapter)
                showToast(this@ChapterActivity, result.message)
                if (result.message == "berhasil_hapus_data") {
                    adapter.deleteItem(posisi)
                }
            }
        }
        builder.setNegativeButton("TIDAK") { dialogInterface, _
->
            dialogInterface.dismiss()
        }
        val dialog: AlertDialog = builder.create()
        dialog.show()
    }
}
```

Output :



9. Tambah Data Komik(Gambar)

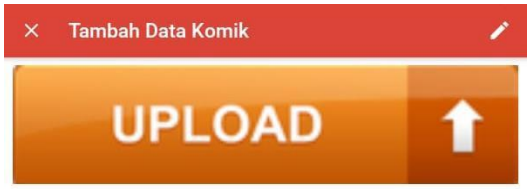
```
class KomikActivity : AppCompatActivity(),KodeinAware {

    companion object {
        const val EXTRA_DATA = "EXTRA_DATA"
    }

    override val kodein by kodein()

    private lateinit var adminViewModel: AdminViewModel
    private val factory: AdminViewModelFactory by instance()
    lateinit var adapter: KomikAdminAdapter
    lateinit var chapter: Chapter
    lateinit var idChapter:String

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val binding:ActivityKomikBinding =
ActivityKomikBinding.inflate(layoutInflater)
        setContentView(binding.root)

        chapter =
intent.getParcelableExtra(ChapterActivity.EXTRA_DATA)!!
```

```
        setSupportActionBar(binding.topBar.toolbar)
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
        supportActionBar?.title = chapter.judul_chapter

        idChapter = chapter.id_chapter.toString()

        adminViewModel = ViewModelProvider(this,
factory).get(AdminViewModel::class.java)

        getKomik(idChapter)

        val komikList: MutableList<Chapter> = mutableListOf()
        adapter = KomikAdminAdapter(komikList)
        binding.rvKomik.layoutManager =
LinearLayoutManager(this)
        val dividerItemDecoration = DividerItemDecoration(
            binding.rvKomik.context,
            LinearLayoutManager.VERTICAL
        )

binding.rvKomik.addItemDecoration(dividerItemDecoration)
        binding.rvKomik.setHasFixedSize(true)
        binding.rvKomik.adapter = adapter

        adminViewModel.komik.observe(this, Observer {
            komikList.clear()
            komikList.addAll(it)
            adapter.notifyDataSetChanged()
            if(it.isEmpty()){
                Toast.makeText(this,"Data Kosong",
Toast.LENGTH_SHORT).show()
            }
        })

        adapter.ItemClick(object :
KomikAdminAdapter.OnItemClick {
            override fun onItemClicked(item: Chapter?) {
//                val intent = Intent(this@ChapterActivity,
KomikActivity::class.java)
//                intent.putExtra(KomikActivity.EXTRA_DATA,
item)
//                startActivity(intent)
            }

            override fun onLongItemClicked(item: Chapter?) {
//                val fragment = FragmentBottomSheet(
//                    dataChapter = item,
//                    layout = 3,
//                    type = 1
//                )
//                fragment.show(supportFragmentManager,
fragment.tag)
//
```

```
supportFragmentManager.registerFragmentLifecycleCallbacks(objec
t :
//
FragmentManager.FragmentLifecycleCallbacks() {
//                    override fun onFragmentViewDestroyed(fm:
FragmentManager, f: Fragment) {
//                        super.onFragmentViewDestroyed(fm, f)
//                        getKomik(idChapter)
//
supportFragmentManager.unregisterFragmentLifecycleCallbacks(thi
s)
//                    }
//                }, false)
            }

            override fun onItemDeleteClicked(item: Chapter?,
posisi: Int) {
                deleteDialog(posisi, item)
            }

        })
    }

    private fun getKomik(id: String) = Coroutines.main {
        adminViewModel.getKomik(id.toInt())
    }
    override fun onSupportNavigateUp(): Boolean {
        onBackPressed()
        return super.onSupportNavigateUp()
    }
    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menuInflater.inflate(R.menu.admin_add_data, menu)
        return true
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean
{
        if (item.itemId == R.id.tambah) {
            val fragment = FragmentBottomSheet(layout = 4,
dataChapter = chapter)
            fragment.show(supportFragmentManager, fragment.tag)

supportFragmentManager.registerFragmentLifecycleCallbacks(objec
t :
                FragmentManager.FragmentLifecycleCallbacks() {
                override fun onFragmentViewDestroyed(fm:
FragmentManager, f: Fragment) {
                    super.onFragmentViewDestroyed(fm, f)
                    getKomik(idChapter)

supportFragmentManager.unregisterFragmentLifecycleCallbacks(thi
s)
                }
            }, false)
```

```
        }
        return super.onOptionsItemSelected(item)
    }

    private fun deleteDialog(posisi: Int, item: Chapter?) {
        val builder = AlertDialog.Builder(this)
        builder.setTitle("Informasi")
        builder.setMessage("Apakah anda ingin menghapus komik
detail ${item?.id_detail}?")
        builder.setPositiveButton("YA") { _, _ ->
            Coroutines.main {
                val result =
adminViewModel.deleteKomik(item!!.id_detail)
                showToast(this@KomikActivity, result.message)
                if (result.message == "berhasil_hapus_data") {
                    adapter.deleteItem(posisi)
                }
            }
        }
        builder.setNegativeButton("TIDAK") { dialogInterface, _
->
            dialogInterface.dismiss()
        }
        val dialog: AlertDialog = builder.create()
        dialog.show()
    }
}
```

Output :



Lampiran 2 : User Manual

1. Pengguna
   a. Buka aplikasi.
   b. Pilih Manga yang mau di baca.
   c. Pilih Chapter yang tersedia.

2. Admin
   a. Login admin dengan masukan user dan password admin.
   b. Pilih icon folder pojok kanan atas untuk input data.
   c. Pilih data yang akan di inputkan :
      1. Data Ketegori.
      2. Data Manga.
   d. Setelah data di inputkan logout pada icon di pojok kanan atas.

Lampiran 3 : Ketentuan Pendadaran

PEMBERITAHUAN SEBELUM UJIAN :Pengumpulan akhir dokumen Tugas Akhir/Skripsi melewati batas akhir ganjil 2022/2023, mahasiswa harus menyelesaikan registrasi dan KRS semester berikutnya.

KRITERIA KELULUSAN UJIAN SIDANG / PENDADARAN

1. Lulus ujian tanpa syarat, disebut kriteria 1.
2. Lulus bersyarat, disebut kriteria 2, yaitu dengan sedikit perbaikan atau penyempurnaan text dan atau program dalam wa 31 October 2023
dan tidak ada ujian lagi. Jika dalam waktu yang ditentukan mahasiswa tersebut tidak dapat menyelesaikan, maka, mahasiswa yang bersangkutan dianggap tidak lulus ujian.
3. Tidak lulus ujian sidang/pendadaran, disebut kriteria 3, dijelaskan, disarankan Ketua Tim Penguji untuk mempelajari

Ketentuan bagi peserta yang tidak lulus ujian sidang / pendadaran.
1)Mahasiswa wajib menempuh ujian sidang/pendadaran ulang
2) Kesempatan ujian sidang/pendadaran ulang hanya diberikan dalam rentang waktu maksimum 6 bulan, setelah ujian
3) Jika sampai batas waktu maksimum 6 bulan tersebut belum dapat diajukan/diselesaikan, maka calon peserta ujian
4) Mahasiswa yang akan menempuh ujian sidang/pendadaran ulang ini diwajibkan membayar biaya ujian setara 2 SKS

Yogyakarta, 8 Maret 2023
Memahami dan
Mematuhi peraturan di

Juliana Purwowidarmanto

Lampiran 4 : Keputusan Hasil Ujian Pendadaran

YAYASAN PENDIDIKAN WIDYA BAKTI YOGYAKARTA

UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA

Jl. Raya Janti (Majapahit) No.143, Yogyakarta, 55198, Telp (0274) 486664,
Website: www.utdi.ac.id , E-mail: info@utdi.ac.id

KEPUTUSAN HASIL UJIAN PENDADARAN

Sesuai dengan hasil sidang pendadaran pada tanggal maka

Nama Mahasiswa JULIANA PURWOWIDARMANTO
NIM / Program Studi 205410172 / Informatika
Jenjang S1

dinyatakan LULUS dengan kriteria 2

Ketua Penguji Femi Dwi Astuti, S.Kom., M.Cs.

Lampiran 5 : Catatan Hasil Pendadaran

YAYASAN PENDIDIKAN WIDYA BAKTI YOGYAKARTA
UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA
Jl. Raya Janti (Majapahit) No.143, Yogyakarta, 55198, Telp (0274) 486664,
Website: www.utdi.ac.id , E-mail: info@utdi.ac.id

Hari, tanggal : Wednesday, 08 March 2023
Waktu : 08.00
Nama : JULIANA PURWOWIDARMANTO
No. Mahasiswa / Jurusa : 205410172 / Informatika

No	Hal yang harus diperbaiki	Pemberi Catatan
1.	Tambahkan 1 komik dengan chapter yang komplit (sampe selesai), ganti tulisan action dengan **Pilih Kategori**	Femi Dwi
2.	font belum sesuai pedoman, kata kunci abstrak urut abjad, hindari bulleted dalam naskah ilmiah, pakai numberring	Femi Dwi, Endang
3.	edit status completed belum dikerjakan, cek penulisan yang ada dalam penulisan	Femi Dwi, Endang
4.	Tambahkan tampilan komik urut per kategori, per judul atau tambahkan **SEARCH**	Femi Dwi