# MVVM + Koin আর্কিটেকচার ব্যাখ্যা (AssignedDefect Project)

এই ডকুমেন্টে প্রতিটি ক্লাস/ফাইলের কাজ, কেন ব্যবহার করা হয়, এবং এর উপকারিতা বিস্তারিতভাবে দেওয়া হলো।

## ১) **MainApplication.kt**

class MainApplication : Application() {  
 override fun onCreate(){  
 super.onCreate()  
 startKoin{  
 androidLogger(Level.ERROR)  
 androidContext(this@MainApplication)  
 modules(listOf(jobModule))  
 }  
 }  
}

**কাজ:** - অ্যাপ চালু হলে Koin (Dependency Injection framework) initialize করে। - Koin কে জানায় কোন মডিউল (jobModule) ব্যবহার করতে হবে।

**কেন ব্যবহার:** - Dependency ম্যানুয়ালি তৈরি না করে Koin স্বয়ংক্রিয়ভাবে বানায়।

**উপকারিতা:** - কোড ক্লিন ও মডিউলার হয়। - Dependency ম্যানেজমেন্ট সহজ হয়।

## ২) **JobsViewModel.kt**

class JobsViewModel(private val jobRepository: JobRepository) : ViewModel() {  
 val assignedDefectList = jobRepository.assignedDefectList  
 fun fetchAssignedDefectList(mechanicPin: String) {  
 viewModelScope.launch {  
 jobRepository.fetchAssignedDefectList(mechanicPin)  
 }  
 }  
}

**কাজ:** - Repository থেকে ডেটা এনে LiveData তে রাখে। - UI (Activity/Fragment) এই LiveData observe করে।

**কেন ব্যবহার:** - ViewModel ডেটা সংরক্ষণ করে rotation change হলেও ডেটা হারায় না।

**উপকারিতা:** - UI এবং Business Logic আলাদা থাকে। - রিইউজেবল ও সহজে টেস্ট করা যায়।

## ৩) **AssignedDefect.kt (Activity)**

class AssignedDefect : AppCompatActivity() {  
 private val jobsViewModel: JobsViewModel by viewModel()  
 ...  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(binding.root)  
 jobsViewModel.fetchAssignedDefectList("4245")  
 jobsViewModel.assignedDefectList.observe(this) { ... }  
 }  
}

**কাজ:** - UI লেয়ার যেখানে RecyclerView দিয়ে ডেটা দেখানো হয়। - ViewModel observe করে ডেটা নিয়ে UI তে দেখায়।

**কেন ব্যবহার:** - ViewModel থেকে আসা ডেটা UI তে bind করতে।

**উপকারিতা:** - UI simple হয়। - Business Logic আলাদা থাকে।

## ৪) **JobRepository.kt (Interface)**

interface JobRepository {  
 val assignedDefectList: LiveData<AssignedDefectResponse?>  
 suspend fun fetchAssignedDefectList(mechanicPin: String)  
}

**কাজ:** - Repository কী কী কাজ করবে তার contract।

**কেন ব্যবহার:** - Loose coupling, ইমপ্লিমেন্টেশন পরিবর্তন করলে ViewModel তাতে প্রভাবিত হয় না।

**উপকারিতা:** - Maintainability ও Testability সহজ।

## ৫) **JobRepositoryImpl.kt (Implementation)**

class JobRepositoryImpl(val context: Context) : JobRepository {  
 private val \_assignedDefectList = MutableLiveData<AssignedDefectResponse?>()  
 override val assignedDefectList: LiveData<AssignedDefectResponse?> get() = \_assignedDefectList  
  
 override suspend fun fetchAssignedDefectList(mechanicPin: String) {  
 try {  
 val response = RetrofitClient.getInstance(Config.IFLEET\_BRAC\_BASE\_URL)  
 ?.getAssignedDefect(mechanicPin)?.await()  
 \_assignedDefectList.postValue(response)  
 } catch (e: Exception) {  
 \_assignedDefectList.postValue(null)  
 }  
 }  
}

**কাজ:** - API থেকে Retrofit দিয়ে ডেটা আনে। - MutableLiveData তে ডেটা সেট করে।

**কেন ব্যবহার:** - ডেটা সোর্স (API, DB, Cache) হ্যান্ডেল করার জন্য।

**উপকারিতা:** - ViewModel কে API নিয়ে চিন্তা করতে হয় না। - Data Layer আলাদা থাকে।

## ৬) **jobModule.kt (Koin Module)**

val jobModule = module {  
 single<JobRepository> { JobRepositoryImpl(androidContext()) }  
 viewModel { JobsViewModel(get()) }  
}

**কাজ:** - Koin এর জন্য Dependency configuration।

**কেন ব্যবহার:** - ম্যানুয়ালি object তৈরি না করে Koin তৈরি করে।

**উপকারিতা:** - Dependency Injection সহজ হয়। - কোড clean ও maintainable হয়।

## ৭) **AssignedDefectAdapter.kt (RecyclerView Adapter)**

class AssignedDefectAdapter(  
 private val listOfAssignedDefectItems: MutableList<AssignedDefectResponseItem>,  
 private val onAssignedDefectClickListener: OnAssignedDefectClickListener  
) : RecyclerView.Adapter<AssignedDefectAdapter.AssignedDefectViewHolder>() {  
 ...  
}

**কাজ:** - RecyclerView এ ডেটা দেখানোর জন্য Adapter। - DiffUtil ব্যবহার করে list efficiently update করে।

**কেন ব্যবহার:** - RecyclerView তে ডেটা bind করার জন্য।

**উপকারিতা:** - UI fast হয় (DiffUtil এর কারণে)। - Memory efficient। - Item click handle করা যায়।

# 🔄 সামগ্রিক আর্কিটেকচার (Data Flow)

1. **UI Layer (AssignedDefect Activity + Adapter)** → ViewModel থেকে ডেটা observe করে UI তে দেখায়।
2. **ViewModel (JobsViewModel)** → Repository থেকে ডেটা আনে।
3. **Repository (JobRepositoryImpl)** → API কল করে ডেটা আনে।
4. **Koin Module (jobModule)** → Dependency Injection এর মাধ্যমে সবকিছু যুক্ত করে।

👉 ফলে কোড ক্লিন, মডিউলার, টেস্টেবল ও maintainable হয়।