Dagger
- What is dependency management
- Design classes with dependencies
- Design based on inversion of control
- Design based on injection
- Designing own dependency management framework
- Singleton & Introduction to Dagger
- History of Dagger 2
- Create your own custom Annotation
- Understanding Dagger Framework
- Project introduction, @Module, @Inject, and @Provides
- Create and use a Component
- Singleton and Scope
- Use Dagger in an Activity
- Share instances between components
- Scope and Component Methods
- Constructor Injection
- Qualifier and Named

Learn Kotlin
- Intro to Kotlin and Type Hierarchy
- Setting it up with project
- Classes
- Variables
- Functions
- Null Safety
- Constructor
- Data class
- Object Declaration and Expression
- Control Flow Expression
- Collections
- Lambda Function
- Extension Function and Let Run Also Apply

Architectural Components
- Introduction to Lifecycle
- Challenges of lifecycle handling
- Activity rotation problem
- Lifecycle States and Events
  - Create a Lifecycle Aware Component
  - Create a TimerToast in an Activity
  - Make the TimerToast lifecycle aware
  - What is a ViewModel?
  - How ViewModel solves screen rotation problems
  - What is a LiveData?
  - Types of LiveData
  - Sharing a ViewModel
  - Using ViewModel and LiveData
  - Create ViewModel and LiveData based TimerToast
  - Use ViewModel and LiveData in an Activity

- RxJava
  - How does threading work in Android?
  - What is RxJava?
  - Components and basic examples
  - Schedulers
  - AsyncTask vs RxJava
  - Operators Examples - Map, Filter, Zip & FlatMap
  - Disposable & CompositeDisposable
  - Types of observables and Create your own observables
  - Solving Search problem with RxJava Operators - Debounce, DistinctUntilChanged, SwitchMap
  - Advantages of RxJava

- Database
  - Relational database concepts
  - Tables and Schema
  - Problems in a bad Schema design
  - Types of Relationships and Foreign Keys
  - Normalization and many-to-many relationships
  - Introduction to Room Database
  - CRUD operations in Room Database
  - Project setup and User Entity
  - Create User DAO and queries
  - Create Room Database instance
  - Using Room Database
  - Dagger setup for Room
  - Making Room queries using RxJava in ViewModel
  - Show Room data in UI using LiveData
  - Create relations in Room Database
  - Embedded
  - Relation and Foreignkey
  - DAO and queries across tables
- Test queries using UI
- Advanced Concepts
- TypeConvertors
- Migration

- Networking
  - Concepts: HTTP, OKHttp, and Retrofit
  - Introduction to Networking
  - What is Retrofit?
  - Network Caching
  - Interceptors
  - Read and Write Timeout
  - Parse data with Gson
  - Retrofit with RxJava
  - Implementing Network APIs through code
  - Project Setup
  - Create Networking Class
  - Configure Retrofit
  - Create Request and Response Model
  - Create POST request
  - Configure Dagger for Networking
  - Make Network call in a ViewModel
  - Create GET request and complex data Model
  - Add Query parameters and Headers
  - Delete query

- MVVM and Instagram project
  - Different types of Architectures
  - An Architecture use case?
  - Feature addition problem
  - Why tests are important
  - Some suggestions for adopting an Architecture
  - Separation of concern
  - No hard dependency principle
  - What is MVC architecture?
  - What is MVP architecture?
  - What is MVVM architecture?
  - MVVM architecture blueprint
  - MVVM package overview
  - Getting started with MVVM
  - Base classes overview
  - Introduction to Generics
  - ViewModel overview
  - Build the Base classes for MVVM
  - Project setup
- Create BaseViewModel
- Create BaseActivity
- Create BaseFragment
- Create ViewModelProviderFactory
- Use ViewModelProviderFactory
- Attach MainActivity UI with LiveData
- Setup Dagger for MainActivity
- Showing Toast
- ViewModelProviderFactory
- How ViewModelProviders works?
- How ViewModelProviders works?
- Lifecycle aware RecyclerView Design
- Problems of using RecyclerView in MVVM
- Principles of lifecycle aware RecyclerView
- Using RecyclerView Adapter callbacks
- Base classes needed
- Activity lifecycle effect on RecyclerView?
- Using Lifecycle aware Adapter in Activity
- Implementing Lifecycle aware RecyclerView
- Create BaseItemViewModel
- Create BaseItemViewHolder
- Dagger setup for ViewHolder
- Lifecycle state change for ViewHolder
- Create BaseAdapter
- ViewHolder's lifecycle change with window attach/detach
- Associating Activity/Fragment lifecycle with ViewHolder
- Create Post list UI
- Create Post Adapter
- Populating RecyclerView with Post list data
- Run the code developed
- Login Screen of MindOrks Instagram App
- Create Login Activity UI
- Create LoginActivity and LoginViewModel
- Create Login fields validations
- Use Login Validator in LoginViewModel
- Integrate Login APIs using Retrofit
- Add Login Repository
- Handle Login UI changes in ViewModel
- Add login UX logic
- Associate Login UI with LiveData
- Run the code developed
- Main screen of MindOrks Instagram App
- UI design overview
- Create MainActivity and empty Fragments
- Add bottom navigation
- Setup Dagger classes
- Add fragment toggle
- Home screen of MindOrks Instagram App
- UI design overview
- API doc overview
- Integrate API using Retrofit
- Add PostRepository
- Create Post List UI
- Create PostItemViewModel and PostAdapter
- Building up the HomeViewModel
- Implement Pagination feature using RxJava
- Add load more feature using RecyclerView
- Add LiveData for UI in PostItemViewModel
- Associating LiveData with PostItemViewHolder
- Post create screen of MindOrks Instagram App
- UI design overview
- Build the Photo fragment UI
- Capture image through Camera
- Pick image through Gallery
- Image handling inside PhotoViewModel
- Multipart image upload
- Post creation
- PhotoFragment, HomeFragment and MainActivity communication
- Update Post List with new Post

- **Unit Testing**
  - What is testing and its advantages
  - Types of Unit Test and packaging
  - Implementation
  - Writing Unit Test
  - Libraries used In Unit Test
  - Writing unit test for ViewModel
  - Writing UI Test

- **Kotlin Coroutines**
  - What exactly are Coroutines?
  - Need for the solution which Kotlin Coroutines provide
  - Dispatchers, suspend, launch, async
  - What are scopes in Kotlin Coroutines?
  - Exception handling in Kotlin Coroutines
  - ViewModelScope For Less Boilerplate Code
  - Complete Coroutines Implementation in NewsApp
  - Final Project - NewsApp Source Code
Interview Kit and Guide

- **Android Build System and Memory Management**
  - Android Build system
  - Introduction
  - JIT and JVM
  - Android Dex File
  - DVM, ART, and AOT
  - Android Memory Management
  - How objects and primitives are stored.
  - Heap memory storage
  - Large Heap
  - Multiple App Ram management
  - Stack and Thread
  - Stack and Heap in multithreaded condition
  - Memory Leaks

- **Android Multithreading and Handler-Looper**
  - Multithreading
  - Main Thread and Event Loop
  - Multithreaded System
  - Monitor and Synchronization
  - ReentrantReadWriteLock
  - Executor Service
  - Atomic Boolean and CountDownLatch
  - Deadlock
  - Android Handler and Looper
  - The need for Handler and Looper
  - Implement SimpleWorker using Thread
  - How Handler and Looper works?
  - What is a message queue?
  - What is HandlerThread?
  - Create a worker using Handler

- **Networking Caching Interceptor Image Loading**
  - What problems Image Loading Library Glide solves?
  - How Glide solves OOM?
  - How Glide solves slow loading issues?
  - How Glide solves UI unresponsiveness?

- **System Design(Mobile): WhatsApp and Location Sharing App**
  - HTTP vs WebSocket
- How does the notification system work?
- WhatsApp Design
- Location Sharing Design
- How Video calling works?

**Demo Video Link:**
https://www.youtube.com/playlist?list=PL6nth5sRD25gy4OnREK4YRETq2YmyBm9B

Created: December, 2020