# **Project Report**

## Project Title:

NOT FOR WILL!!! FOR DONE!!! — A To-Do List Android App using Kotlin, Jetpack Compose, and Room Database

## Objective:

To build a fully functional, user-friendly To-Do list application that allows users to:

* Add tasks with a due date
* Mark tasks as completed
* Delete tasks when done
* Store all data locally using Room Database

This project was developed as a first attempt to learn full-cycle Android app development with modern tools.

## Tools and Technologies Used:

* **Programming Language**: Kotlin
* **IDE**: Android Studio
* **UI Toolkit**: Jetpack Compose
* **Architecture**: MVVM (Model-View-ViewModel)
* **Database**: Room Persistence Library

## Step-by-Step Procedure:

### Step 1: Project Setup

* Created a new Android project in Android Studio.
* Chose **Empty Compose Activity**.
* Named the application: NOT FOR WILL!!! FOR DONE!!!
* Set up minimum SDK to 21 and compile SDK to 35.

### Step 2: Gradle Configuration

Updated the build.gradle.kts files:

* Added plugins:
* plugins {  
   id("com.android.application")  
   id("org.jetbrains.kotlin.android")  
   id("org.jetbrains.kotlin.kapt")  
  }
* Declared Room and Compose dependencies.
* Enabled compose = true in buildFeatures.
* Synced the Gradle files to ensure the dependencies were correctly resolved.

### Step 3: Data Layer

#### 1. Entity: Task.kt

@Entity(tableName = "tasks")  
data class Task(  
 @PrimaryKey(autoGenerate = true) val id: Int = 0,  
 val title: String,  
 val isDone: Boolean = false,  
 val date: String  
)

#### 2. DAO: TaskDao.kt

@Dao  
interface TaskDao {  
 @Query("SELECT \* FROM tasks ORDER BY id DESC")  
 fun getAllTasks(): Flow<List<Task>>  
  
 @Insert(onConflict = OnConflictStrategy.REPLACE)  
 suspend fun insertTask(task: Task)  
  
 @Delete  
 suspend fun deleteTask(task: Task)  
  
 @Update  
 suspend fun updateTask(task: Task)  
}

#### 3. Database: TaskDatabase.kt

@Database(entities = [Task::class], version = 1)  
abstract class TaskDatabase : RoomDatabase() {  
 abstract fun taskDao(): TaskDao  
  
 companion object {  
 @Volatile private var INSTANCE: TaskDatabase? = null  
  
 fun getDatabase(context: Context): TaskDatabase {  
 return INSTANCE ?: synchronized(this) {  
 val instance = Room.databaseBuilder(  
 context.applicationContext,  
 TaskDatabase::class.java,  
 "task\_database"  
 ).build()  
 INSTANCE = instance  
 instance  
 }  
 }  
 }  
}

### Step 4: ViewModel Layer

#### ViewModel: TaskViewModel.kt

class TaskViewModel(private val dao: TaskDao) : ViewModel() {  
 val tasks: StateFlow<List<Task>> = dao.getAllTasks().stateIn(  
 viewModelScope, SharingStarted.Lazily, emptyList()  
 )  
  
 fun addTask(title: String, date: String) {  
 viewModelScope.launch {  
 dao.insertTask(Task(title = title, date = date))  
 }  
 }  
  
 fun updateTask(task: Task) {  
 viewModelScope.launch { dao.updateTask(task) }  
 }  
  
 fun deleteTask(task: Task) {  
 viewModelScope.launch { dao.deleteTask(task) }  
 }  
}

#### Factory: TaskViewModelFactory.kt

class TaskViewModelFactory(private val dao: TaskDao) : ViewModelProvider.Factory {  
 override fun <T : ViewModel> create(modelClass: Class<T>): T {  
 if (modelClass.isAssignableFrom(TaskViewModel::class.java)) {  
 return TaskViewModel(dao) as T  
 }  
 throw IllegalArgumentException("Unknown ViewModel class")  
 }  
}

### Step 5: UI Layer (Jetpack Compose)

#### Main UI: MainScreen.kt

@Composable  
fun MainScreen(viewModel: TaskViewModel) {  
 val taskList by viewModel.tasks.collectAsState()  
 // Compose UI to show task list, input field, and buttons  
}

#### Theming: Theme.kt

* Created a custom theme file NOTFORWILLFORDONETheme with light and dark color schemes.
* Used Material3 Design components.

### Step 6: MainActivity Setup

class MainActivity : ComponentActivity() {  
 private val viewModel: TaskViewModel by viewModels {  
 TaskViewModelFactory(TaskDatabase.getDatabase(this).taskDao())  
 }  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContent {  
 NOTFORWILLFORDONETheme {  
 MainScreen(viewModel = viewModel)  
 }  
 }  
 }  
}

**Respective Files and their Functions: -**

| **File** | **Folder** | **Purpose** |
| --- | --- | --- |
| Task.kt | data | Room entity |
| TaskDao.kt | data | DAO with insert, delete, update, getAll |
| TaskDatabase.kt | data | Room DB singleton |
| TaskViewModel.kt | viewmodel | Holds StateFlow and business logic |
| TaskViewModelFactory.kt | viewmodel | Factory to create ViewModel |
| MainScreen.kt | ui | Composable to show list, buttons |
| MainActivity.kt | root | Launches UI and ViewModel |
| Theme.kt | ui.theme | Custom Compose theme |

## 

## 

## Final Output:

* A working Android app that can:
  + Add tasks with a date
  + View all tasks in reverse order
  + Mark tasks as done (change visual appearance)
  + Delete tasks

## Learnings and Challenges:

* Gained strong understanding of Jetpack Compose UI framework.
* Learned how Room simplifies local data persistence.
* Applied MVVM pattern to separate concerns.
* Faced issues with Gradle sync and missing dependencies — resolved them step-by-step.

## Future Improvements:

* Add notification reminders
* Include filtering by date
* Sync with cloud (Firebase)
* Add animations and swipe gestures

## Conclusion:

This project marks the beginning of my journey into Android app development. It helped me understand core concepts like Compose, Room, and MVVM architecture. I’m excited to build more feature-rich apps in the future!

**Developer:** Subhadra Bhattacharyya

**Project Name:** NOT FOR WILL!!! FOR DONE!!!