1. **Final Milestone Case Study: Grocery Shopping List App (Real-Time Scenario)**

**Problem Statement:**

Develop a mobile application to manage a user's grocery shopping lists with the following requirements:

* Persistent offline data storage using **Room Database**.
* Support for CRUD operations (Create, Read, Update, Delete).
* Clean UI with intuitive navigation between screens.

**App Features and Screens**

**1. Home Screen (List of Grocery Items)**

* Displays a scrollable list of grocery items stored in the Room DB.
* Each item shows:
  + Item name
  + Quantity
  + Purchase status (checkbox for bought/unbought)
* "Add Item" floating action button (FAB) for navigation to the Add Item screen.
* Swipe to delete an item from the list.

**Test Objectives:**

* Fetch all items from Room DB and display them in a RecyclerView.
* Verify swipe-to-delete functionality.
* Test navigation to the Add Item screen.

**2. Add Item Screen**

* A form to input item details:
  + Item name (Text input)
  + Quantity (Number input)
  + Save button to persist data in Room DB.

**Test Objectives:**

* Validate form inputs (e.g., empty item names should show an error).
* Verify that saving a new item updates the Room DB and reflects on the Home Screen.
* Test navigation back to the Home Screen after saving.

**3. Edit Item Screen**

* Similar UI to the Add Item Screen but pre-filled with the selected item's details.
* Allows editing the item name or quantity.
* "Update" button to save changes in Room DB.
* "Cancel" button to discard changes.

**Test Objectives:**

* Verify that selected item details are pre-populated correctly.
* Test data update logic in the Room DB.
* Check for proper navigation back to the Home Screen after updating or canceling.

**4. Filter Screen**

* Allows users to filter the list based on purchase status:
  + "Show All"
  + "Show Bought Items Only"
  + "Show Unbought Items Only"
* Displays the filtered list in a RecyclerView.

**Test Objectives:**

* Ensure correct filtering logic for purchased and unpurchased items.
* Test that Room DB queries efficiently fetch the correct data subset.
* Verify navigation back to the Home Screen from the Filter Screen.

**Technical Components**

1. **Room Database:**
   * Entities for GroceryItem with fields: id, name, quantity, and status.
   * DAO for data operations (Insert, Update, Delete, Query).
2. **Navigation Components:**
   * Handle screen transitions.
3. **ViewModel:**
   * Manage UI data and business logic.
4. **Repository Pattern:**
   * Abstraction layer between ViewModel and Room DB.
5. **Final Milestone Real-Time Case Study: Task Management App (Mobile Application with Room DB) – 4 Screens**

**Scenario Title:**

**TaskEase – A Task Management App with Offline Storage (Using Room DB)**

**Problem Statement:**

A mobile application is required to help users manage their daily tasks efficiently, even when offline. The app should allow users to create, update, and delete tasks, categorize them, and mark them as completed. Room Database should be used for storing tasks locally, ensuring data persistence.

**Requirements:**

1. Users should be able to create tasks with a title, description, priority, and due date.
2. Tasks should be categorized (e.g., Work, Personal, Study).
3. Users should be able to update, delete, and mark tasks as completed.
4. The app should store all tasks in **Room Database** to work offline.
5. A status filter should allow users to view **Pending, Completed, or All** tasks.
6. The UI should be responsive and use **android UI Components**

**Application Flow (4 Screens)**

**Screen 1: Task List Screen (Home Screen)**

* Displays all tasks stored in **Room Database**.
* Tasks are grouped by category (Work, Personal, Study, etc.).
* Each task shows a **title, priority, and status (Completed/Pending)**.
* A floating action button (FAB) allows users to add a new task.
* A filter option lets users view **All, Completed, or Pending** tasks.

**Test Objectives:**  
✅ Ensure tasks load correctly from Room DB.  
✅ Verify the filtering function (All, Completed, Pending).  
✅ Test smooth scrolling for large task lists.  
✅ Check if tapping a task opens the **Task Detail Screen**.  
✅ Validate navigation to the **Add Task Screen** using FAB.

**Screen 2: Add Task Screen**

* Users can create a new task with the following details:
  + Task Title
  + Description
  + Due Date (Date Picker)
  + Priority (High, Medium, Low)
  + Category (Dropdown with predefined options: Work, Personal, Study)
* A "Save Task" button stores the task in **Room Database**.
* A "Cancel" button discards the task.

**Test Objectives:**  
✅ Verify task details are stored in Room DB.  
✅ Ensure users cannot save an empty task (validation check).  
✅ Confirm that selecting a category and priority works correctly.  
✅ Test the Date Picker functionality.  
✅ Check if pressing "Cancel" navigates back without saving.

**Screen 3: Task Detail Screen**

* Displays full details of a selected task.
* Users can:
  + Edit task details.
  + Mark the task as Completed/Pending.
  + Delete the task.

**Test Objectives:**  
✅ Ensure task details are loaded from Room DB correctly.  
✅ Test the Edit feature (updates should reflect in Room DB).  
✅ Verify the "Mark as Completed" toggle works properly.  
✅ Check that deleting a task removes it from the database.  
✅ Validate the back navigation to the **Task List Screen**.

**Screen 4: Settings & Backup Screen**

* Allows users to:
  + Enable/Disable **Task Notifications** for reminders.
  + Backup/Restore tasks from **local storage**.

**Test Objectives:**  
✅ Ensure toggling notifications updates shared preferences.  
✅ Test backup feature (Room DB data should be saved as a file).  
✅ Validate that restoring tasks reloads them into the database.

**Expected Outcomes:**

1. Users can create, update, delete, and filter tasks offline.
2. Room DB ensures data is persistent, even after the app is closed.
3. The app is responsive and handles large task lists smoothly.
4. Task status updates (Completed/Pending) work as expected.
5. **Final Milestone Real-Time Scenario Case Study (Mobile App with Room DB)**

**Application: Smart Medicine Reminder App**

*A mobile application that helps users track and manage their medication schedules efficiently.*

**Problem Statement:**

Many individuals, especially elderly patients or those with chronic illnesses, forget to take their medications on time. A mobile application is needed that allows users to store their prescriptions, receive reminders, and track their medication history. This app should use **Room Database** for local data storage.

**Application Requirements:**

1. **Users can add medications** with details such as name, dosage, frequency, and time.
2. **Scheduled reminders** should notify users when it's time to take a medication.
3. **History tracking** should allow users to mark medications as taken or skipped.
4. **Offline access** is required to ensure users can view and manage their medication data without an internet connection.

**Screens & Features**

**Screen 1: Home Screen (Dashboard)**

* Displays a list of medications scheduled for the current day.
* Shows medication status: **Pending**, **Taken**, or **Skipped**.
* "Add Medication" button to navigate to the Add Medication screen.

**Test Objectives:**

1. Ensure that the list of scheduled medications loads correctly from the Room Database.
2. Validate that the medication status updates correctly upon user interaction.
3. Check that clicking “Add Medication” navigates to the next screen.

**Screen 2: Add Medication Screen**

* Form to enter:
  + **Medicine Name** (Text Input)
  + **Dosage** (Dropdown Selection)
  + **Frequency** (Once, Twice, Thrice a day, Custom)
  + **Time(s) to Take** (Time Picker)
  + **Start & End Date** (Date Picker)
* Save button stores data in the Room Database.

**Test Objectives:**

1. Verify that all input fields function correctly and store valid data.
2. Ensure data is saved in the Room Database upon clicking "Save".
3. Validate that navigating back to the Home Screen shows the newly added medication.

**Screen 3: Reminder & Notification Screen**

* Displays upcoming medication reminders.
* Sends notifications when it's time to take medicine.
* Allows users to mark medication as **Taken** or **Skipped**.

**Test Objectives:**

1. Confirm that reminders trigger at the correct time.
2. Validate the push notification functionality.
3. Ensure marking a reminder as **Taken** or **Skipped** updates the Room Database correctly.

**Screen 4: History & Reports Screen**

* Shows a list of previously taken or missed medications.
* Filters to view history by **Day, Week, or Month**.
* Summary of missed doses.

**Test Objectives:**

1. Validate that historical data is fetched correctly from Room Database.
2. Ensure the filter functionality works properly.
3. Check if the summary of missed doses is accurate.

**Expected Outcomes**

1. **Seamless CRUD operations** using Room Database.
2. **Offline data access**, allowing users to manage their schedules without the internet.
3. **Accurate reminders and notifications** for medication intake.
4. **User-friendly history tracking** for past medication adherence.