## Proposed Android App Architecture (Java-based)

This architecture uses the recommended **MVVM (Model-View-ViewModel)** pattern, which separates the UI from the business logic. It also incorporates a dedicated data layer and handles the unique requirements of an App Widget.

### Key Components

* **UI/View Layer:** This layer is responsible for rendering UI components and handling user input.
  + MainActivity & Fragments: Displays the main timetable view, as well as forms for adding, editing, and deleting entries.
  + TimetableWidgetProvider: This is the broadcast receiver that handles widget lifecycle events (e.g., onUpdate, onEnabled). It acts as the bridge between the system and your app's data.
  + TimetableWidgetFactory: A RemoteViewsService.RemoteViewsFactory that provides the data for the widget's ListView or GridView. It communicates with the data layer to get the timetable entries.
* **ViewModel Layer:** The ViewModel holds and manages UI-related data in a lifecycle-conscious way. It acts as a bridge between the View and the data source.
  + TimetableViewModel: Fetches timetable data from the Repository and exposes it as LiveData or StateFlow to the MainActivity and Fragments. It also contains logic for adding, deleting, and updating entries.
* **Domain Layer (Optional but Recommended):** This layer contains the business logic and use cases.
  + TimetableRepository: Abstracts the data sources. It is the single source of truth for timetable data. It can decide whether to fetch data from the local database or a remote API. It provides clean APIs to the ViewModel.
* **Data Layer:** This layer provides and manages the data for the application.
  + TimetableDatabase (Room): A local database that stores all the timetable entries (e.g., subject name, professor, time, room). It defines the TimetableEntry entity and the TimetableDao (Data Access Object) to perform database operations.
  + TimetableApiService: A service that could be used to fetch timetable data from a remote server (e.g., for syncing across devices). For a simple app, this might be optional.

### Data Flow

1. The MainActivity observes the TimetableViewModel.
2. The TimetableViewModel requests data from the TimetableRepository.
3. The TimetableRepository gets the data from the TimetableDatabase.
4. The TimetableViewModel updates the LiveData, and the MainActivity updates its UI.
5. The TimetableWidgetProvider receives an update broadcast (ACTION\_APPWIDGET\_UPDATE).
6. It starts the TimetableWidgetService, which uses the TimetableWidgetFactory.
7. The TimetableWidgetFactory gets the latest timetable data from the TimetableRepository and populates the widget's ListView.

<center>

</center>

## User Stories

These user stories describe the app's core requirements from a user's perspective.

### Core Functionality

* **As a student**, I want to **add a new class to my timetable**, so that I can keep track of my schedule.
* **As a student**, I want to **edit an existing class entry**, so that I can correct mistakes or update information.
* **As a student**, I want to **delete a class from my timetable**, so that I can remove cancelled or completed classes.
* **As a student**, I want to **view my full weekly timetable** in the main app, so that I have a comprehensive overview of my schedule.
* **As a student**, I want to be able to **switch between a standard weekly view and a day order view**, so that I can accommodate my college's scheduling system.

### Widget Functionality

* **As a student**, I want to **add a widget to my home screen**, so that I can see my upcoming classes at a glance without opening the app.
* **As a student**, I want the **widget to automatically display my classes for the current day**, so that the information is always relevant.
* **As a student**, I want the **widget to update automatically** when I make changes in the app, so that the information is always in sync.
* **As a student**, I want to **tap on the widget** to open the main app, so that I can quickly access my full timetable.

### Data Management

* **As a student**, I want to **enter details for each class**, including the subject name, time, location, and professor, so that my timetable is complete.
* **As a student**, I want the app to **persist my timetable data** even if I close the app, so that I don't lose my schedule.
* **As a student**, I want to be able to **assign a day of the week or a day order number** to each class, so that the app displays the correct schedule.
* **As a student**, I want to **set the current day order** for the week, so that the app and widget display the correct schedule for today.

### Extra Features (Optional)

* **As a student**, I want to **set a reminder** for a class, so that I don't forget about it.
* **As a student**, I want to **view my timetable for any given week**, not just the current one, so that I can plan ahead.