**Assessment 2: Newsletter Selection (Daily Run)**

**Objective**

Implement a method that returns the **set of newsletters to send to a given user for “today”**, respecting user preferences, scheduling rules, and holiday exclusions. Write unit tests covering all rules.

**Data Model (minimum)**

**User**

* Id (uniqueidentifier / int)
* Email (varchar)
* IsOptedIn (bit) — receives newsletters at all
* PriorityOnly (bit) — if true, only receive priority newsletters
* WeekendOptIn (bit) — allowed to receive on Sat/Sun

**Newsletter**

* Id (uniqueidentifier / int)
* Title (nvarchar)
* IsPriority (bit)
* IsWeekendEligible (bit) — whether this newsletter is allowed on weekends
* (optional) IsActive (bit)

**UserNewsletterSchedule**

* UserId (FK → User)
* NewsletterId (FK → Newsletter)
* SendDate (date) — planned send date for that user/newsletter combo

**Holiday**

* Date (date) — national holiday dates (seed a few rows)

You may add indexes/constraints as needed. Keep schema simple.

**Business Rules (apply in this order)**

1. **Opt-in:** If IsOptedIn == false → return **empty set**.
2. **Date match:** Consider only UserNewsletterSchedule rows where SendDate == today.
3. **Holiday exclusion:** If **today is a Holiday**, return **empty set** (no sends today).
4. **Weekend rule:**
   * If today is **Sat/Sun** and WeekendOptIn == false → return **empty set**.
   * If weekend and allowed, still ensure each selected newsletter is weekend-eligible (IsWeekendEligible == true).
5. **Priority filter:**
   * If PriorityOnly == true, include **only** newsletters where IsPriority == true.
6. **Active/newsletter validity:** (optional) Exclude inactive newsletters if you include IsActive.

Result: a distinct list of Newsletter (or IDs) for that user, for today.

**Required Method**

Implement **one** entry point in your BL:

IReadOnlyList<Newsletter> GetNewslettersForUserToday(User user, DateOnly today);

* The method may call a DAL/repository to read schedules, newsletter metadata, and holidays.
* Keep it **deterministic** (pass today in; don’t read system clock inside the core logic).

**Unit Tests (xUnit/FluentAssertions or equivalent)**

Cover at least:

1. **Opt-out** → returns empty.
2. **Holiday** (today is holiday) → returns empty even if schedules exist.
3. **Weekday, opted-in, normal** → returns scheduled newsletters.
4. **Weekend without consent** → empty.
5. **Weekend with consent** → returns only weekend-eligible items.
6. **PriorityOnly true** → filters non-priority out.
7. **Multiple schedules** → returns distinct list for the user (no duplicates).
8. **No schedule for today** → empty.
9. **Edge:** PriorityOnly + WeekendOptIn + Holiday → still empty (holiday wins).
10. **Optional:** Inactive newsletter excluded if you model IsActive.

Aim for clear Arrange-Act-Assert and descriptive test names.

**Seed/Test Data (suggested)**

* Users:
  + U1: OptedIn=true, PriorityOnly=false, WeekendOptIn=false
  + U2: OptedIn=true, PriorityOnly=true, WeekendOptIn=true
  + U3: OptedIn=false (control)
* Newsletters:
  + N1: Priority=false, IsWeekendEligible=false
  + N2: Priority=true, IsWeekendEligible=true
* Schedules: rows mapping U1/U2 to N1/N2 for specific dates (weekday and weekend cases).
* Holidays: include one date that your tests target.