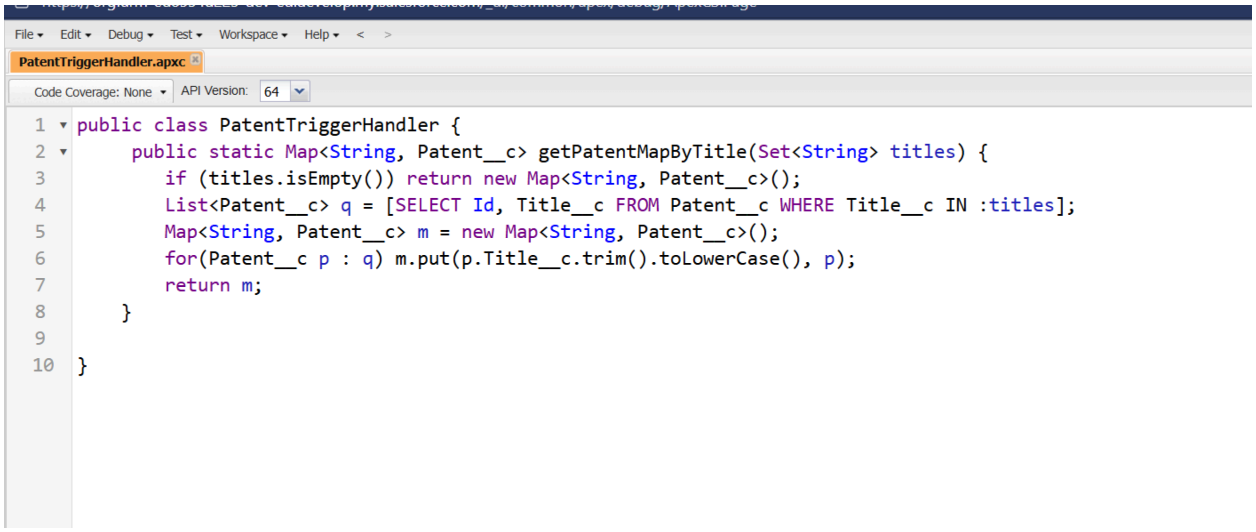
Phase 5: Apex Programming (Developer) — Step-by-step (IP & Patent Management)

---

1) Classes & Objects — Required

What to do (steps):

1. Create a handler class `PatentTriggerHandler.cls` (we'll call it from the trigger).
2. Keep classes small and single-purpose. Use `with sharing` for classes that run in user context, `without sharing` only when necessary.



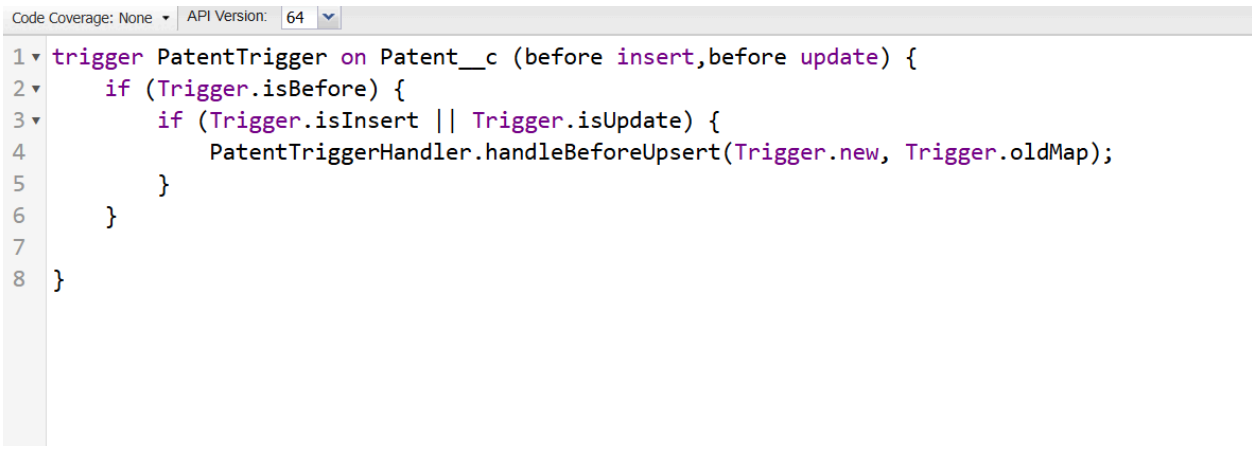
```
public class PatentTriggerHandler {
     public static Map<String, Patent__c> getPatentMapByTitle(Set<String> titles) {
        if (titles.isEmpty()) return new Map<String, Patent__c>();
        List<Patent__c> q = [SELECT Id, Title__c FROM Patent__c WHERE Title__c IN :titles];
        Map<String, Patent__c> m = new Map<String, Patent__c>();
        for(Patent__c p : q) m.put(p.Title__c.trim().toLowerCase(), p);
        return m;
    }

}
```

---

2) Apex Triggers (before/after insert/update/delete) — Required (basic)

What to do (steps):

1. Create **one trigger per object** (Patent__c). Keep it minimal — delegate to handler class.



```
trigger PatentTrigger on Patent__c (before insert,before update) {
    if (Trigger.isBefore) {
        if (Trigger.isInsert || Trigger.isUpdate) {
            PatentTriggerHandler.handleBeforeUpsert(Trigger.new, Trigger.oldMap);
        }
    }

}
```

3) Trigger Design Pattern — Required

Steps / rules to follow:

1. **One trigger per object.** Trigger only delegates.
2. Implement a handler class with static methods (e.g., `handleBeforeUpsert`).
3. Use a static boolean guard to prevent recursion if the handler performs DML that re-triggers.
4. Bulkify logic — always design for many records.

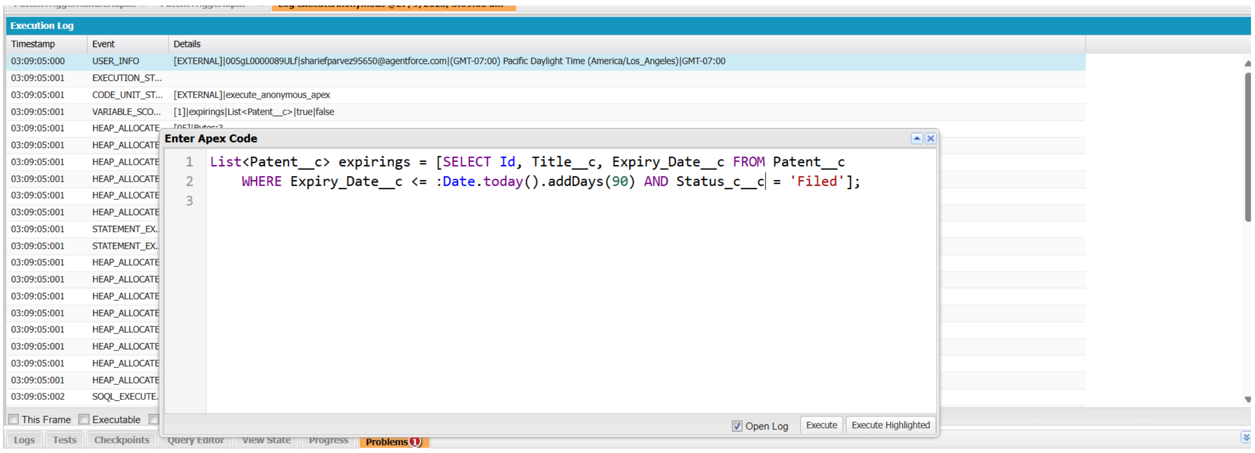
4) SOQL & SOSL — Required (SOQL essential, SOSL optional)

Steps:

1. Use **SOQL** for precise record queries (Patent lists, renewals). Always use selective WHERE clauses and LIMIT if needed.
2. Use **SOSL** only if you need text search across multiple fields (not necessary for MVP).
3. Avoid SOQL inside loops — use a single query with IN filters and Maps.

SOQL example:

```
List<Patent__c> expirings = [SELECT Id, Title__c, Expiry_Date__c FROM
Patent__c
    WHERE Expiry_Date__c <= :Date.today().addDays(90) AND Status__c =
'Filed'];
```



Execution Log

```
Timestamp     Event            Details
03:09:05:000  USER_INFO        [EXTERNAL]|005gL0000089ULf|shariefparvez95650@agentforce.com|(GMT-07:00) Pacific Daylight Time (America/Los_Angeles)|GMT-07:00
03:09:05:001  EXECUTION_ST...
03:09:05:001  CODE_UNIT_ST...  [EXTERNAL]|execute_anonymous_apex
03:09:05:001  VARIABLE_SCO...  [1]|expirings|List<Patent__c>|true|false
03:09:05:001  HEAP_ALLOCATE
03:09:05:001  HEAP_ALLOCATE
03:09:05:001  HEAP_ALLOCATE
03:09:05:001  HEAP_ALLOCATE
03:09:05:001  HEAP_ALLOCATE
03:09:05:001  HEAP_ALLOCATE
03:09:05:001  STATEMENT_EX...
03:09:05:001  STATEMENT_EX...
03:09:05:001  HEAP_ALLOCATE
03:09:05:001  HEAP_ALLOCATE
03:09:05:001  HEAP_ALLOCATE
03:09:05:001  HEAP_ALLOCATE
03:09:05:001  HEAP_ALLOCATE
03:09:05:001  HEAP_ALLOCATE
03:09:05:001  HEAP_ALLOCATE
03:09:05:001  HEAP_ALLOCATE
03:09:05:002  SOQL_EXECUTE...
```

Enter Apex Code

```
List<Patent__c> expirings = [SELECT Id, Title__c, Expiry_Date__c FROM Patent__c
    WHERE Expiry_Date__c <= :Date.today().addDays(90) AND Status_c__c = 'Filed'];
```

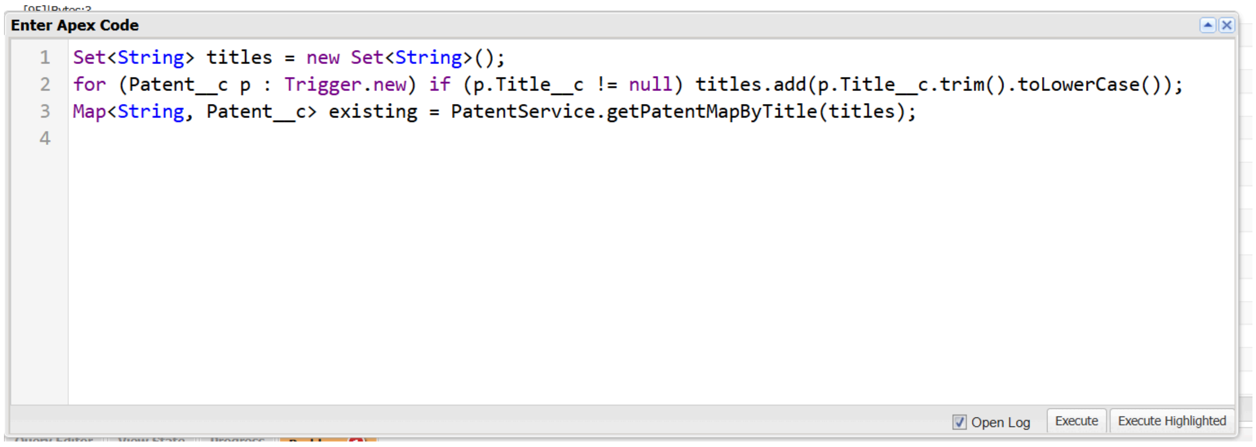
5) Collections: List, Set, Map — Required

Why & steps:

1. Use `Set<String>` to collect unique titles for queries.
2. Query results into `List<Patent__c>` then build `Map<Id, Patent__c>` or `Map<String, Patent__c>` for O(1) lookups.
3. Always iterate collections with `for` loops.

Quick pattern:

```
Set<String> titles = new Set<String>();
for (Patent__c p : Trigger.new) if (p.Title__c != null)
titles.add(p.Title__c.trim().toLowerCase());
Map<String, Patent__c> existing =
PatentService.getPatentMapByTitle(titles);
```



```
Set<String> titles = new Set<String>();
for (Patent__c p : Trigger.new) if (p.Title__c != null) titles.add(p.Title__c.trim().toLowerCase());
Map<String, Patent__c> existing = PatentService.getPatentMapByTitle(titles);
```

---

6) Control Statements — Required (basic)

Steps:

- Use `if/else`, `for` loops, `switch` (if needed) — keep logic readable.
- Guard against nulls (`String.isBlank`, `p.Field__c != null`) and boundary cases.

---

7) Batch Apex — Not necessary for the project

Why not:
We won't process large datasets in this 1-week MVP. Batch Apex is used to handle tens of thousands of records; for our demo data and initial org it's unnecessary and adds complexity.

---

8) Scheduled Apex — Not necessary for the project

Why not:
Renewal reminders are implemented using **Scheduled Flows** (admin tool). No need to write Scheduled Apex for this MVP

---

9) Future Methods — Not necessary (prefer Queueable)

Why not:
`@future` is older and limited. Use Queueable for async needs. For our MVP we'll avoid `@future`.

---

10) Exception Handling — Required

Steps:

1. Wrap external operations or risky DML in `try { } catch(Exception e) { }`.
2. Log errors — in tests just `System.debug(e)`; in production create a `Log__c` record for critical failures (optional).
3. Avoid swallowing exceptions silently — rethrow when necessary.

Example:

```
try {
    // DML or callout
} catch(Exception e) {
    System.debug('Error in PatentService: ' + e.getMessage());
    // optionally create a Log__c record or set an error field
    throw e; // if you want upstream to know
}
```

---

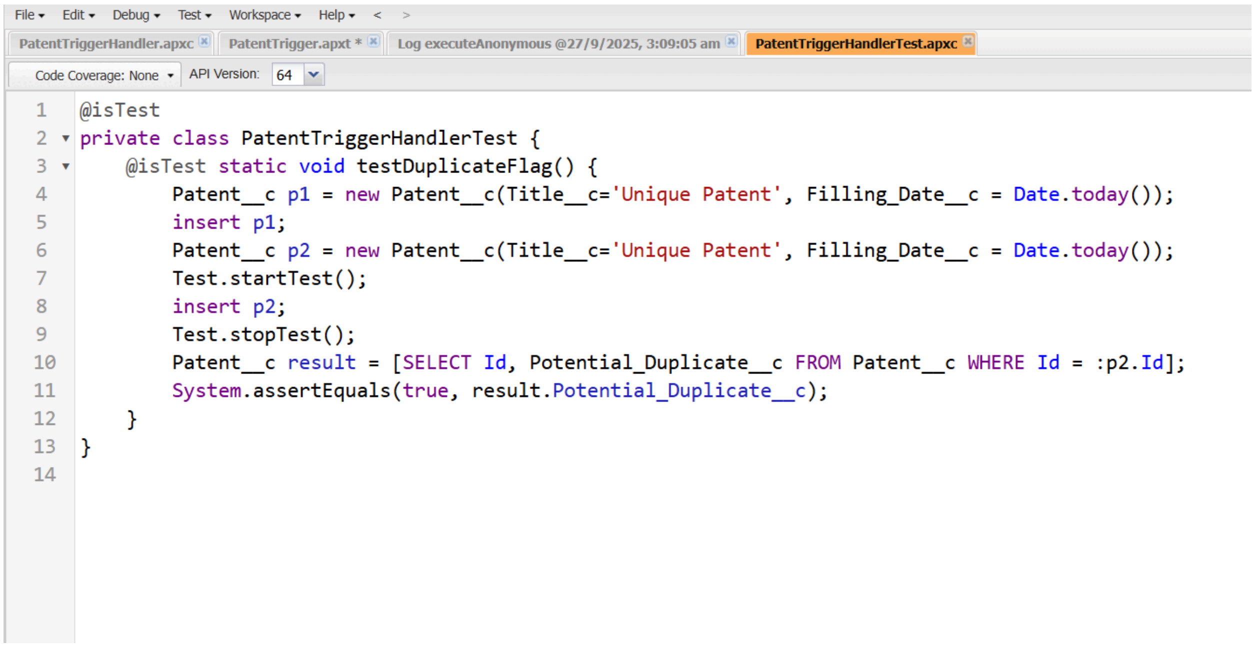
11) Test Classes — Required (must do)

Steps (must follow):

1. Create test classes for every Apex class and trigger. Name them `PatentTriggerHandlerTest`, `PatentServiceTest`.
2. Use `@isTest` and `Test.startTest()` / `Test.stopTest()`.
3. Insert test records in test context — do **not** rely on org data.

4. For callouts use `HttpCalloutMock` and set mock in tests.

5. Aim for meaningful coverage (target: pass tests; 75% org coverage not required in Dev Org but necessary for packaging).


File Edit Debug Test Workspace Help < >
PatentTriggerHandler.apxc
PatentTrigger.apxt *
Log executeAnonymous @27/9/2025, 3:09:05 am
PatentTriggerHandlerTest.apxc
Code Coverage: None
API Version: 64

```
@isTest
private class PatentTriggerHandlerTest {
    @isTest static void testDuplicateFlag() {
        Patent__c p1 = new Patent__c(Title__c='Unique Patent', Filling_Date__c = Date.today());
        insert p1;
        Patent__c p2 = new Patent__c(Title__c='Unique Patent', Filling_Date__c = Date.today());
        Test.startTest();
        insert p2;
        Test.stopTest();
        Patent__c result = [SELECT Id, Potential_Duplicate__c FROM Patent__c WHERE Id = :p2.Id];
        System.assertEquals(true, result.Potential_Duplicate__c);
    }
}
```

---

12) Asynchronous Processing — Minimal use recommended

Guidance:

- For this project: prefer **Flows** for scheduled work, and use **Queueable** only if you plan to do callouts or heavier processing.

- **Batch Apex** and **@future** are not needed.