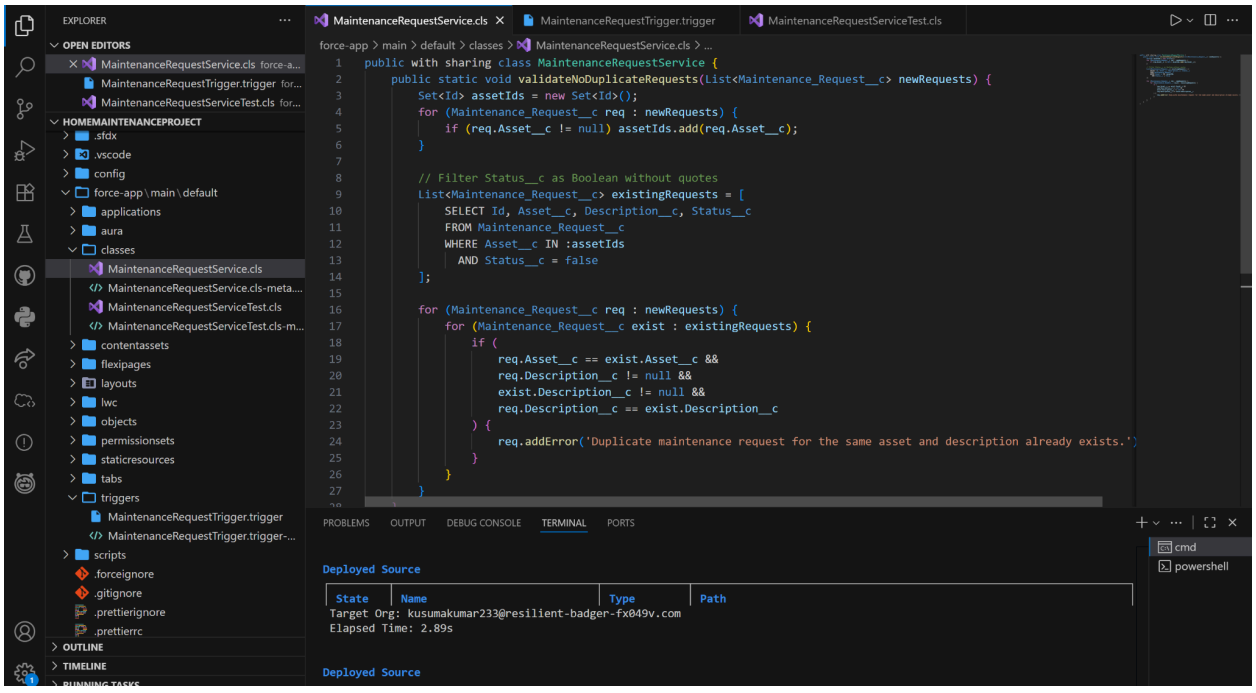
Phase 5:

Apex Programming for Home Maintenance and Repair Management

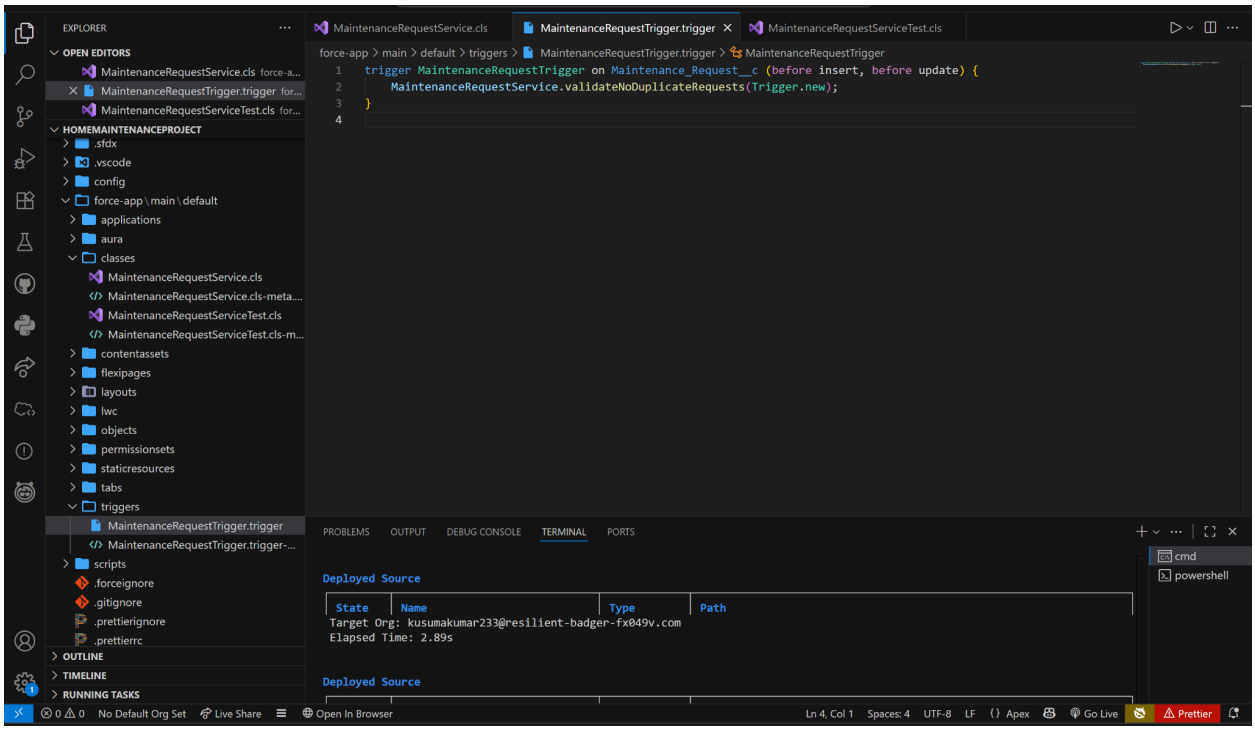
Step 1: Apex Classes & Objects

- Create a service class, e.g., MaintenanceRequestService, to hold all business logic related to maintenance requests.
- Main responsibilities:
    - Validate new maintenance requests (e.g., prevent duplicate requests for same asset and issue within a time window).
    - Enforce business rules centrally for reuse by triggers and other automations.
    - Keep triggers lightweight and focused only on orchestration.
- Design clean, modular, and reusable code patterns.



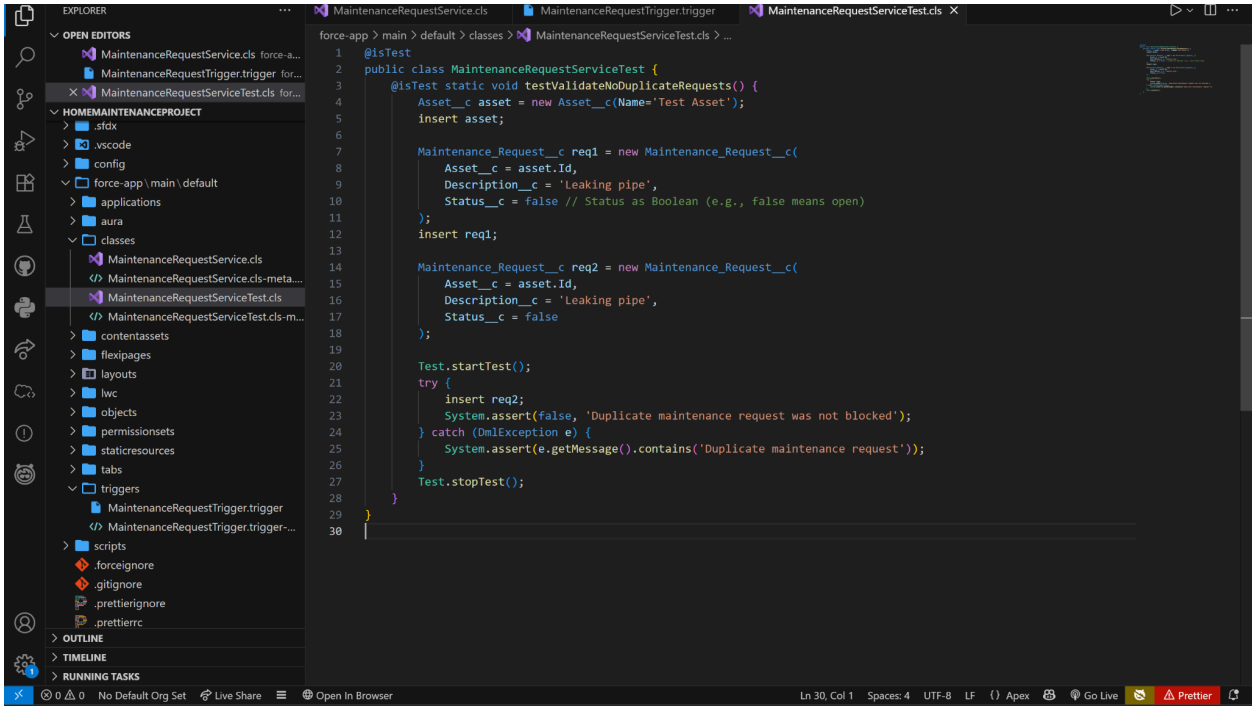
Step 2: Apex Triggers

- Implement a trigger MaintenanceRequestTrigger on Maintenance_Request__c object.
- Trigger runs before insert and before update to block invalid or duplicate maintenance requests.
- Use the service class for all validations and processing logic.
- Follow the Trigger Handler Design Pattern to separate context management from logic.



Step 3: Test Classes

- Write a dedicated test class for the service class and trigger.
- Positive test case: successful submission of a unique maintenance request.
- Negative test case: prevent duplicate maintenance request for the same asset and issue.
- Add assertions to verify expected behavior.
- Ensure 75%+ code coverage for deployment.
- Deactivate conflicting automations/flaws during testing if needed for consistency.



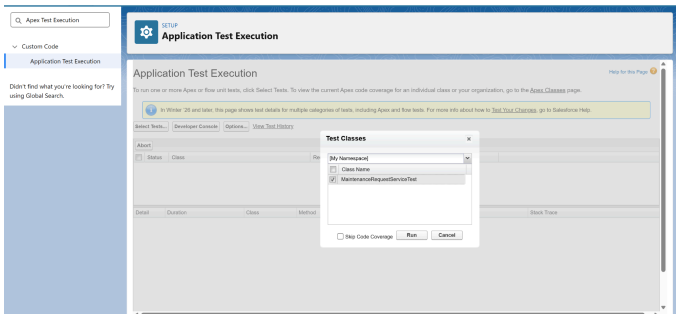

```
C:\Users\KUSUMA KUMAR\HomeMaintenanceProject>sf project deploy start --source-dir force-app/main/default --target-org HomeMaintenancePlaygrou
nd
 »   Warning: @salesforce/cli update available from 2.107.6 to 2.108.6.

──────────────── Deploying Metadata ────────────────

Deploying v64.0 metadata to kusumakumar233@resilient-badger-fx049v.com using the v65.0 SOAP API.

✓ Preparing 333ms
✓ Waiting for the org to respond 1.33s
✓ Deploying Metadata 2.05s
  ▸ Components: 3/3 (100%)
○ Running Tests - Skipped
○ Updating Source Tracking - Skipped
✓ Done 0ms

Status: Succeeded
Deploy ID: 0Afd200000Gzb4fCAB
Target Org: kusumakumar233@resilient-badger-fx049v.com
Elapsed Time: 3.73s
```



Step 4: SOQL & SOSL Queries

- Use SOQL queries to retrieve maintenance requests by asset, status, or date.
- Use SOSL sparingly for global search requirements (e.g., searching assets or request descriptions).
- Prefer SOQL when precise filters and relationships are needed.
- Optimize queries for bulk processing in triggers and asynchronous Apex.

Step 5: Use of Collections (List, Set, Map)

- Use Lists to hold maintenance requests for processing.
- Use Sets to ensure uniqueness of Asset–Issue pairs to prevent duplicates.
- Use Maps for efficient lookups, e.g., map asset IDs to their records or related maintenance requests.
- Collections optimize performance and clarity in processing bulk records.

Step 6: Control Statements

- Use if-else conditions to implement business rules (e.g., validate priority levels, status).
- Use loops to iterate over collections in bulk trigger contexts.
- Apply decision logic consistently to all records and scenarios.

Step 7: Asynchronous Apex

- Use Future methods for lightweight tasks like sending notifications (Email/SMS) after request creation, without slowing user experience.
- Use Queueable Apex for bulk operations such as updating statuses of requests or applying fees.
- Use Batch Apex to process large volumes of requests, for example:
    - Automatically close completed requests older than a certain date.
    - Mark overdue requests and escalate warnings.
- Use Scheduled Apex to automate daily summary reports or maintenance follow-up reminders to managers or technicians.
- This ensures system scalability and process automation.

Step 8: Exception Handling

- Catch and handle exceptions in service classes to provide clear user messages, e.g., “Duplicate request detected.”
- Log unexpected errors for debugging without impacting user experience.
- Provide fail-safe mechanisms during bulk updates or external integrations.

Step 9: Real-World Asynchronous Use Cases

- Design Batch Apex jobs to auto-close stale maintenance requests.
- Use Queueable Apex to assign technicians or update priorities based on workload dynamically.
- Use Future methods to send real-time notifications (email/SMS) when a new request is submitted or updated.
- Use Scheduled Apex for automatic sending of daily maintenance dashboards to facility managers.

Achievements Expected for Home Maintenance Project Phase 5

- Enforce maintenance and repair business rules via Apex code.
- Develop modular, reusable service classes with supporting triggers.
- Build and execute test classes for robustness and code coverage compliance.
- Effectively apply SOQL, SOSL, collections, and control structures.
- Implement asynchronous Apex (future, queueable, batch, scheduled) for scalability.
- Ensure robust and user-friendly exception handling.