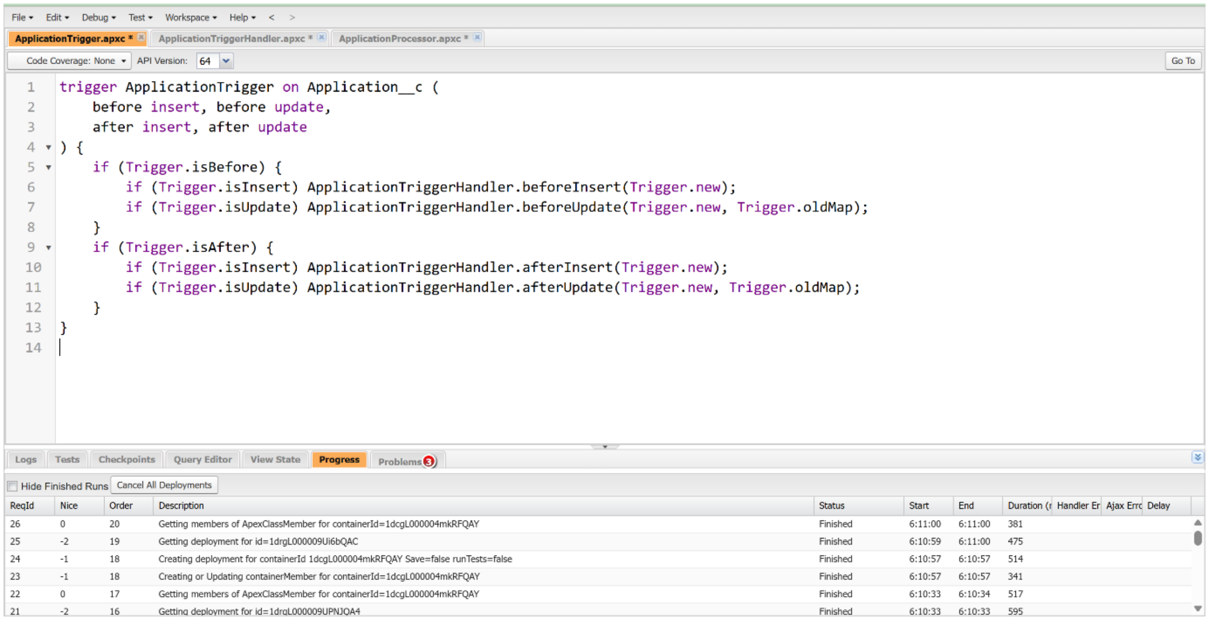
InternLink Hub -“A Central Platform for Internships & Placements”

Phase 5: Apex Programming (Developer) — Requirements for InternLink Hub

1.Classes & Objects

- Develop Apex classes to handle core project logic such as application processing, placement status updates, and recruiter notifications.
- Use Salesforce objects such as Student__c, Application__c, Placement__c, and Internship__c to store and manage relevant data efficiently.



File
Edit
Debug
Test
Workspace
Help
ApplicationTrigger.apxc *
ApplicationTriggerHandler.apxc *
ApplicationProcessor.apxc *
Code Coverage: None
API Version: 64
Go To

```
trigger ApplicationTrigger on Application__c (
    before insert, before update,
    after insert, after update
) {
    if (Trigger.isBefore) {
        if (Trigger.isInsert) ApplicationTriggerHandler.beforeInsert(Trigger.new);
        if (Trigger.isUpdate) ApplicationTriggerHandler.beforeUpdate(Trigger.new, Trigger.oldMap);
    }
    if (Trigger.isAfter) {
        if (Trigger.isInsert) ApplicationTriggerHandler.afterInsert(Trigger.new);
        if (Trigger.isUpdate) ApplicationTriggerHandler.afterUpdate(Trigger.new, Trigger.oldMap);
    }
}
```

ReqId	Nice	Order	Description	Status	Start	End	Duration (r	Handler Er	Ajax Errc	Delay
26	0	20	Getting members of ApexClassMember for containerId=1dcgL000004mkRFQAY	Finished	6:11:00	6:11:00	381			
25	-2	19	Getting deployment for id=1drgL000009Ui6bQAC	Finished	6:10:59	6:11:00	475			
24	-1	18	Creating deployment for containerId 1dcgL000004mkRFQAY Save=false runTests=false	Finished	6:10:57	6:10:57	514			
23	-1	18	Creating or Updating containerMember for containerId=1dcgL000004mkRFQAY	Finished	6:10:57	6:10:57	341			
22	0	17	Getting members of ApexClassMember for containerId=1dcgL000004mkRFQAY	Finished	6:10:33	6:10:34	517			
21	-2	16	Getting deployment for id=1drgL000009UPNJQA4	Finished	6:10:33	6:10:33	595			

2. Apex Triggers

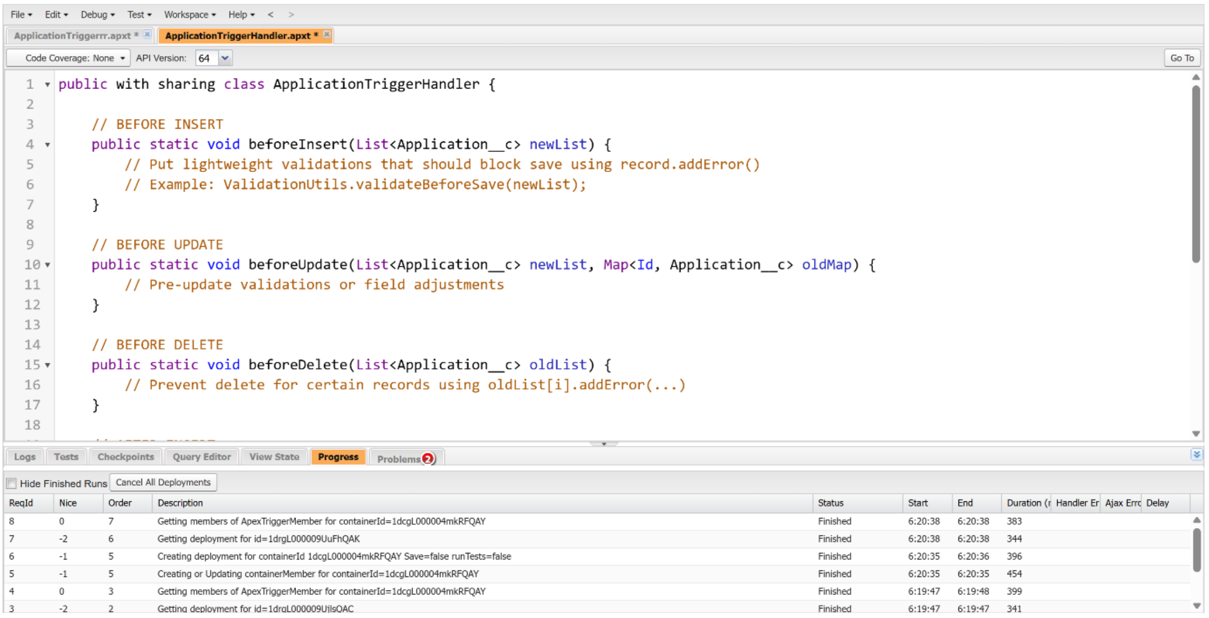
For this project, I have created Apex Triggers to automate actions on the Application__c object whenever records are inserted, updated, or deleted. Triggers are used to enforce complex business logic and ensure proper workflow execution beyond what declarative tools can achieve.

- **Trigger Name:** ApplicationTrigger
- **Object:** Application__c
- **Purpose:** To manage automation when a student applies for an internship, updates their application status, or when records are deleted. This ensures consistent behavior across the system.
- **Logic:**
  - **Before Insert/Update:** Perform data validations before saving records (e.g., ensuring required fields are filled correctly).
  - **After Insert:** Notify recruiters when a new application is created with status "Applied".
  - **After Update:** Update related records (such as Placement__c) or trigger notifications when the application status changes.

    - **Before Delete/After Delete:** Control or track record deletions if needed, ensuring data integrity.

- **Benefit:** Helps enforce business rules, maintain data consistency, and automate recruiter notifications without manual intervention.

- **Description:** This trigger follows the **One Trigger per Object pattern** with a dedicated handler class. The handler manages different events (before insert, after insert, before update, after update, etc.), ensuring the logic is clean, reusable, and bulkified to handle multiple records efficiently.



```
public with sharing class ApplicationTriggerHandler {

    // BEFORE INSERT
    public static void beforeInsert(List<Application__c> newList) {
        // Put lightweight validations that should block save using record.addError()
        // Example: ValidationUtils.validateBeforeSave(newList);
    }

    // BEFORE UPDATE
    public static void beforeUpdate(List<Application__c> newList, Map<Id, Application__c> oldMap) {
        // Pre-update validations or field adjustments
    }

    // BEFORE DELETE
    public static void beforeDelete(List<Application__c> oldList) {
        // Prevent delete for certain records using oldList[i].addError(...)
    }

```

ReqId	Nice	Order	Description	Status	Start	End	Duration (r	Handler Er	Ajax Errc	Delay
8	0	7	Getting members of ApexTriggerMember for containerId=1dcgL000004mkRFQAY	Finished	6:20:38	6:20:38	383			
7	-2	6	Getting deployment for id=1drgL000009UuFhQAK	Finished	6:20:38	6:20:38	344			
6	-1	5	Creating deployment for containerId 1dcgL000004mkRFQAY Save=false runTests=false	Finished	6:20:35	6:20:36	396			
5	-1	5	Creating or Updating containerMember for containerId=1dcgL000004mkRFQAY	Finished	6:20:35	6:20:35	454			
4	0	3	Getting members of ApexTriggerMember for containerId=1dcgL000004mkRFQAY	Finished	6:19:47	6:19:48	399			
3	-2	2	Getting deployment for id=1drgL000009UjlsQAC	Finished	6:19:47	6:19:47	341			

---

3. Batch Apex

For this project, I have created a Batch Apex job to handle large-scale data processing that cannot be executed synchronously. Batch Apex is used to process thousands of records efficiently in manageable chunks, ensuring automation and reporting for the InternLink Hub system.

- **Batch Class Name:** CreatePlacementsBatch

- **Object(s):** Application__c, Placement__c, Student__c, Internship__c

- **Purpose:** To automatically create Placement__c records for applications with status **“Applied”** where no placement exists, and to reconcile placement statuses in bulk. This ensures that placement data remains accurate and up-to-date across the system.

- **Logic:**

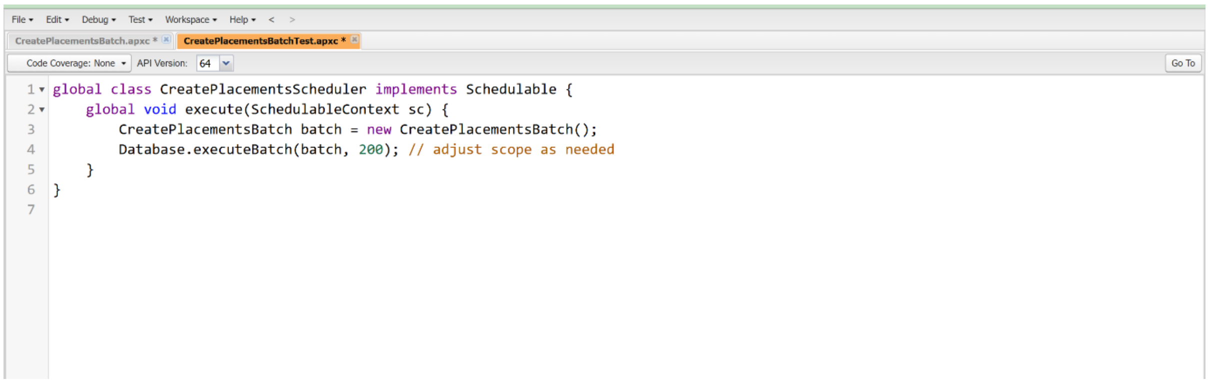
    - **Start:** Select all Application__c records with Status__c = 'Applied' and no related placement.

    - **Execute:** For each batch of records:

        - Collect Student and Internship details.

        - Create and insert missing Placement records.
        - Capture errors and log them for reporting.
    - **Finish:** Send a summary of the job (placements created, errors logged) and optionally chain follow-up processes such as notifications.

- **Benefit:** Improves efficiency by automating repetitive tasks, eliminates manual errors in placement creation, and ensures the system can process large volumes of records without hitting governor limits.
- **Description:** This Batch Apex job is designed following best practices (bulkification, error handling, and logging). It runs either on-demand by an admin or through a scheduled job (e.g., nightly), providing real-time accuracy in placement records and consistent reporting.



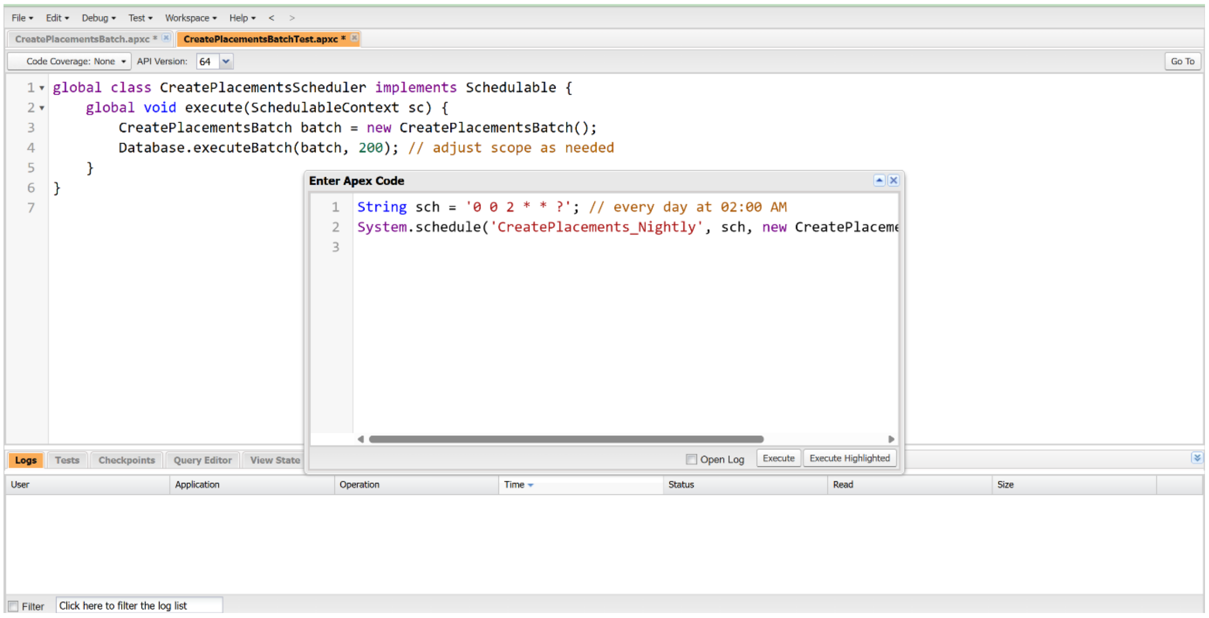
```
global class CreatePlacementsScheduler implements Schedulable {
    global void execute(SchedulableContext sc) {
        CreatePlacementsBatch batch = new CreatePlacementsBatch();
        Database.executeBatch(batch, 200); // adjust scope as needed
    }
}
```

---

4. Asynchronous Processing

For this project, I have implemented Asynchronous Processing to handle time-consuming operations in the background without affecting user experience. This ensures efficiency, scalability, and compliance with Salesforce governor limits.

- **Techniques Used:** Batch Apex, Queueable Apex, Scheduled Apex, Future Methods.
- **Purpose:** To process large volumes of records, perform background tasks, and automate scheduled processes without blocking the main transaction.
- **Logic:**
    - **Batch Apex:** Large dataset processing in chunks.
    - **Queueable Apex:** Smaller, queued jobs for notifications or dependent tasks.
    - **Scheduled Apex:** Run jobs at specific times (e.g., nightly updates).
    - **Future Methods:** Perform asynchronous callouts or background updates.
- **Benefit:** Improves system performance, enables automation, reduces manual work, and ensures scalable processing for growing datasets.
- **Description:** Asynchronous processing in InternLink Hub enables reliable execution of background tasks, freeing resources for user operations while maintaining data accuracy and efficiency.



```
global class CreatePlacementsScheduler implements Schedulable {
    global void execute(SchedulableContext sc) {
        CreatePlacementsBatch batch = new CreatePlacementsBatch();
        Database.executeBatch(batch, 200); // adjust scope as needed
    }
}
```

Enter Apex Code

```
String sch = '0 0 2 * * ?'; // every day at 02:00 AM
System.schedule('CreatePlacements_Nightly', sch, new CreatePlaceme
```

Here's the **documentation-style entry** for **Exception Handling** in the same format as your earlier sections:

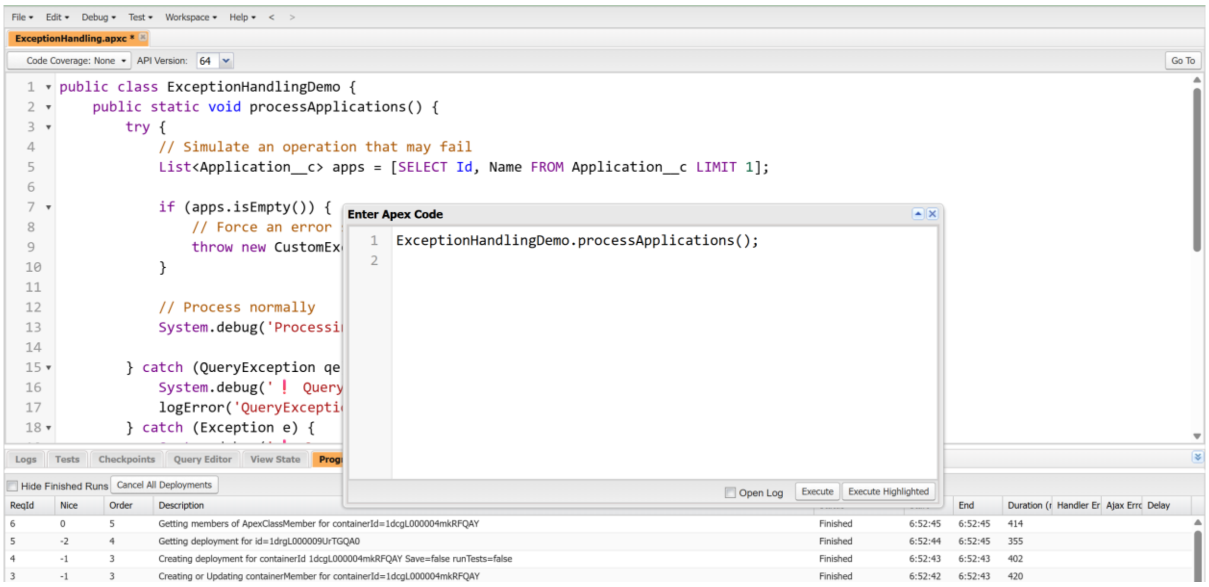
---

5. Exception Handling

For this project, I have implemented Exception Handling to ensure robust error management across all Apex classes, triggers, and asynchronous processes in the InternLink Hub system. Exception handling improves system stability, provides meaningful error feedback, and ensures smooth execution of automation even when unexpected errors occur.

- **Purpose:** To detect, manage, and log errors gracefully without disrupting the system processes or user experience.
- **Logic:**
  - Use try-catch blocks in Apex classes, triggers, and batch jobs to handle runtime exceptions.
  - Catch specific exceptions where possible (e.g., DmlException, QueryException) and handle accordingly.
  - Log errors with relevant context into a custom object (e.g., Error_Log__c) or send email alerts to admins.
  - Maintain user-friendly error messages without exposing technical details.
- **Benefit:**
  - Improves system reliability and stability.
  - Helps identify issues quickly through logs and alerts.
  - Prevents full process failure by handling errors gracefully.
- **Description:** Exception handling in InternLink Hub follows best practices for Salesforce development, ensuring that all automated processes handle errors effectively, log

important details for troubleshooting, and notify appropriate stakeholders without breaking the user experience.


File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
ExceptionHandling.apxc *
Code Coverage: None ▾ API Version: 64
Go To

```
public class ExceptionHandlingDemo {
    public static void processApplications() {
        try {
            // Simulate an operation that may fail
            List<Application__c> apps = [SELECT Id, Name FROM Application__c LIMIT 1];

            if (apps.isEmpty()) {
                // Force an error
                throw new CustomEx
            }

            // Process normally
            System.debug('Processi

        } catch (QueryException qe
            System.debug('❗ Query
            logError('QueryExcepti
        } catch (Exception e) {
```

Enter Apex Code

```
ExceptionHandlingDemo.processApplications();
```

ReqId	Nice	Order	Description			End	Duration (r	Handler Er	Ajax Errc	Delay
6	0	5	Getting members of ApexClassMember for containerId=1dcgL000004mkRFQAY	Finished	6:52:45	6:52:45	414			
5	-2	4	Getting deployment for id=1drgL000009UrTGQA0	Finished	6:52:44	6:52:45	355			
4	-1	3	Creating deployment for containerId 1dcgL000004mkRFQAY Save=false runTests=false	Finished	6:52:43	6:52:43	402			
3	-1	3	Creating or Updating containerMember for containerId=1dcgL000004mkRFQAY	Finished	6:52:42	6:52:43	420			

.