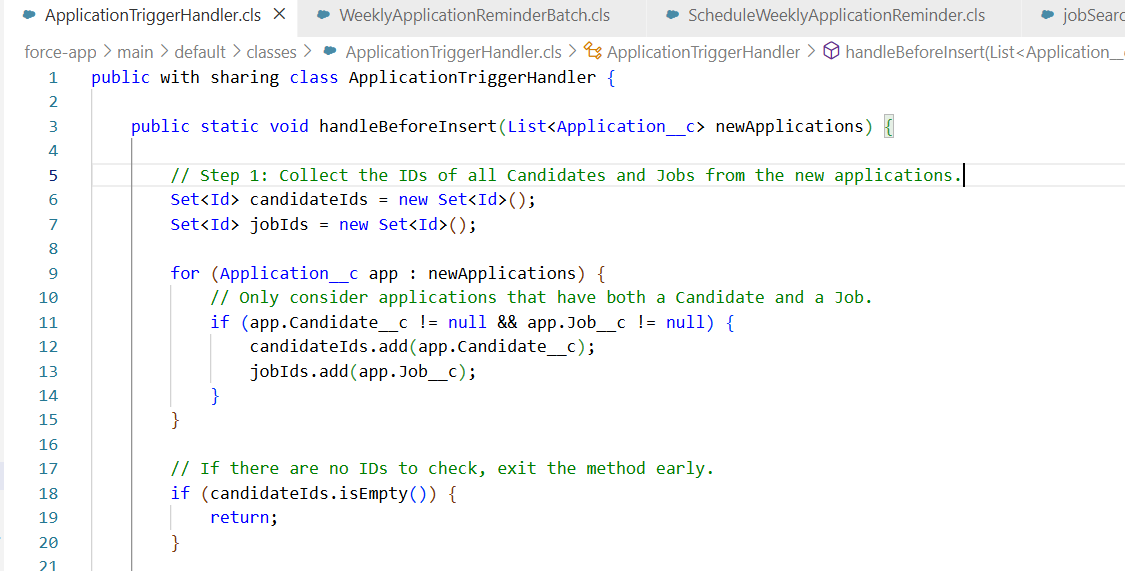
**JOB RECRUITMENT & HIRING MANAGEMENT SYSTEM**

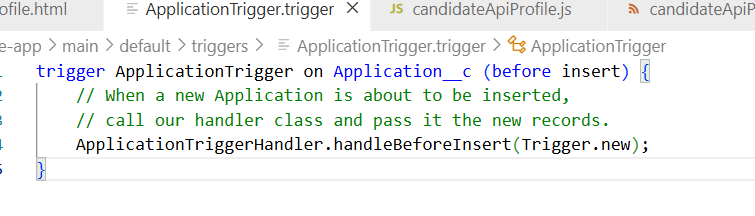
**NAME: KOLLI JAGAN MOHAN RAO**

**Phase 5: Apex Programming (Developer)**

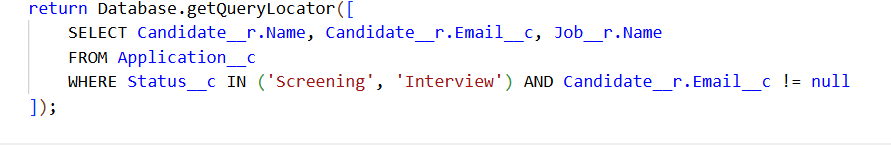
* **Classes & Objects**:
  + We created several Apex **classes** to contain our business logic. A key example is the **ApplicationTriggerHandler** class, which acts as a service class for our trigger. An **object** (in the programming sense) of this class is never instantiated with the new keyword because all its methods were defined as static. This static approach is a common pattern for trigger handlers.



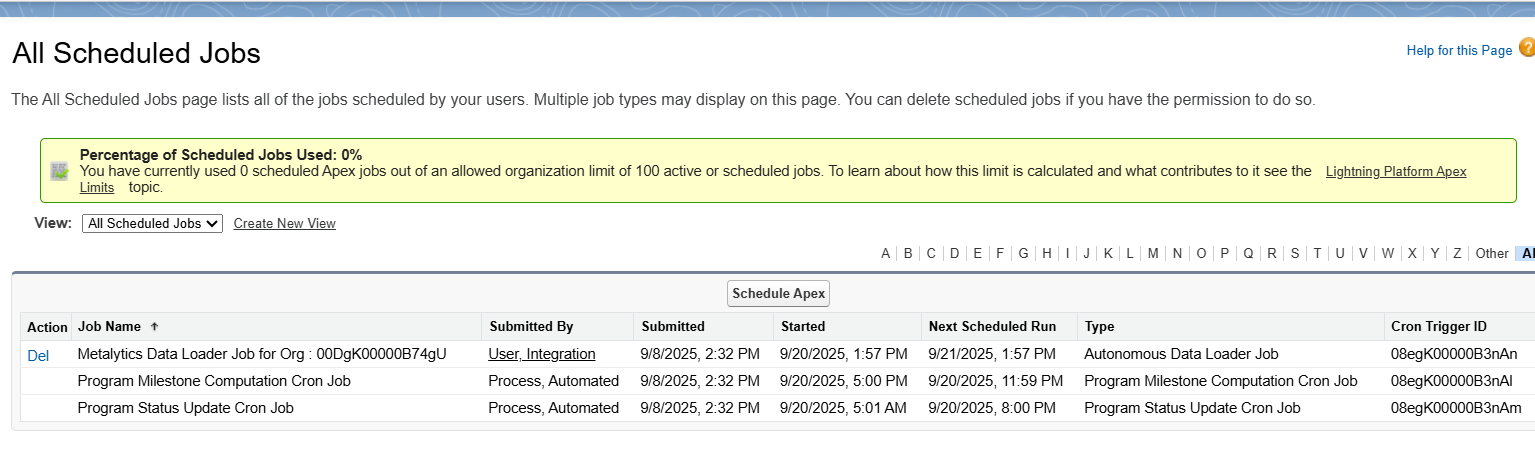
* **Apex Triggers (before/after insert/update/delete)**:
  + We implemented one **Apex Trigger** named ApplicationTrigger on the Application\_\_c object.
  + It was configured to run in the **before insert** context. This context was chosen specifically so we could validate the data and prevent the duplicate record from ever being saved to the database.

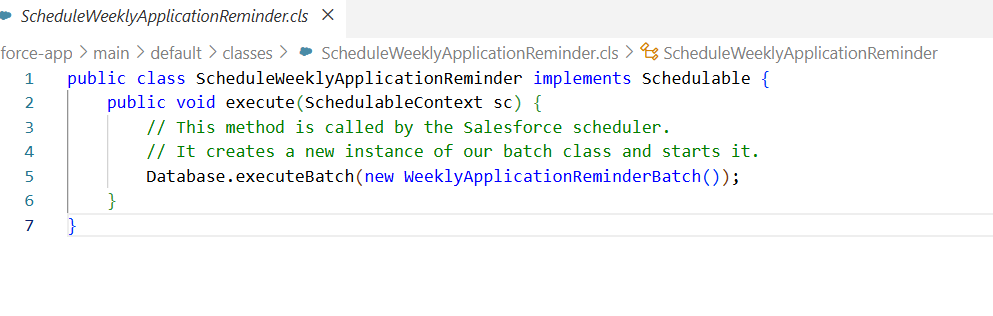


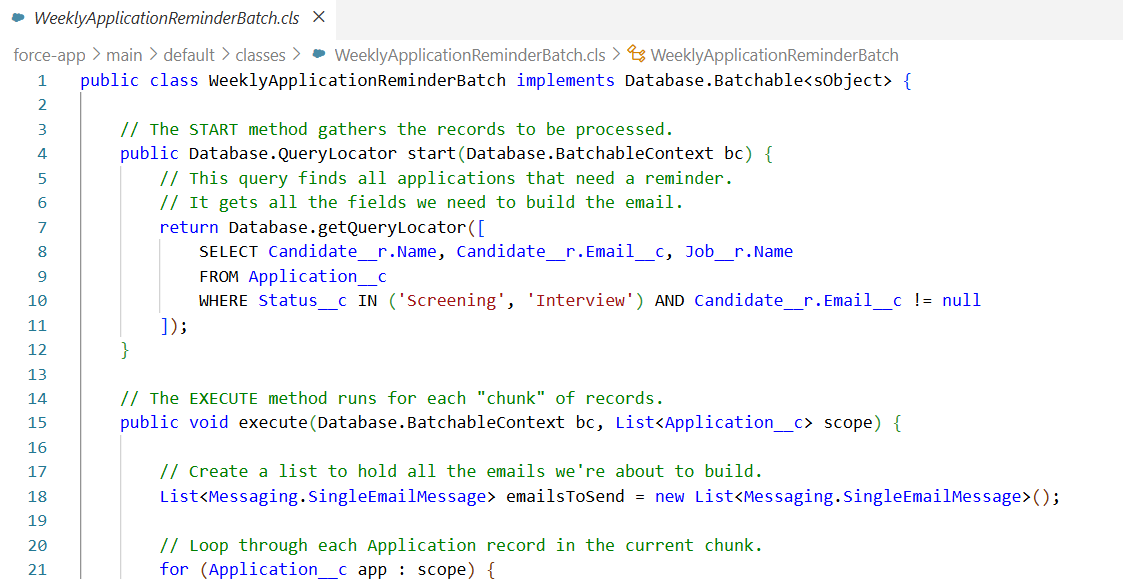
* **SOQL & SOSL**:
  + **SOQL**: We used the Salesforce Object Query Language (SOQL) extensively.
    - In the ApplicationTriggerHandler, a SOQL query was used to find existing Application\_\_c records that matched the candidate and job IDs of the records being inserted: SELECT Candidate\_\_c, Job\_\_c FROM Application\_\_c WHERE Candidate\_\_c IN :candidateIds AND Job\_\_c IN :jobIds
    - In the WeeklyApplicationReminderBatch, a SOQL query in the start method was used to gather the records for processing: SELECT Candidate\_\_r.Name, ... FROM Application\_\_c WHERE Status\_\_c IN ('Screening', 'Interview')
  + **SOSL**: We did **not** use SOSL (Salesforce Object Search Language) as our queries were targeted at known fields on specific objects, for which SOQL is the appropriate tool.



* **Control Statements**:
  + We used for loops to iterate over our collections of records (e.g., for (Application\_\_c app : newApplications)).
  + We used if statements to apply conditional logic, such as checking if an application's key already existed (if (existingKeys.contains(key))) or checking the HTTP status code in our API callout (if (response.getStatusCode() == 200)).
* **Batch Apex**:
  + We created a **Batch Apex** class, WeeklyApplicationReminderBatch, which implements the Database.Batchable interface. This class was designed to process a large number of Application records, find those in progress, and send follow-up emails.
* **Scheduled Apex**:
  + We created a **Scheduled Apex** class, ScheduleWeeklyApplicationReminder, which implements the Schedulable interface. Its sole purpose is to instantiate and execute our WeeklyApplicationReminderBatch class. We then scheduled this class to run weekly through the Salesforce UI.







* **Test Classes**:
  + While we did not write the test classes in our step-by-step guide, it was a required deliverable in the project plan. A proper implementation would involve creating test classes (e.g., ApplicationTriggerHandler\_Test) with the @isTest annotation to create sample data, execute the methods, and use System.assertEquals() to verify that the logic works correctly and meets the 75% code coverage requirement for deployment.



