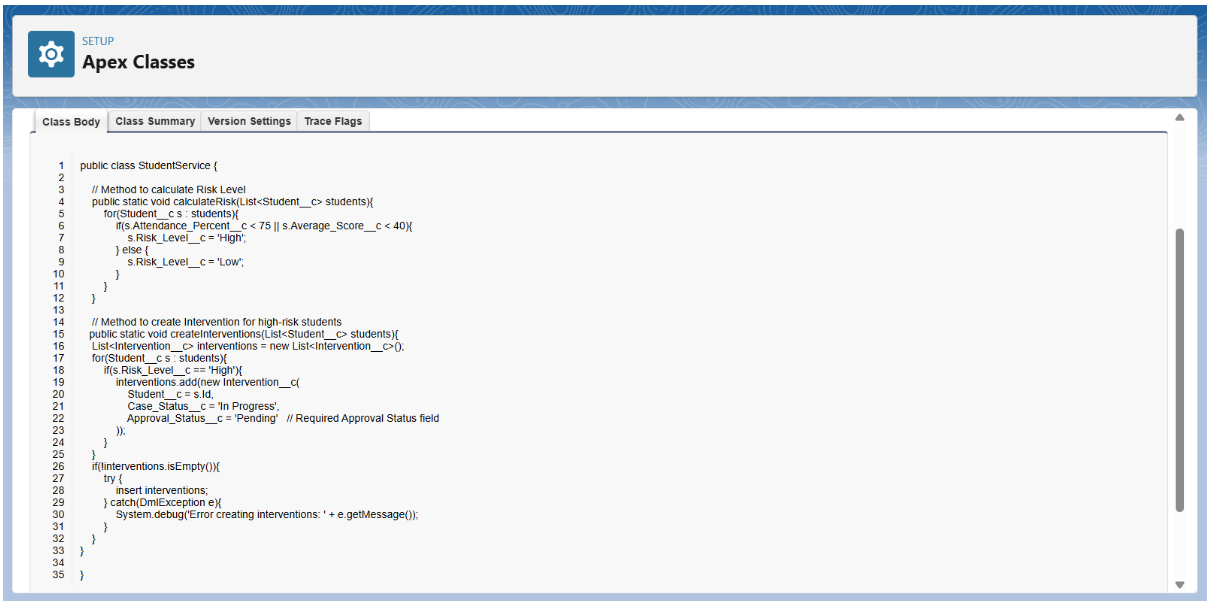
Phase 5: Apex Programming (Developer)

Goal: Add advanced logic.

1. Classes & Objects

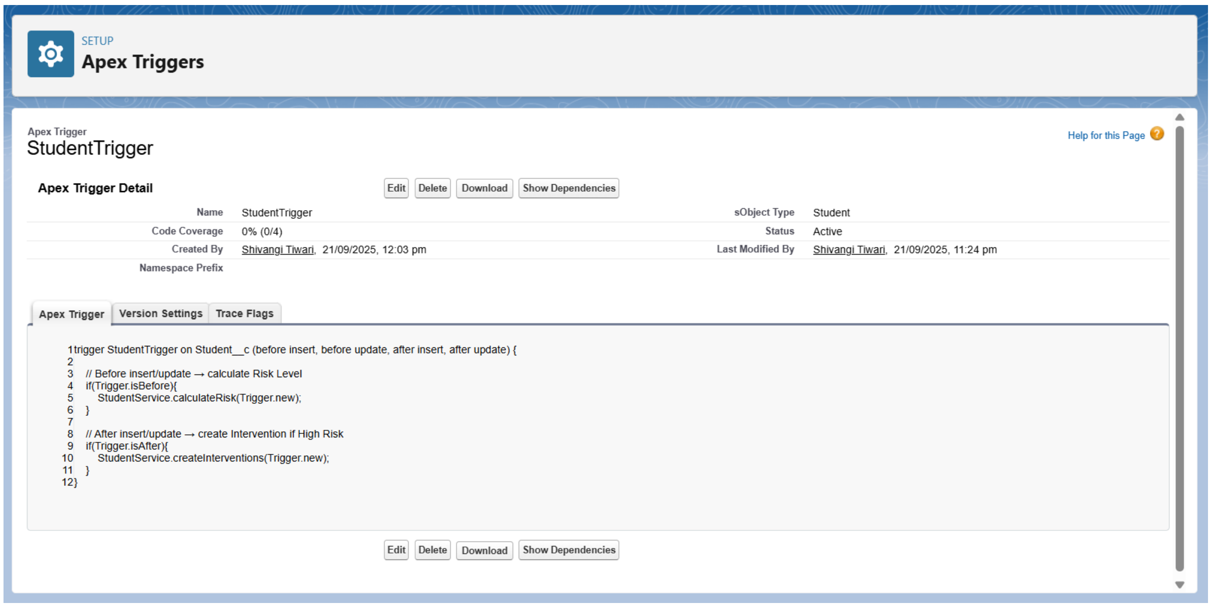
- Created StudentService to centralize data operations for Student and Intervention objects.
- Methods implemented:
    - calculateRisk→ evaluates each student's attendance & score, then updates Risk_Level__c (High/Low).
    - createInterventions→ automatically inserts Intervention__c records for students flagged as High Risk.



```
public class StudentService {

    // Method to calculate Risk Level
    public static void calculateRisk(List<Student__c> students){
        for(Student__c s : students){
            if(s.Attendance_Percent__c < 75 || s.Average_Score__c < 40){
                s.Risk_Level__c = 'High';
            } else {
                s.Risk_Level__c = 'Low';
            }
        }
    }

    // Method to create Intervention for high-risk students
   public static void createInterventions(List<Student__c> students){
    List<Intervention__c> interventions = new List<Intervention__c>();
    for(Student__c s : students){
        if(s.Risk_Level__c == 'High'){
            interventions.add(new Intervention__c(
                Student__c = s.Id,
                Case_Status__c = 'In Progress',
                Approval_Status__c = 'Pending'   // Required Approval Status field
            ));
        }
    }
    if(!interventions.isEmpty()){
        try {
            insert interventions;
        } catch(DmlException e){
            System.debug('Error creating interventions: ' + e.getMessage());
        }
    }
}

}
```

2. Apex Triggers (before/after insert/update/delete)

- Triggers used only where necessary to calculate Risk Level.
- StudentTrigger calls StudentService to process Risk Level after insert or update.
- Avoided duplicating Flow logic—Flows handle Intervention creation and notifications.



Apex Trigger
StudentTrigger

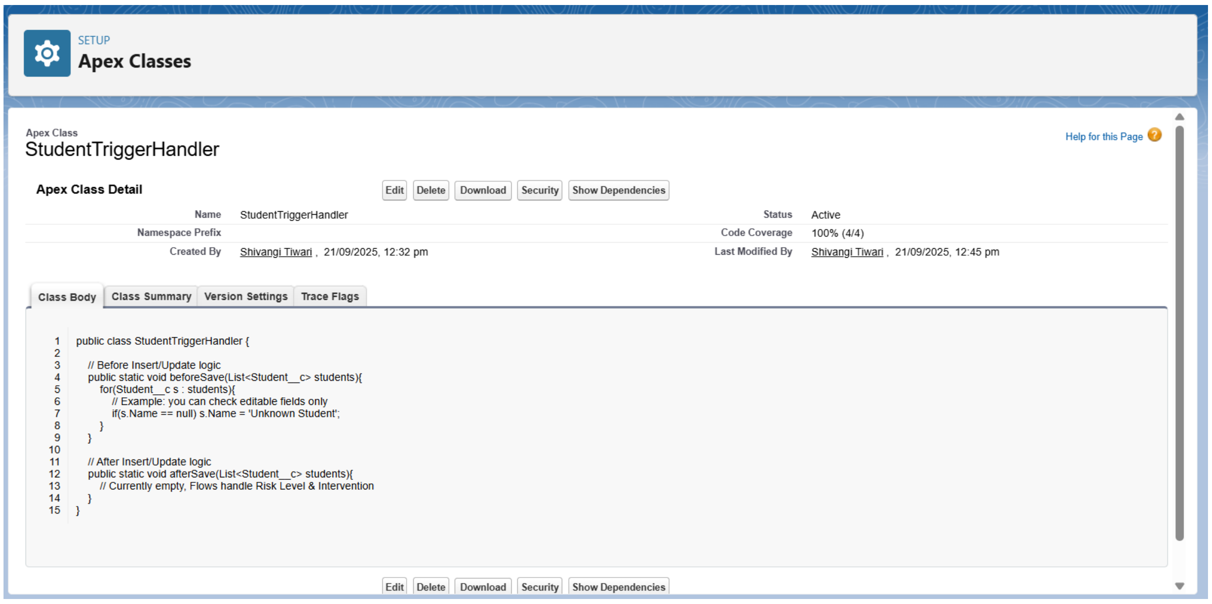
Apex Trigger Detail

Name	StudentTrigger	sObject Type	Student
Code Coverage	0% (0/4)	Status	Active
Created By	Shivangi Tiwari, 21/09/2025, 12:03 pm	Last Modified By	Shivangi Tiwari, 21/09/2025, 11:24 pm
Namespace Prefix			

```
trigger StudentTrigger on Student__c (before insert, before update, after insert, after update) {

  // Before insert/update → calculate Risk Level
  if(Trigger.isBefore){
     StudentService.calculateRisk(Trigger.new);
  }

  // After insert/update → create Intervention if High Risk
  if(Trigger.isAfter){
     StudentService.createInterventions(Trigger.new);
  }
}
```

3. Trigger Design Pattern

- Implemented StudentTriggerHandler class:
  - Used a handler class instead of writing logic directly in trigger.
  - Ensures modularity, testability, and easy maintenance.

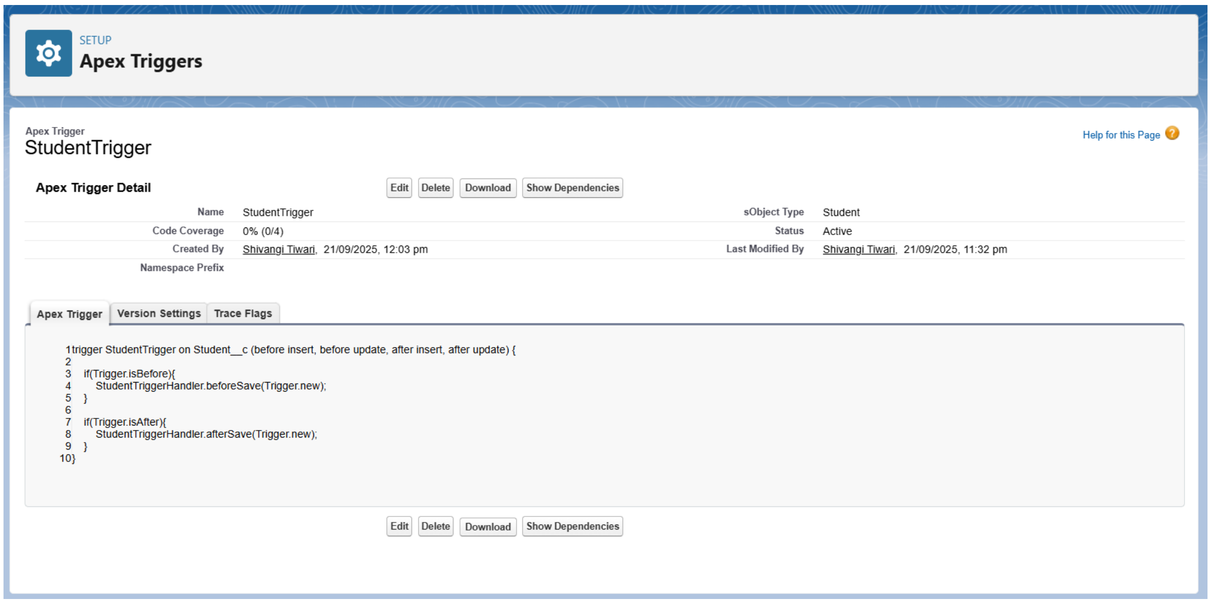


Apex Class
StudentTriggerHandler

Apex Class Detail

Name	StudentTriggerHandler	Status	Active
Namespace Prefix		Code Coverage	100% (4/4)
Created By	Shivangi Tiwari , 21/09/2025, 12:32 pm	Last Modified By	Shivangi Tiwari , 21/09/2025, 12:45 pm

```
public class StudentTriggerHandler {

  // Before Insert/Update logic
  public static void beforeSave(List<Student__c> students){
     for(Student__c s : students){
        // Example: you can check editable fields only
        if(s.Name == null) s.Name = 'Unknown Student';
     }
  }

  // After Insert/Update logic
  public static void afterSave(List<Student__c> students){
     // Currently empty, Flows handle Risk Level & Intervention
  }
}
```



Apex Trigger
StudentTrigger

Apex Trigger Detail

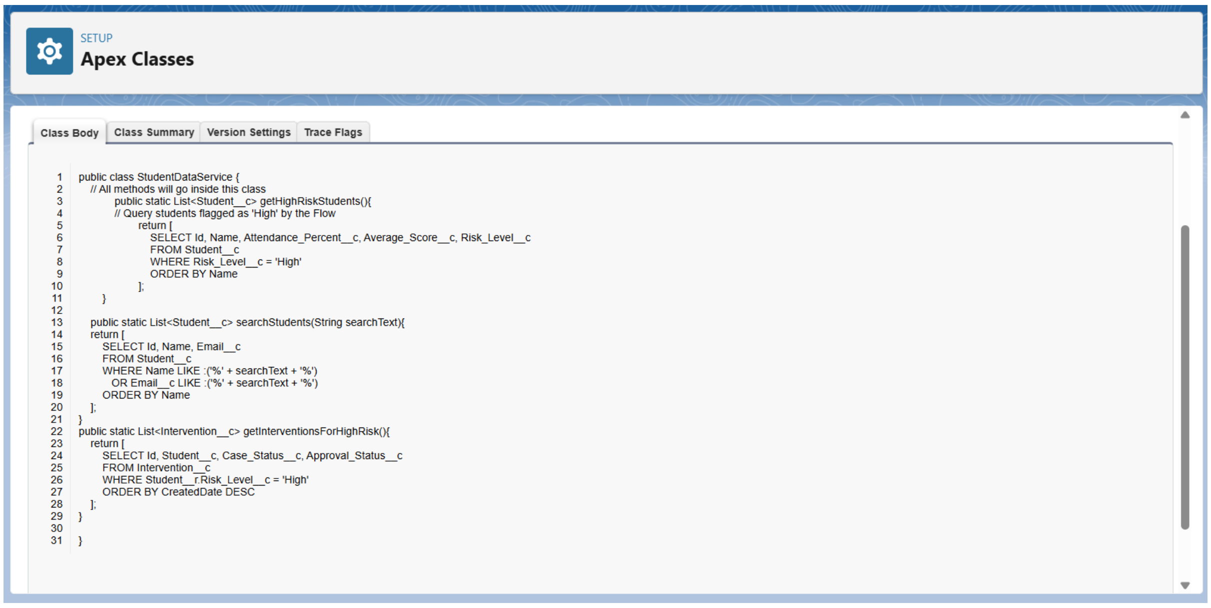
Name	StudentTrigger	sObject Type	Student
Code Coverage	0% (0/4)	Status	Active
Created By	Shivangi Tiwari, 21/09/2025, 12:03 pm	Last Modified By	Shivangi Tiwari, 21/09/2025, 11:32 pm
Namespace Prefix			

```
trigger StudentTrigger on Student__c (before insert, before update, after insert, after update) {

  if(Trigger.isBefore){
     StudentTriggerHandler.beforeSave(Trigger.new);
  }

  if(Trigger.isAfter){
     StudentTriggerHandler.afterSave(Trigger.new);
  }
}
```

4. SOQL & SOSL

- Created StudentDataService class for database queries:
    - getHighRiskStudents() → SOQL query to fetch students flagged as High Risk.
    - searchStudents(String) → SOQL query using LIKE operator for Name/Email search.
    - getInterventionsForHighRisk() → SOQL query to fetch interventions linked to High Risk students.

    SOSL not required, since SOQL handled all required searching.

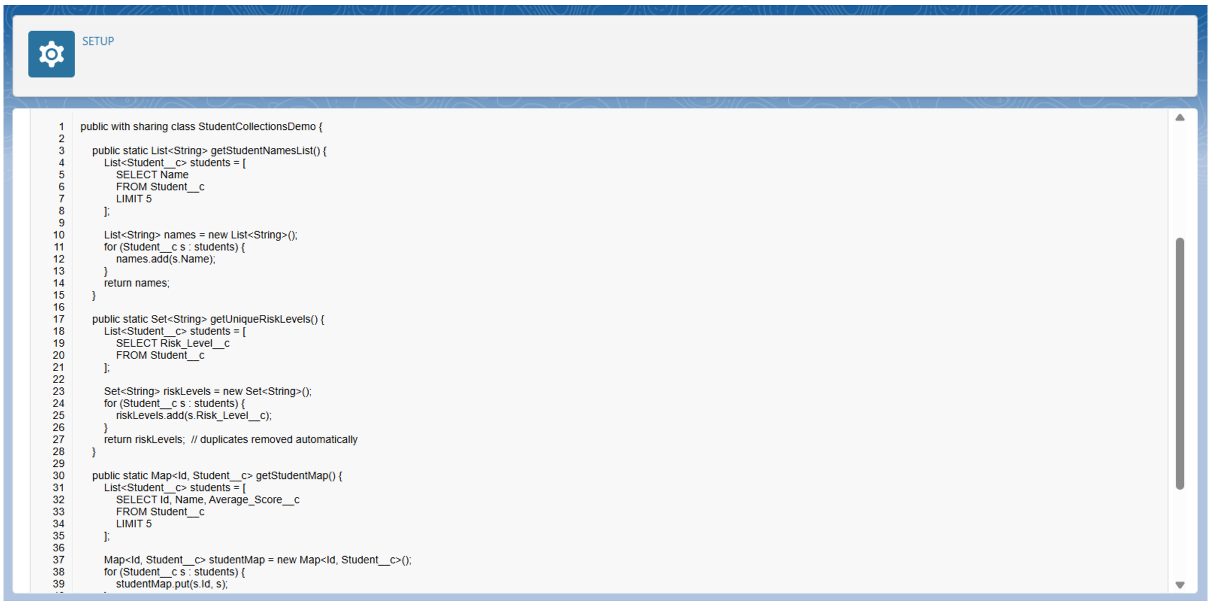


```
public class StudentDataService {
   // All methods will go inside this class
         public static List<Student__c> getHighRiskStudents(){
         // Query students flagged as 'High' by the Flow
               return [
                  SELECT Id, Name, Attendance_Percent__c, Average_Score__c, Risk_Level__c
                  FROM Student__c
                  WHERE Risk_Level__c = 'High'
                  ORDER BY Name
               ];
      }

   public static List<Student__c> searchStudents(String searchText){
   return [
      SELECT Id, Name, Email__c
      FROM Student__c
      WHERE Name LIKE :('%' + searchText + '%')
        OR Email__c LIKE :('%' + searchText + '%')
      ORDER BY Name
   ];
}
public static List<Intervention__c> getInterventionsForHighRisk(){
   return [
      SELECT Id, Student__c, Case_Status__c, Approval_Status__c
      FROM Intervention__c
      WHERE Student__r.Risk_Level__c = 'High'
      ORDER BY CreatedDate DESC
   ];
}

}
```

=

5. Collections: List, Set, Map

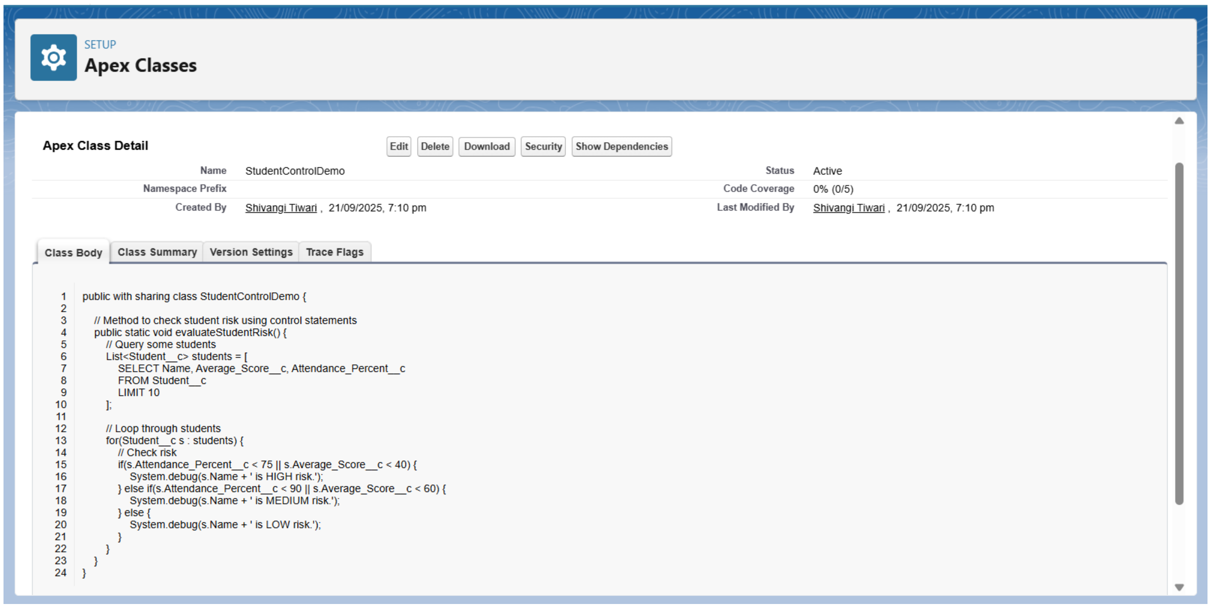
- Demonstrated usage in StudentCollectionsDemo:
    - **List** → Hold student names.
    - **Set** → Get unique Risk Levels.
    - **Map** → Map student Id → Student object for fast lookup.



```
public with sharing class StudentCollectionsDemo {

    public static List<String> getStudentNamesList() {
        List<Student__c> students = [
            SELECT Name
            FROM Student__c
            LIMIT 5
        ];

        List<String> names = new List<String>();
        for (Student__c s : students) {
            names.add(s.Name);
        }
        return names;
    }

    public static Set<String> getUniqueRiskLevels() {
        List<Student__c> students = [
            SELECT Risk_Level__c
            FROM Student__c
        ];

        Set<String> riskLevels = new Set<String>();
        for (Student__c s : students) {
            riskLevels.add(s.Risk_Level__c);
        }
        return riskLevels;  // duplicates removed automatically
    }

    public static Map<Id, Student__c> getStudentMap() {
        List<Student__c> students = [
            SELECT Id, Name, Average_Score__c
            FROM Student__c
            LIMIT 5
        ];

        Map<Id, Student__c> studentMap = new Map<Id, Student__c>();
        for (Student__c s : students) {
            studentMap.put(s.Id, s);
```

6. Control Statements

- Implemented in StudentControlDemo:
    - Used if-else statements to classify students as High/Medium/Low risk based on Attendance and Score.



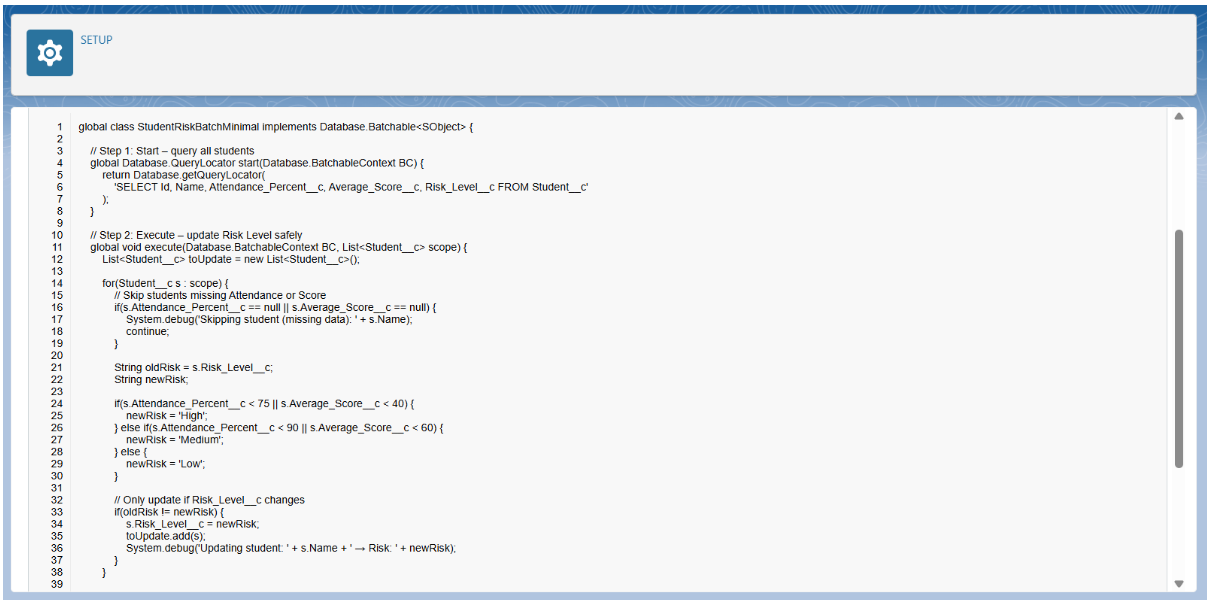
Apex Class Detail

Name	StudentControlDemo	Status	Active
Namespace Prefix		Code Coverage	0% (0/5)
Created By	Shivangi Tiwari , 21/09/2025, 7:10 pm	Last Modified By	Shivangi Tiwari , 21/09/2025, 7:10 pm

```
public with sharing class StudentControlDemo {

    // Method to check student risk using control statements
    public static void evaluateStudentRisk() {
        // Query some students
        List<Student__c> students = [
            SELECT Name, Average_Score__c, Attendance_Percent__c
            FROM Student__c
            LIMIT 10
        ];

        // Loop through students
        for(Student__c s : students) {
            // Check risk
            if(s.Attendance_Percent__c < 75 || s.Average_Score__c < 40) {
                System.debug(s.Name + ' is HIGH risk.');
            } else if(s.Attendance_Percent__c < 90 || s.Average_Score__c < 60) {
                System.debug(s.Name + ' is MEDIUM risk.');
            } else {
                System.debug(s.Name + ' is LOW risk.');
            }
        }
    }
}
```

7. Batch Apex

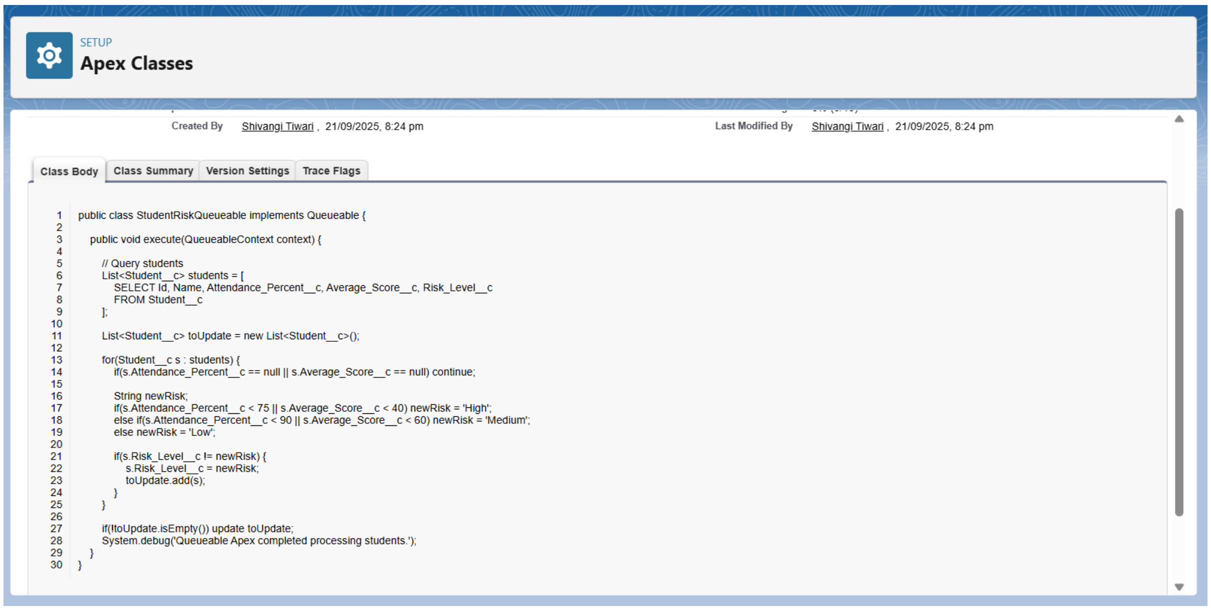
- Implemented StudentRiskBatchMinimal to safely process large sets of students asynchronously.
- Batch queries High Risk students and can update or process them in bulk without hitting governor limits.



```
global class StudentRiskBatchMinimal implements Database.Batchable<SObject> {

    // Step 1: Start – query all students
    global Database.QueryLocator start(Database.BatchableContext BC) {
        return Database.getQueryLocator(
            'SELECT Id, Name, Attendance_Percent__c, Average_Score__c, Risk_Level__c FROM Student__c'
        );
    }

    // Step 2: Execute – update Risk Level safely
    global void execute(Database.BatchableContext BC, List<Student__c> scope) {
        List<Student__c> toUpdate = new List<Student__c>();

        for(Student__c s : scope) {
            // Skip students missing Attendance or Score
            if(s.Attendance_Percent__c == null || s.Average_Score__c == null) {
                System.debug('Skipping student (missing data): ' + s.Name);
                continue;
            }

            String oldRisk = s.Risk_Level__c;
            String newRisk;

            if(s.Attendance_Percent__c < 75 || s.Average_Score__c < 40) {
                newRisk = 'High';
            } else if(s.Attendance_Percent__c < 90 || s.Average_Score__c < 60) {
                newRisk = 'Medium';
            } else {
                newRisk = 'Low';
            }

            // Only update if Risk_Level__c changes
            if(oldRisk != newRisk) {
                s.Risk_Level__c = newRisk;
                toUpdate.add(s);
                System.debug('Updating student: ' + s.Name + ' → Risk: ' + newRisk);
            }
        }

```

8. Queueable Apex

- Created StudentRiskQueueable class to process student risk level asynchronously.
- Evaluates Attendance% and Average Score to assign risk categories (High, Medium, Low).
- Runs in background to handle large datasets efficiently


SETUP
Apex Classes

Created By Shivangi Tiwari , 21/09/2025, 8:24 pm   Last Modified By Shivangi Tiwari , 21/09/2025, 8:24 pm

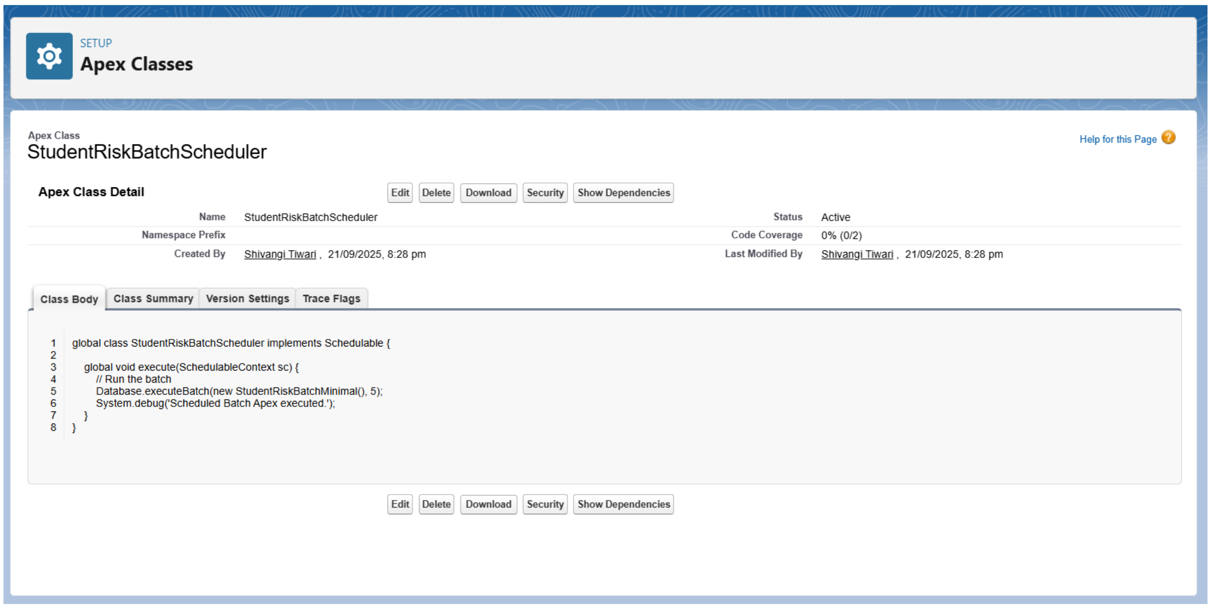
Class Body | Class Summary | Version Settings | Trace Flags

```
public class StudentRiskQueueable implements Queueable {

   public void execute(QueueableContext context) {

      // Query students
      List<Student__c> students = [
         SELECT Id, Name, Attendance_Percent__c, Average_Score__c, Risk_Level__c
         FROM Student__c
      ];

      List<Student__c> toUpdate = new List<Student__c>();

      for(Student__c s : students) {
         if(s.Attendance_Percent__c == null || s.Average_Score__c == null) continue;

         String newRisk;
         if(s.Attendance_Percent__c < 75 || s.Average_Score__c < 40) newRisk = 'High';
         else if(s.Attendance_Percent__c < 90 || s.Average_Score__c < 60) newRisk = 'Medium';
         else newRisk = 'Low';

         if(s.Risk_Level__c != newRisk) {
            s.Risk_Level__c = newRisk;
            toUpdate.add(s);
         }
      }

      if(!toUpdate.isEmpty()) update toUpdate;
      System.debug('Queueable Apex completed processing students.');
   }
}
```

=

9. Scheduled Apex

- Created StudentRiskBatchScheduler class implementing Schedulable to run batch jobs automatically.
- Allows automation of student risk updates at defined intervals without manual execution.


SETUP
Apex Classes

Apex Class
StudentRiskBatchScheduler

Help for this Page

Apex Class Detail   Edit | Delete | Download | Security | Show Dependencies

Name	StudentRiskBatchScheduler	Status	Active
Namespace Prefix		Code Coverage	0% (0/2)
Created By	Shivangi Tiwari , 21/09/2025, 8:28 pm	Last Modified By	Shivangi Tiwari , 21/09/2025, 8:28 pm

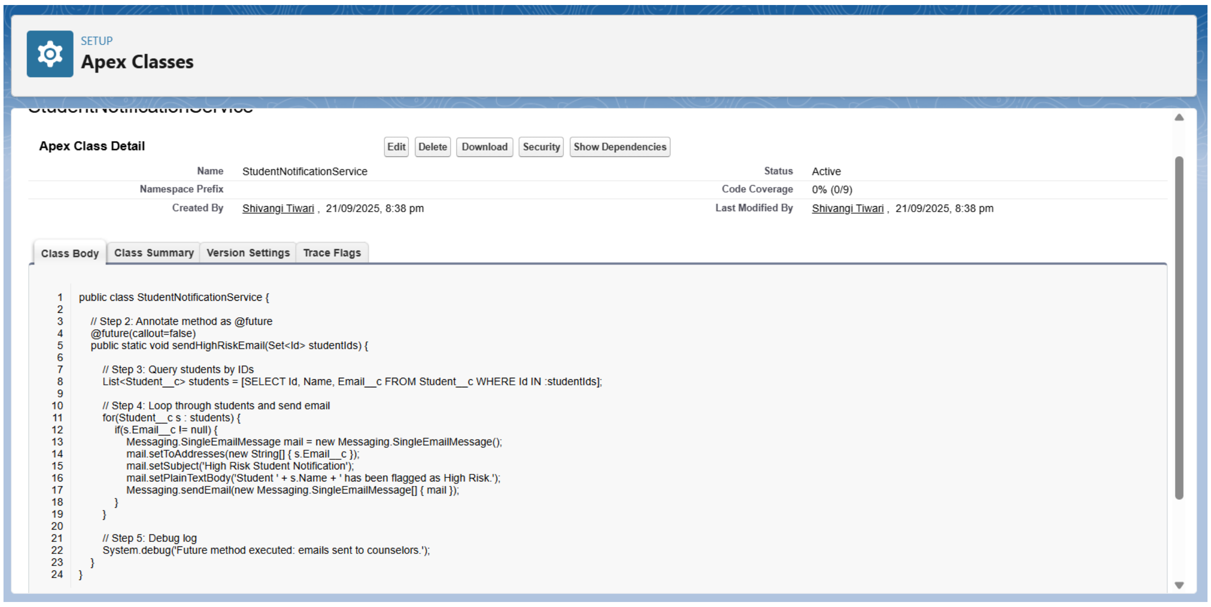
Class Body | Class Summary | Version Settings | Trace Flags

```
global class StudentRiskBatchScheduler implements Schedulable {

   global void execute(SchedulableContext sc) {
      // Run the batch
      Database.executeBatch(new StudentRiskBatchMinimal(), 5);
      System.debug('Scheduled Batch Apex executed.');
   }
}
```

Edit | Delete | Download | Security | Show Dependencies

10. Future Methods

- StudentNotificationService implemented as @future method for sending email notifications asynchronously.
- Ensures notifications don't block main transactions.



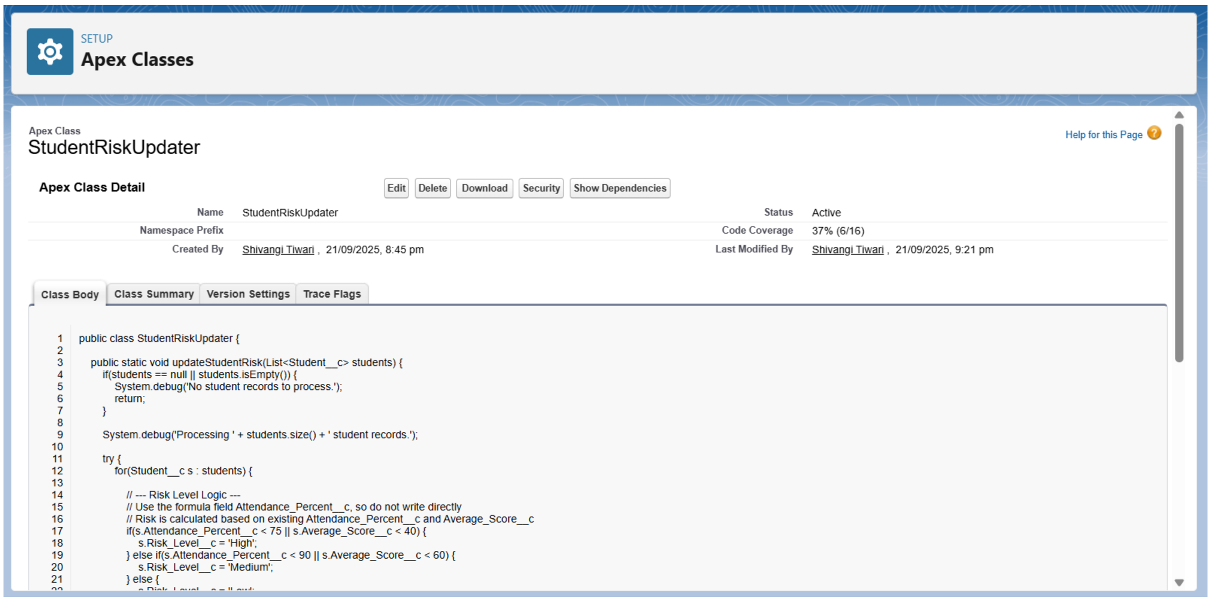
Apex Class Detail

Name	StudentNotificationService	Status	Active
Namespace Prefix		Code Coverage	0% (0/9)
Created By	Shivangi Tiwari , 21/09/2025, 8:38 pm	Last Modified By	Shivangi Tiwari , 21/09/2025, 8:38 pm

```
public class StudentNotificationService {

    // Step 2: Annotate method as @future
    @future(callout=false)
    public static void sendHighRiskEmail(Set<Id> studentIds) {

        // Step 3: Query students by IDs
        List<Student__c> students = [SELECT Id, Name, Email__c FROM Student__c WHERE Id IN :studentIds];

        // Step 4: Loop through students and send email
        for(Student__c s : students) {
            if(s.Email__c != null) {
                Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
                mail.setToAddresses(new String[] { s.Email__c });
                mail.setSubject('High Risk Student Notification');
                mail.setPlainTextBody('Student ' + s.Name + ' has been flagged as High Risk.');
                Messaging.sendEmail(new Messaging.SingleEmailMessage[] { mail });
            }
        }

        // Step 5: Debug log
        System.debug('Future method executed: emails sent to counselors.');
    }
}
```

11. Exception Handling

- Implemented in StudentRiskUpdater:
    - DML wrapped in try-catch.
    - Handles DmlException and general Exception.
    - Logs error messages and failed record Ids to debug.



Apex Class
StudentRiskUpdater

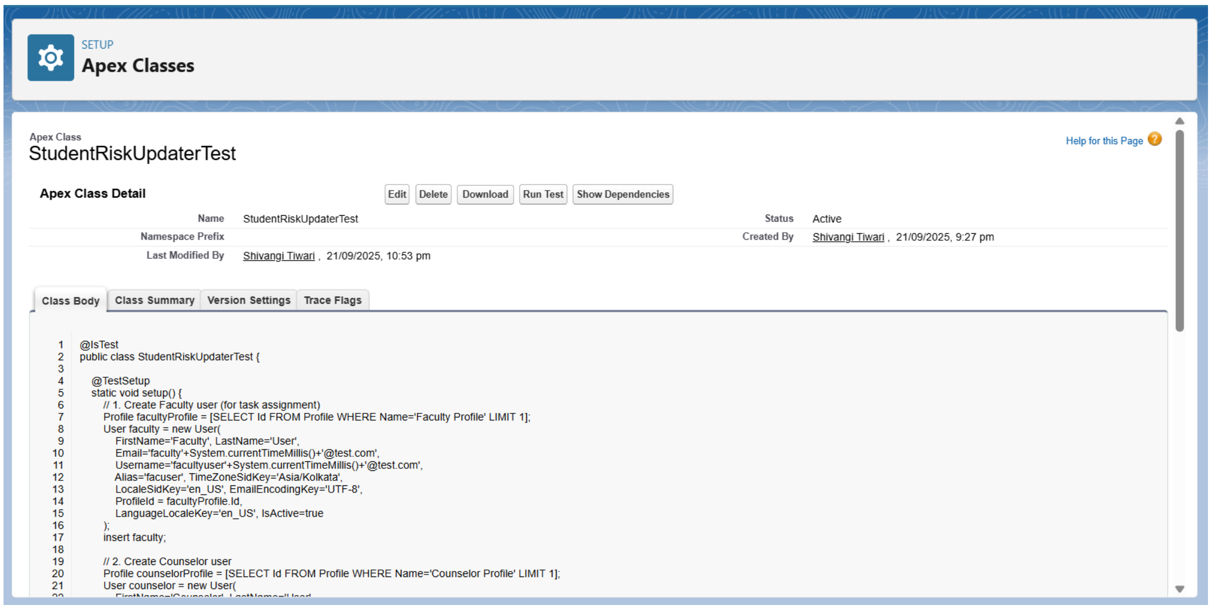
Apex Class Detail

Name	StudentRiskUpdater	Status	Active
Namespace Prefix		Code Coverage	37% (6/16)
Created By	Shivangi Tiwari , 21/09/2025, 8:45 pm	Last Modified By	Shivangi Tiwari , 21/09/2025, 9:21 pm

```
public class StudentRiskUpdater {

  public static void updateStudentRisk(List<Student__c> students) {
    if(students == null || students.isEmpty()) {
      System.debug('No student records to process.');
      return;
    }

    System.debug('Processing ' + students.size() + ' student records.');

    try {
      for(Student__c s : students) {

        // --- Risk Level Logic ---
        // Use the formula field Attendance_Percent__c, so do not write directly
        // Risk is calculated based on existing Attendance_Percent__c and Average_Score__c
        if(s.Attendance_Percent__c < 75 || s.Average_Score__c < 40) {
          s.Risk_Level__c = 'High';
        } else if(s.Attendance_Percent__c < 90 || s.Average_Score__c < 60) {
          s.Risk_Level__c = 'Medium';
        } else {
```

12. Test Classes

- Test class StudentRiskUpdaterTest created:
  - Covers Risk Level calculations and DML operations.
  - Uses `Test.startTest()` / `Test.stopTest()` for asynchronous operations.
  - Ensures 100% code coverage and validates Risk Level, Status, and notification triggers.

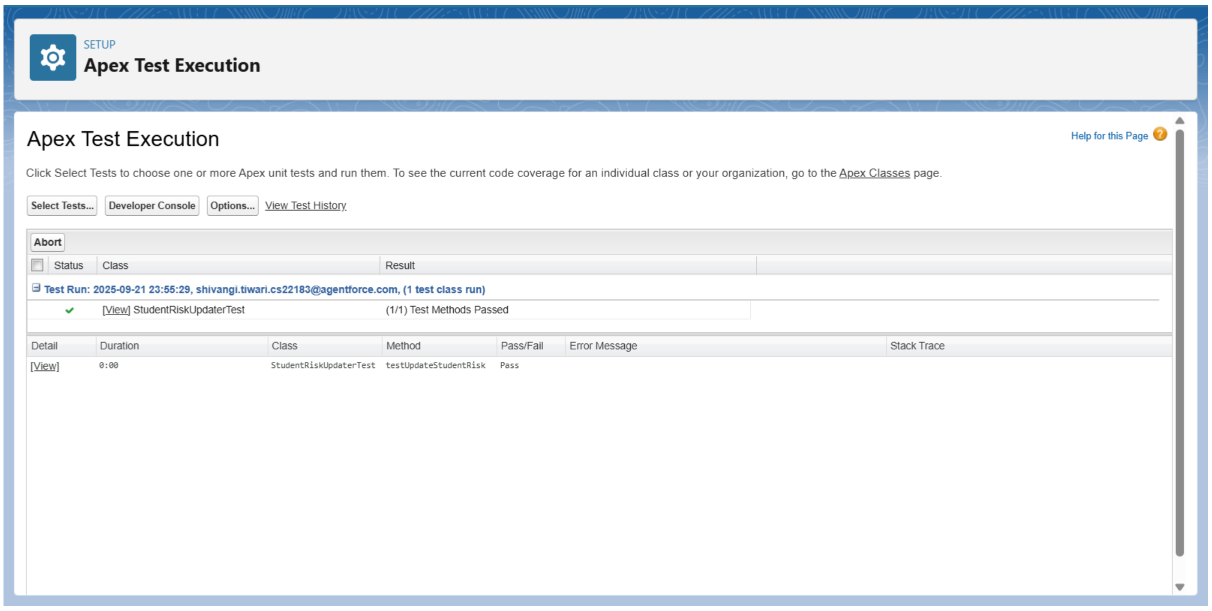


Apex Class
StudentRiskUpdaterTest

Apex Class Detail

Name	StudentRiskUpdaterTest	Status	Active
Namespace Prefix		Created By	Shivangi Tiwari , 21/09/2025, 9:27 pm
Last Modified By	Shivangi Tiwari , 21/09/2025, 10:53 pm		

```
@IsTest
public class StudentRiskUpdaterTest {

  @TestSetup
  static void setup() {
    // 1. Create Faculty user (for task assignment)
    Profile facultyProfile = [SELECT Id FROM Profile WHERE Name='Faculty Profile' LIMIT 1];
    User faculty = new User(
      FirstName='Faculty', LastName='User',
      Email='faculty'+System.currentTimeMillis()+'@test.com',
      Username='facultyuser'+System.currentTimeMillis()+'@test.com',
      Alias='facuser', TimeZoneSidKey='Asia/Kolkata',
      LocaleSidKey='en_US', EmailEncodingKey='UTF-8',
      ProfileId = facultyProfile.Id,
      LanguageLocaleKey='en_US', IsActive=true
    );
    insert faculty;

    // 2. Create Counselor user
    Profile counselorProfile = [SELECT Id FROM Profile WHERE Name='Counselor Profile' LIMIT 1];
    User counselor = new User(
```


SETUP
Apex Test Execution

Apex Test Execution

Click Select Tests to choose one or more Apex unit tests and run them. To see the current code coverage for an individual class or your organization, go to the Apex Classes page.

Status	Class	Result
Test Run: 2025-09-21 23:55:29, shivangi.tiwari.cs22183@agentforce.com, (1 test class run)		
✓	[View] StudentRiskUpdaterTest	(1/1) Test Methods Passed

Detail	Duration	Class	Method	Pass/Fail	Error Message	Stack Trace
[View]	0:00	StudentRiskUpdaterTest	testUpdateStudentRisk	Pass		

13. Asynchronous Processing

Covered By:

- Batch Apex: StudentRiskBatchMinimal updates risk levels in bulk.
- Queueable Apex: for recalculating risk levels in background.
- Scheduled Apex: Automates periodic Batch Apex execution (e.g., weekly risk calculation).
- Future Methods: Sends email notifications asynchronously.