Smart College CRM – Student Management System on SalesforcePhase 5: Apex Programming (Developer)

**Goal: To extend the functionality of Smart College CRM using Apex to handle custom logic for student management, including automated admission processing, fee validation, course enrollment, and notifications, ensuring scalability, reliability, and bulk-safe operations."**

**1. Apex Components**

**A. Apex Classes & Objects**

| **Class Name** | **Purpose** | **Example Logic** |
| --- | --- | --- |
| **AdmissionManager.cls** | Validates applications, updates admission status, triggers notifications. | Assigns admission status “Enrolled” after admin approval, sends email to student. |
| **FeeManager.cls** | Handles fee validation, payment status updates, and reminders. | Updates Fee Status = “Paid” when payment is received, prevents overpayment. |
| **CourseEnrollmentHandler.cls** | Manages course enrollment logic for students. | Validates course capacity, enrolls student, updates StudentCourse junction object. |

Example: AdmissionManager.cls

public class AdmissionManager {

public static void approveAdmission(Id applicationId){

Application\_\_c app = [SELECT Id, Student\_\_c, Status\_\_c FROM Application\_\_c WHERE Id = :applicationId];

if(app.Status\_\_c != 'Approved'){

app.Status\_\_c = 'Approved';

update app;

// Update Student status

Student\_\_c stu = [SELECT Id, Admission\_Status\_\_c FROM Student\_\_c WHERE Id = :app.Student\_\_c];

stu.Admission\_Status\_\_c = 'Enrolled';

update stu;

// Send notification/email (pseudo code)

// Messaging.sendEmail(emailTemplateId, stu.Email\_\_c);

}

}

}

**B. Apex Triggers**

| **Trigger Name** | **Purpose** | **Logic** |
| --- | --- | --- |
| **ApplicationApprovalTrigger** | Ensures Admission Status updates when application is approved. | Calls AdmissionManager.approveAdmission() on update. |
| **FeeValidationTrigger** | Prevents overpayment or negative fee values. | Validates Fee\_\_c Amount before insert/update. |
| **CourseCapacityTrigger** | Prevents course over-enrollment. | Checks StudentCourse\_\_c count vs Course Capacity\_\_c. |

Example Trigger: FeeValidationTrigger

trigger FeeValidationTrigger on Fee\_\_c (before insert, before update) {

for(Fee\_\_c f : Trigger.new){

if(f.Amount\_\_c <= 0){

f.addError('Fee amount must be greater than 0.');

}

}

}

**C. Trigger Design Pattern**

All triggers call **handler classes** instead of writing logic directly in triggers.  
**Benefits:** Bulk-safe, maintainable, scalable, easier testing.

**D. SOQL & SOSL Examples**

* **SOQL:** Fetch all students enrolled in a course.

List<StudentCourse\_\_c> enrollments = [SELECT Student\_\_r.Name, Course\_\_r.Course\_Name\_\_c FROM StudentCourse\_\_c WHERE Course\_\_c = :courseId];

* **SOSL:** Search students by name or email.

List<List<SObject>> results = [FIND 'John\*' IN ALL FIELDS RETURNING Student\_\_c(Name, Email\_\_c), Application\_\_c(Status\_\_c)];

**E. Asynchronous Apex**

| **Type** | **Use Case** | **Example** |
| --- | --- | --- |
| **Batch Apex** | Send bulk fee reminders before due dates | Batch class iterates over Fee\_\_c records with Pending status |
| **Queueable Apex** | Generate admission confirmation emails asynchronously | Queueable class executes email sending for large student batches |
| **Scheduled Apex** | Nightly report of new admissions or pending applications | Scheduled job runs batch/queueable classes daily |
| **Future Methods** | Post-enrollment notifications | Sends welcome email asynchronously |

Example Queueable Class:

public class FeeReminderQueueable implements Queueable {

private Id feeId;

public FeeReminderQueueable(Id fId){ this.feeId = fId; }

public void execute(QueueableContext qc){

Fee\_\_c f = [SELECT Id, Amount\_\_c, Student\_\_c FROM Fee\_\_c WHERE Id = :feeId LIMIT 1];

// send reminder email

}

}

**F. Exception Handling**

* Errors such as over-enrollment, invalid payment, or missing student details are handled with **try-catch** blocks.

try {

CourseEnrollmentHandler.enrollStudent(studentId, courseId);

} catch(Exception e) {

System.debug('Error: ' + e.getMessage());

}

**G. Test Classes**

* Ensure **75%+ code coverage**.
* Validate admissions, fee updates, course enrollment, and async operations.

Example Test Class:

@isTest

public class AdmissionManagerTest {

@isTest static void testApproveAdmission(){

Student\_\_c stu = new Student\_\_c(Name='Test Student');

insert stu;

Application\_\_c app = new Application\_\_c(Student\_\_c=stu.Id, Status\_\_c='Submitted');

insert app;

Test.startTest();

AdmissionManager.approveAdmission(app.Id);

Test.stopTest();

Student\_\_c updatedStu = [SELECT Admission\_Status\_\_c FROM Student\_\_c WHERE Id = :stu.Id];

System.assertEquals('Enrolled', updatedStu.Admission\_Status\_\_c);

}

}

**Deliverables (Phase 5)**

* **Apex Classes: AdmissionManager, FeeManager, CourseEnrollmentHandler**
* **Triggers with handler classes for Application, Fee, and Enrollment validation**
* **Batch, Queueable, Scheduled Apex jobs**
* **Exception handling for safe execution**
* **Unit test classes with high coverage**

**Outcome:**

**Prevents over-enrollment in courses and ensures fee compliance  
 Automates admission approvals, fee updates, and course enrollment  
 Provides scalable async solutions for reminders, reports, and notifications  
 Strengthens reliability with exception handling and test coverage**