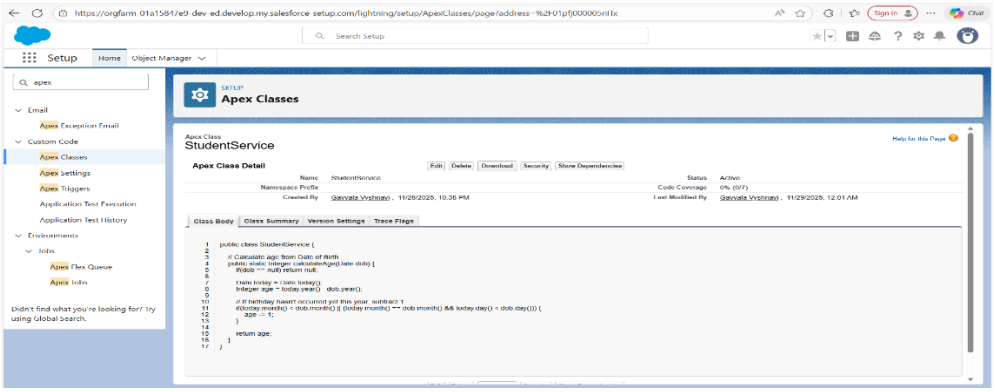
Phase 5: Apex Programming (Developer)

Goal of this Phase

The goal of Phase 5 was to implement **Apex programming features** in Salesforce to support the Student Management System. This included creating **classes, triggers, SOQL/SOSL queries, collections, asynchronous processing, exception handling, and test classes** to ensure the application is efficient, scalable, and reliable.

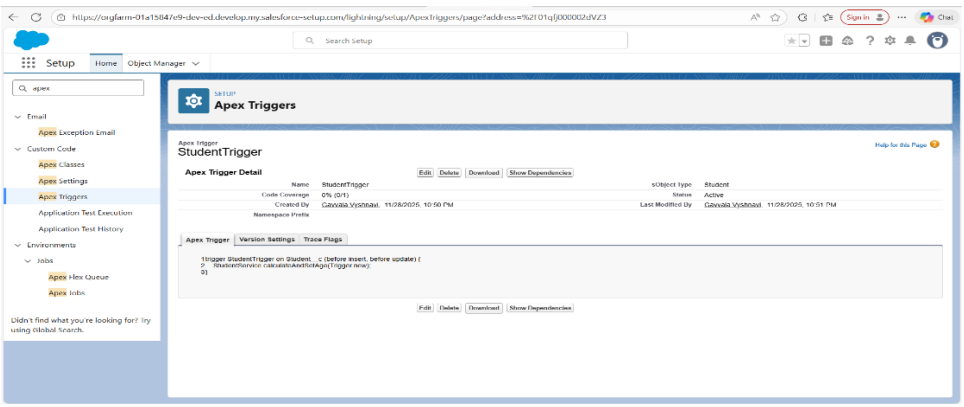
1. Classes & Objects

- Created **StudentService** Apex class to handle business logic.
- Added methods for calculating student age based on Date of Birth and updating records.



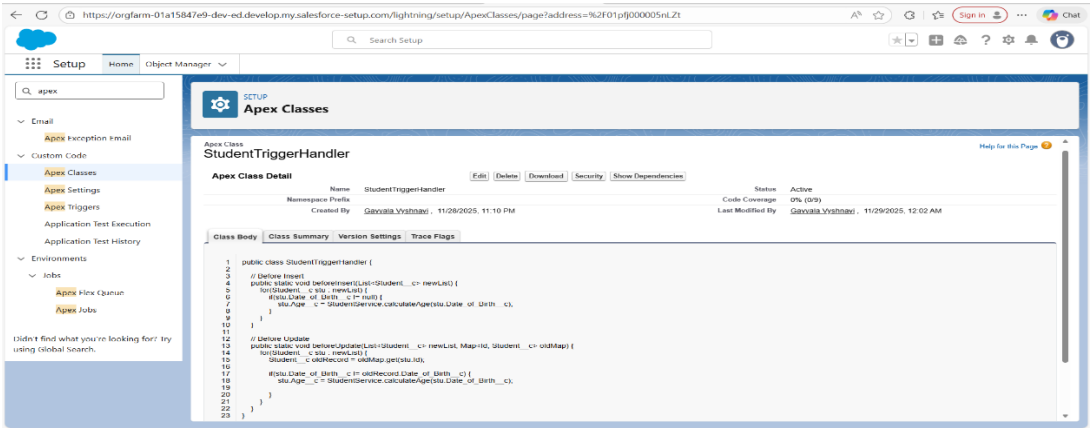
2. Apex Triggers (Before/After Insert/Update/Delete)

- Implemented a **StudentTrigger** to:
  - Calculate Age before insert/update when Date of Birth is set.
  - Prevent invalid updates using before update triggers.
  - Maintain consistency when student records are deleted.



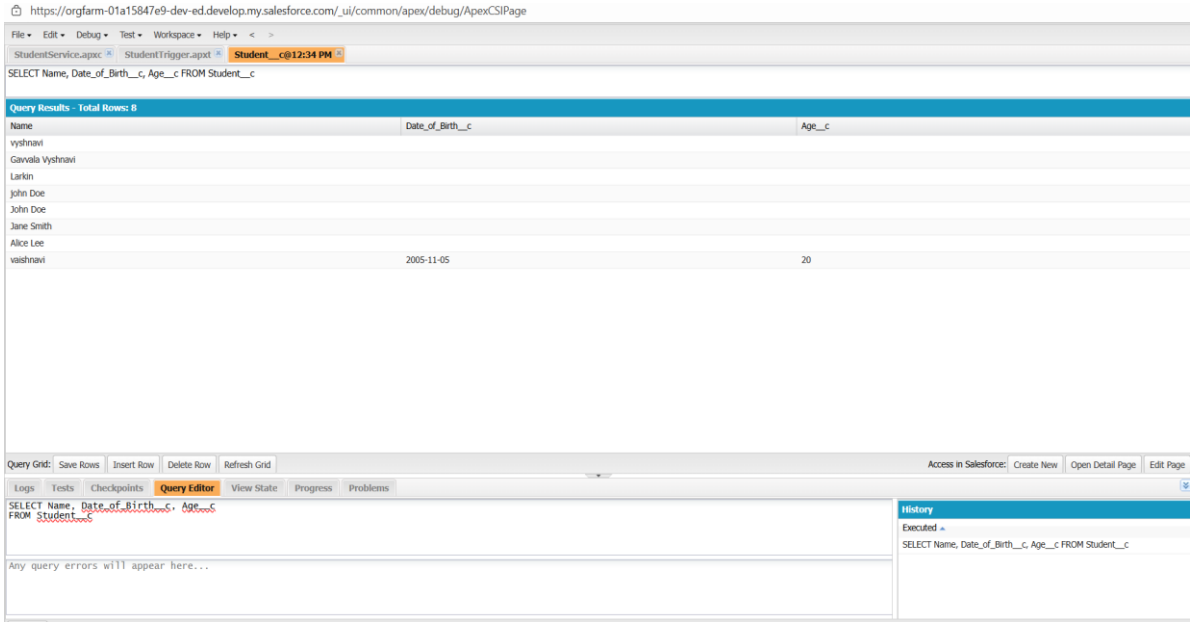
3. Trigger Design Pattern

- Applied **Handler Class Pattern**:
  - Trigger delegates logic to StudentTriggerHandler class.
  - Ensures clean separation of logic, reusable code, and easier maintenance.



4. SOQL & SOSL

- Used **SOQL queries** to fetch Student records with fields like Id, Date_of_Birth__c, and Age__c.
- Demonstrated **SOSL** for searching students by name across multiple fields.

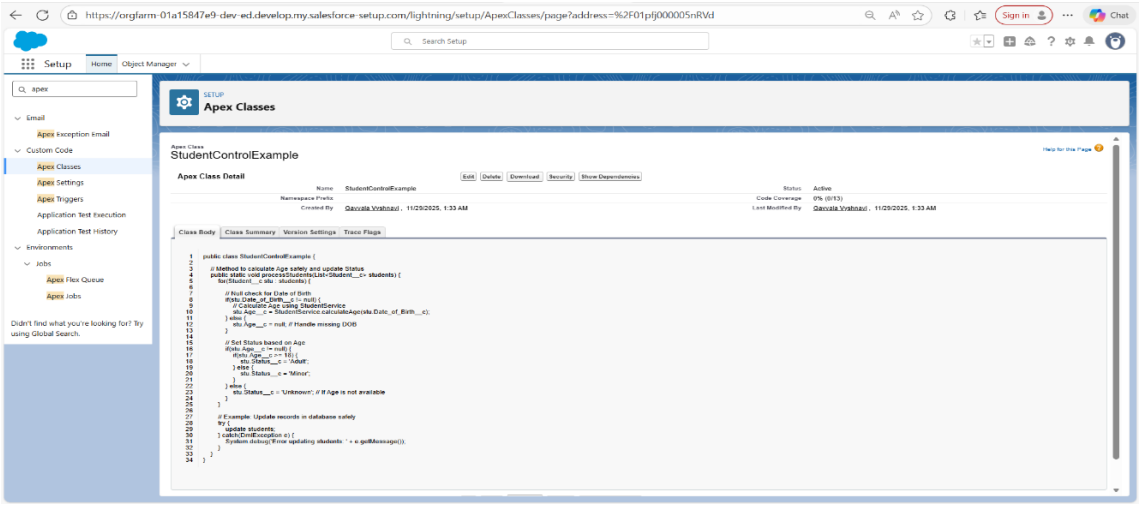


5. Collections (List, Set, Map)

- Used **List<Student__c>** for batch updates.
- Applied **Set<Id>** to handle unique record IDs.
- Implemented **Map<Id, Student__c>** to efficiently compare old and new trigger contexts.

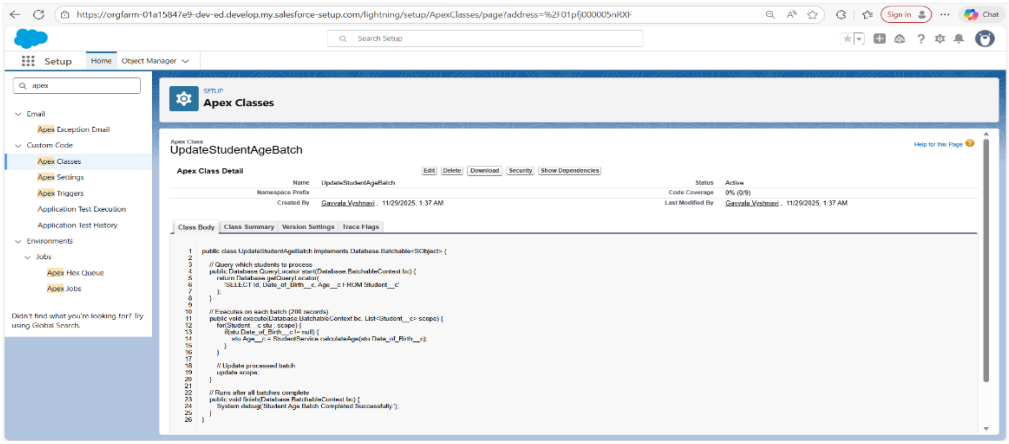
6. Control Statements

- Used **If-Else conditions** for null checks.
- Applied **For loops** to process multiple student records.
- Incorporated **Try-Catch blocks** for exception handling.



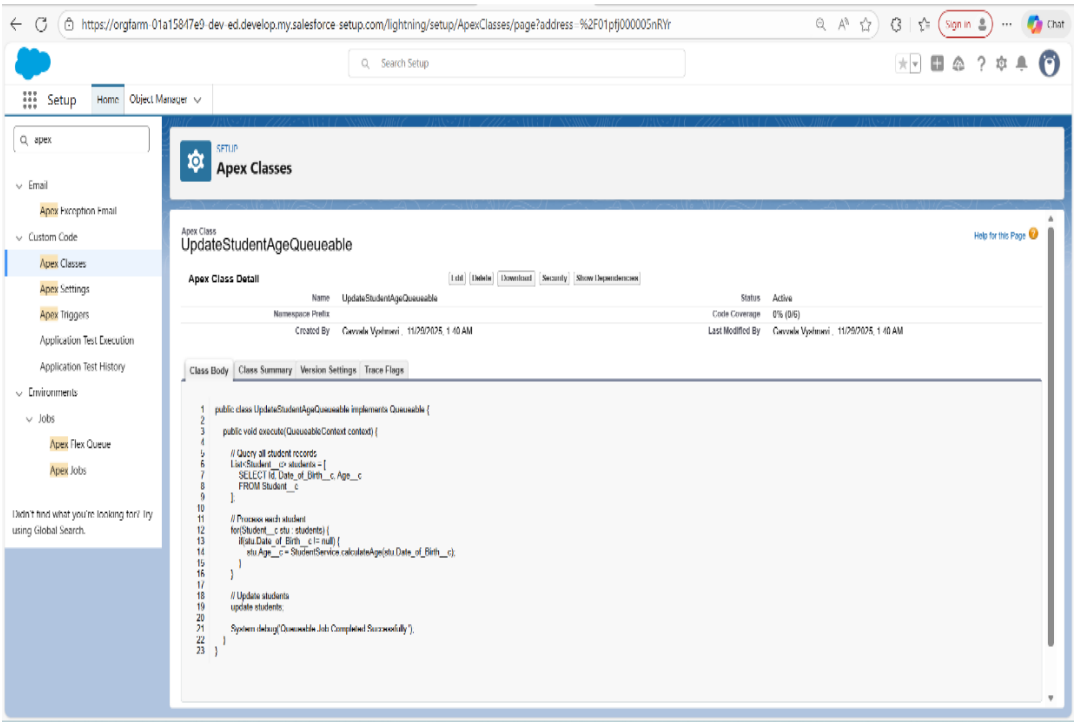
7. Batch Apex

- Created **UpdateStudentAgeBatch** class implementing Database.Batchable<SObject>.
- Processes students in batches to calculate and update Age.
- Scheduled and monitored using **Apex Jobs**.



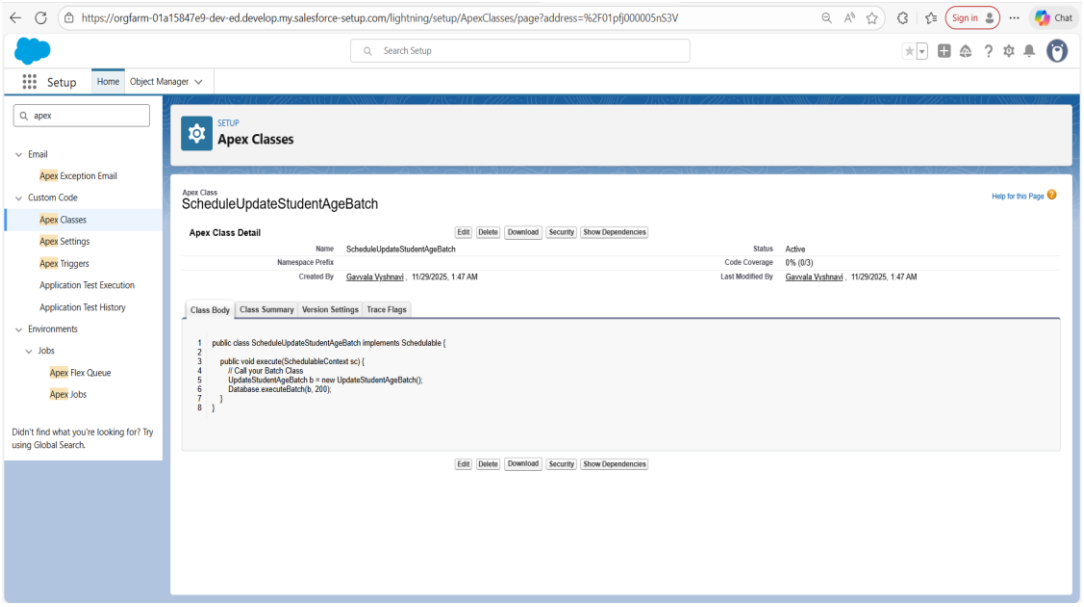
8. Queueable Apex

- Developed **UpdateStudentAgeQueueable** class implementing Queueable.
- Allows background execution of student age updates.
- Can be chained for sequential execution.



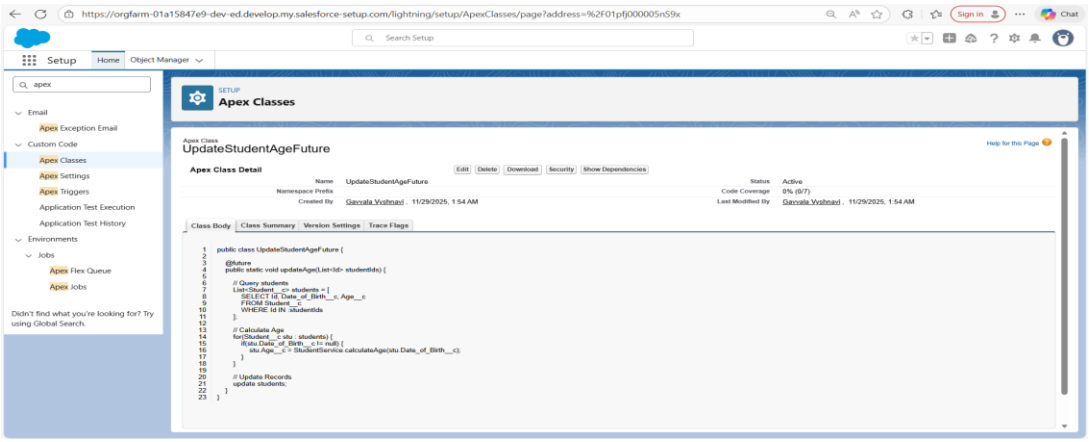
9. Scheduled Apex

- Created **ScheduleUpdateStudentAgeBatch** class implementing Schedulable.
- Scheduled batch jobs to run at predefined times.



10. Future Methods

- Implemented **UpdateStudentAgeFuture** class with @future annotation.
- Runs background updates asynchronously for lightweight processing.

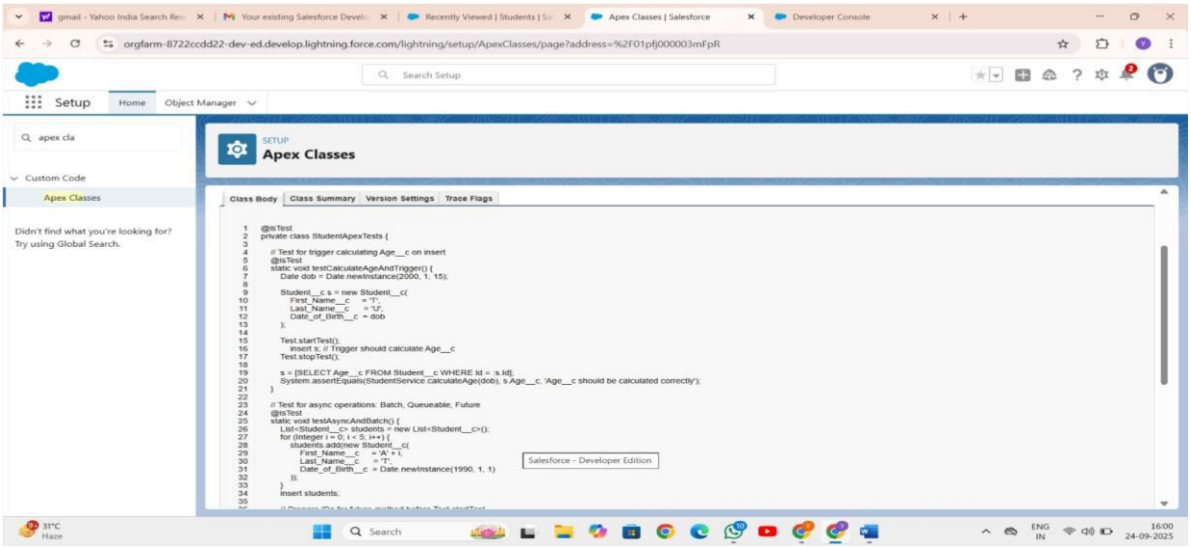


11 .ExceptionHandling

- Wrapped logic in **try-catch** blocks.
- Handled null Date of Birth scenario to prevent runtime errors.
- Ensured no unhandled exceptions crash the system.

12. Test Classes

- Created **TestStudentService** and other test classes with @isTest.
- Verified Batch, Queueable, Future, and Exception Handling methods.
- Achieved **code coverage** and validated system reliability.



13. Asynchronous Processing

- Implemented and tested all four types: **Batch, Queueable, Scheduled, and Future Methods**.
- Verified job execution via **Apex Jobs** in Setup.
- Ensured scalability for handling large volumes of Student records.