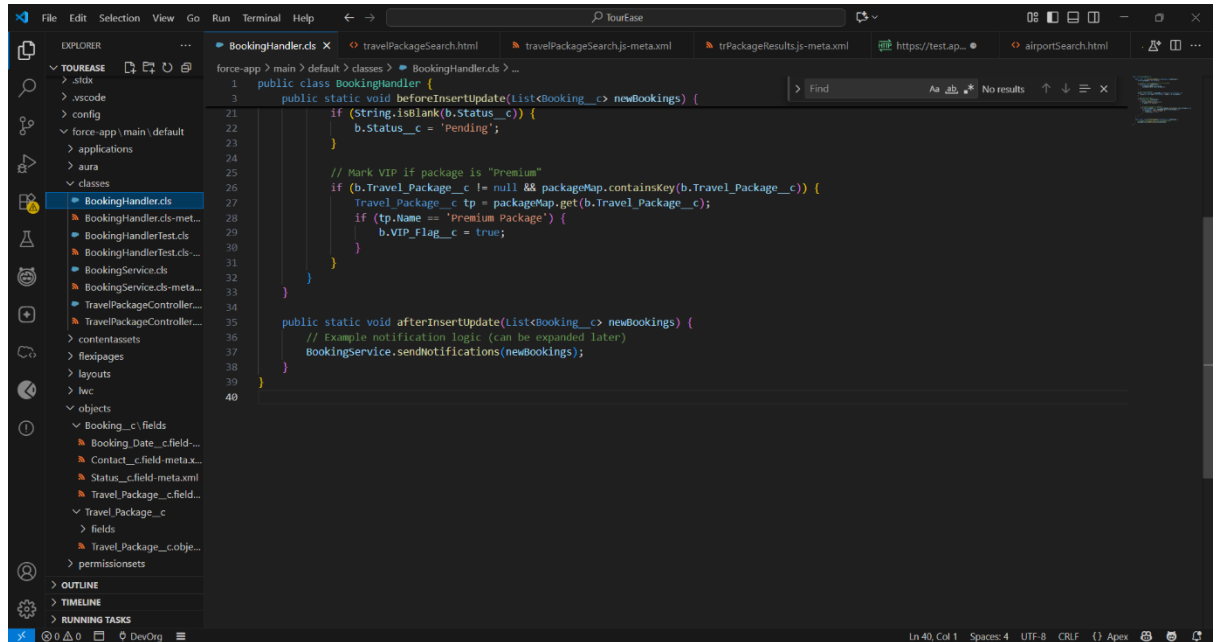


## Phase 5: Apex Programming (Developer) – TourEase Project

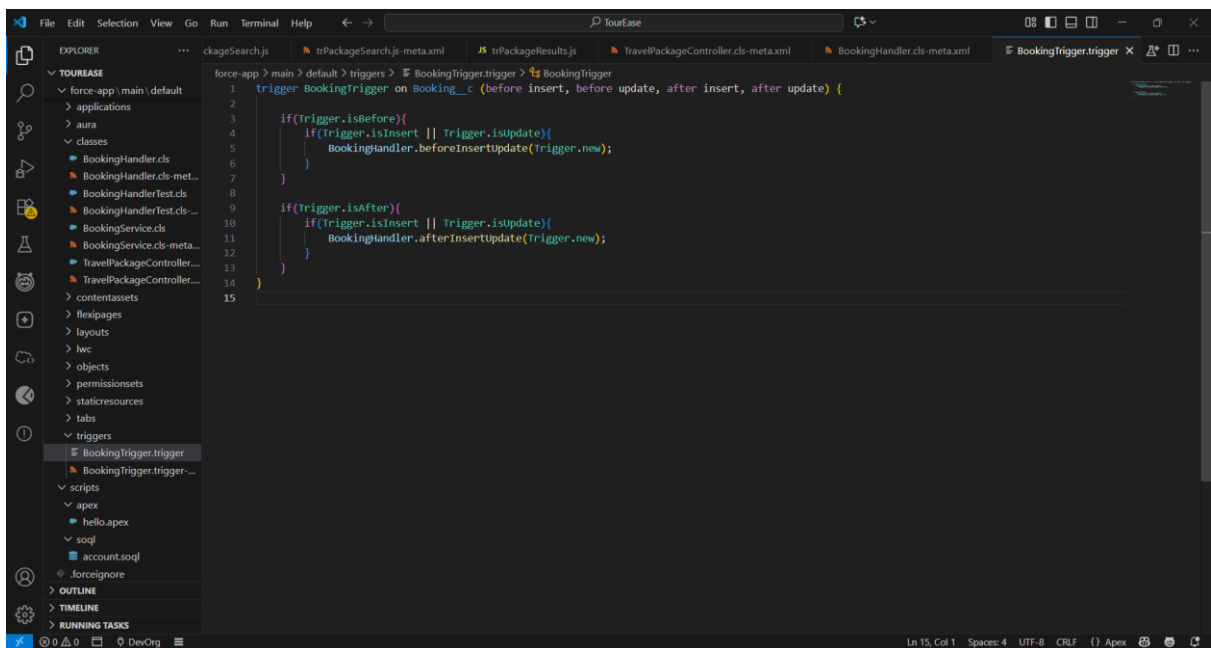
### 1. Classes & Objects

- **Purpose:** Encapsulate logic and data; create reusable methods for your project.



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

- **Purpose:** Execute logic automatically when records change.



### 3. Trigger Design Pattern

- **Purpose:** Organize triggers to separate business logic (trigger handler pattern).
- **Example:**

```
trigger BookingTrigger on Booking__c (before insert, before update) {  
    BookingTriggerHandler.handleBeforeInsertUpdate(Trigger.new);  
}
```

```
public class BookingTriggerHandler {  
    public static void handleBeforeInsertUpdate(List<Booking__c> bookings){  
        for(Booking__c b : bookings){  
            if(b.Status__c == null) b.Status__c = 'Pending';  
        }  
    }  
}
```

---

### 4. SOQL & SOSL

- **Purpose:** Query Salesforce data (SOQL) or search records (SOSL).
- 

### 5. Collections: List, Set, Map

- **Purpose:** Store multiple records efficiently.
- **Example:**

```
List<Booking__c> bookings = [SELECT Id, Name FROM Booking__c];  
Set<Id> bookingIds = new Set<Id>();  
Map<Id, Booking__c> bookingMap = new Map<Id, Booking__c>(bookings);
```

---

### 6. Control Statements

- **Purpose:** Apply logic using if, for, while, switch.
- **Example:**

```
for(Booking__c b : bookings){  
    if(b.Status__c == 'Pending'){  
        b.Status__c = 'Confirmed';  
    }  
}
```

```
}
```

---

## 7. Batch Apex

- **Purpose:** Process large datasets asynchronously.
- **Example:**

```
global class BookingBatch implements Database.Batchable<sObject> {  
    global Database.QueryLocator start(Database.BatchableContext BC){  
        return Database.getQueryLocator([SELECT Id, Status__c FROM Booking__c WHERE  
        Status__c='Pending']);  
    }  
    global void execute(Database.BatchableContext BC, List<Booking__c> scope){  
        for(Booking__c b : scope) b.Status__c = 'Confirmed';  
        update scope;  
    }  
    global void finish(Database.BatchableContext BC){}  
}
```

---

## 8. Queueable Apex

- **Purpose:** Run asynchronous jobs with complex logic.
- **Example:**

```
public class BookingQueueable implements Queueable {  
    public void execute(QueueableContext context){  
        List<Booking__c> bookings = [SELECT Id, Status__c FROM Booking__c WHERE  
        Status__c='Pending'];  
        for(Booking__c b : bookings) b.Status__c = 'Confirmed';  
        update bookings;  
    }  
}
```

The screenshot shows the Salesforce Developer Console with the following Apex code in the editor:

```

21 bookings.add(new Booking__c(
22     Travel_Package__c = pkg.Id,
23     Number_of_Seats__c = 3,
24     Booking_Date__c = Date.today(),
25     Status__c = 'Confirmed' // Required for validation
26 ));
27 insert bookings;
28
29 // Run batch
30 Test.startTest();
31 BookingBatch batch = new BookingBatch();
32 Database.executeBatch(batch, 100);
33 Test.stopTest();
34
35 // Verify Booked_Seats__c
36 pkg = [SELECT Booked_Seats__c FROM Travel_Package__c WHERE Id = :pkg.Id];
37 System.assertEquals(8, pkg.Booked_Seats__c, 'Booked seats should match total bookings');
38 }
39
40

```

Below the code editor is a table with the following columns: User, Application, Operation, Time, Status, Read, and Size. The table contains 6 rows of log data.

User	Application	Operation	Time	Status	Read	Size
Chidwala Testis	Unknown	ApexTestHandler	9/26/2025, 10:06:14 PM	Success	Unread	30.46 KB
Chidwala Testis	Unknown	ApexTestHandler	9/26/2025, 10:06:11 PM	Success	Unread	2.18 KB
Chidwala Testis	Unknown	ApexTestHandler	9/26/2025, 10:06:08 PM	Success	Unread	12.05 KB
Chidwala Testis	Unknown	ApexTestHandler	9/26/2025, 10:06:08 PM	Success	Unread	95.07 KB
Chidwala Testis	Unknown	ApexTestHandler	9/26/2025, 10:06:05 PM	Success	Unread	6 KB
Chidwala Testis	Unknown	ApexTestHandler	9/26/2025, 10:06:05 PM	Success	Unread	11.08 KB

## 9. Scheduled Apex

- **Purpose:** Run Apex classes at a scheduled time.
- **Example:**

```

public class BookingScheduler implements Schedulable {
    public void execute(SchedulableContext sc){
        BookingQueueable job = new BookingQueueable();
        System.enqueueJob(job);
    }
}

```

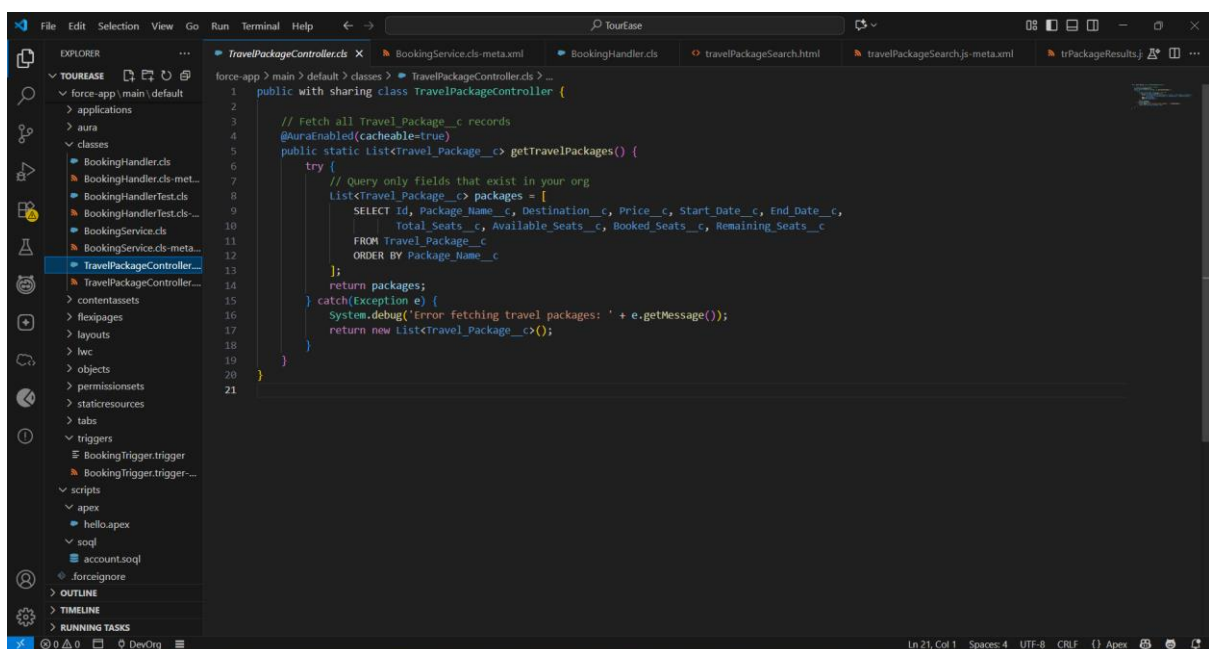
The screenshot shows the Salesforce Setup page with the 'Scheduled Jobs' section selected. The page displays a table of all scheduled jobs with the following columns: Action, Job Name, Submitted By, Submitted, Started, Next Scheduled Run, Type, and Cron Trigger ID.

Action	Job Name	Submitted By	Submitted	Started	Next Scheduled Run	Type	Cron Trigger ID
Del	CommIncrementalSitemapJob-00C000000guf-0CM000000gmU	Tedla Chidwala	9/23/2025, 12:09 PM	9/26/2025, 5:45 PM	9/28/2025, 5:45 PM	Sitemap SEO Incremental Job	00e000002gCo
Del	CommSitemapJob-00C000000guf-0CM000000gmU	Tedla Chidwala	9/23/2025, 12:09 PM	9/27/2025, 7:19 PM	9/27/2025, 7:19 PM	Sitemap SEO Generation Job	00e000002gCn
Del	Metalytics Data Loader Job for Org - 00D000000guf	User: Automation	9/18/2025, 9:52 PM	9/27/2025, 8:57 AM	9/28/2025, 8:57 AM	Autonomous Data Loader Job	00e000003Tku
	Program Milestone Computation Cron Job	Process, Automated	9/18/2025, 9:52 PM	9/27/2025, 9:59 AM	9/27/2025, 11:59 AM	Program Milestone Computation Cron Job	00e000003Tku
	Program Status Update Cron Job	Process, Automated	9/18/2025, 9:52 PM	9/27/2025, 9:59 AM	9/27/2025, 9:59 PM	Program Status Update Cron Job	00e000003Tku

## 10. Exception Handling

- **Purpose:** Catch and manage runtime errors.
- **Example:**

```
try {  
    update bookings;  
} catch(DmlException e){  
    System.debug('Error updating bookings: ' + e.getMessage());  
}
```



## 11. Test Classes

- **Purpose:** Ensure code works and deploy to production.
- **Example:**

@isTest

```
public class BookingControllerTest {  
    @isTest static void testConfirmBooking(){  
        Booking__c b = new Booking__c(Name='Test Booking', Status__c='Pending');  
        insert b;  
        BookingController.confirmBooking(b.Id);  
        b = [SELECT Status__c FROM Booking__c WHERE Id=:b.Id];  
        System.assertEquals('Confirmed', b.Status__c);  
    }  
}
```

```
}  
  
}
```

The screenshot displays the Salesforce Developer Console interface. The top section shows the Apex code for a test class, `BookingBatchTest.apex`. The code includes a `test` method that creates a `BookingBatch` object, inserts it into the database, and then verifies the results using `System.assertEquals`.

```
21 bookings.add(new Booking__c(  
22     Travel_Package__c = pkg.Id,  
23     Number_of_Seats__c = 3,  
24     Booking_Date__c = Date.today(),  
25     Status__c = 'Confirmed' // Required for validation  
26 ));  
27 insert bookings;  
28  
29 // Run batch  
30 Test.startTest();  
31 BookingBatch batch = new BookingBatch();  
32 Database.executeBatch(batch, 100);  
33 Test.stopTest();  
34  
35 // Verify Booked_Seats__c  
36 pkg = [SELECT Booked_Seats__c FROM Travel_Package__c WHERE Id = :pkg.Id];  
37 System.assertEquals(8, pkg.Booked_Seats__c, 'Booked seats should match total bookings');  
38 }  
39 }  
40 }
```

The bottom section of the console shows a table of logs. The table has columns for User, Application, Operation, Time, Status, Read, and Size. The logs show that the test class was executed successfully.

User	Application	Operation	Time	Status	Read	Size
Chadila Tedia	Unknown	ApexTestHandler	9/26/2025, 10:06:14 PM	Success	Unread	30.46 KB
Chadila Tedia	Unknown	ApexTestHandler	9/26/2025, 10:06:11 PM	Success	Unread	2.18 KB
Chadila Tedia	Unknown	ApexTestHandler	9/26/2025, 10:06:08 PM	Success	Unread	12.05 KB
Chadila Tedia	Unknown	ApexTestHandler	9/26/2025, 10:06:08 PM	Success	Unread	95.07 KB
Chadila Tedia	Unknown	ApexTestHandler	9/26/2025, 10:06:05 PM	Success	Unread	6 KB
Chadila Tedia	Unknown	ApexTestHandler	9/26/2025, 10:06:05 PM	Success	Unread	11.08 KB

## Conclusion:

In Phase 5, the TourEase project utilized Apex programming to automate and manage core booking processes, including notifications and data handling. By implementing classes, triggers, SOQL/SOSL queries, collections, and control logic, along with asynchronous methods like Batch, Queueable, and Scheduled Apex, the system efficiently processes data while ensuring reliability through exception handling and test classes. This phase highlights how structured Apex code enables scalable, maintainable, and automated operations in a real-world Salesforce application.