**Salesforce Project Implementation Phases**

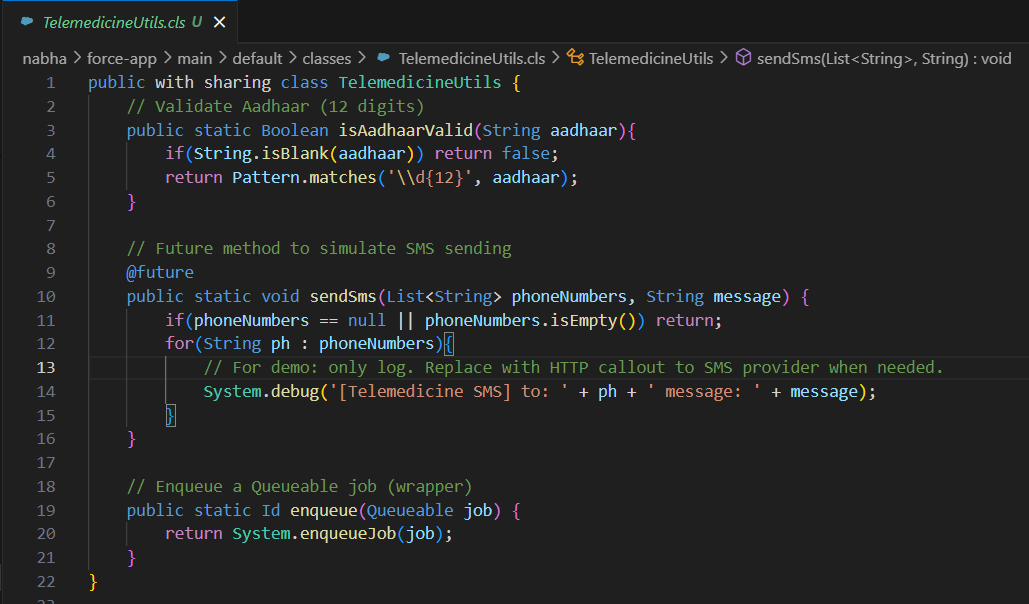
**Phase 5: Apex Programming (Developer)**

In this phase, I have extend Salesforce point-and-click features with **Apex programming** to implement advanced business logic for our Telemedicine CRM. Since healthcare use-cases often require automation, bulk processing, and integrations, Apex gives us flexibility and scalability.

* **Classes & Objects**

I have created “Apex classes” to encapsulate business logic.

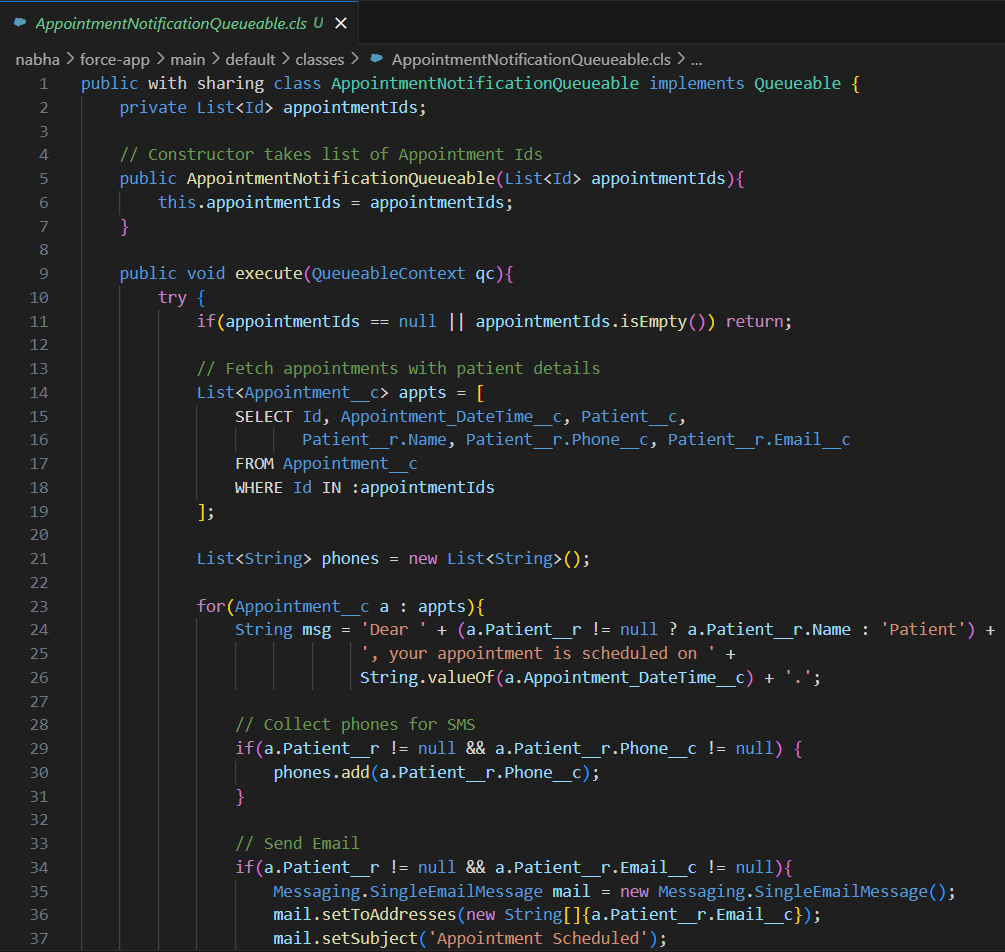
- **TelemedicineService**



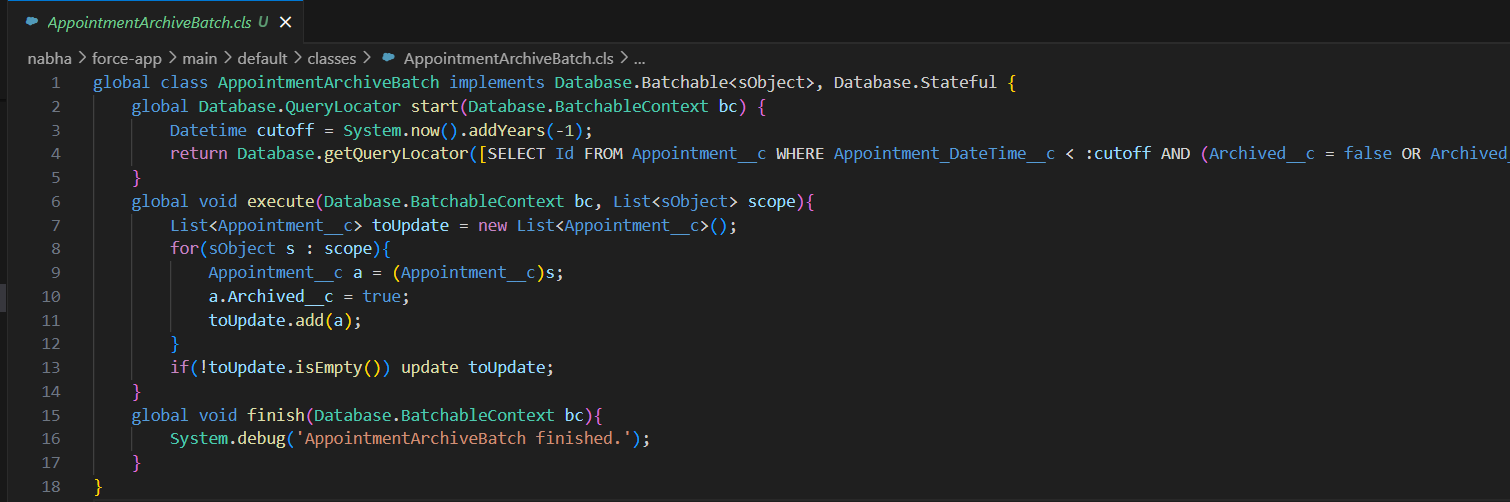
- **PurchaseOrderService**



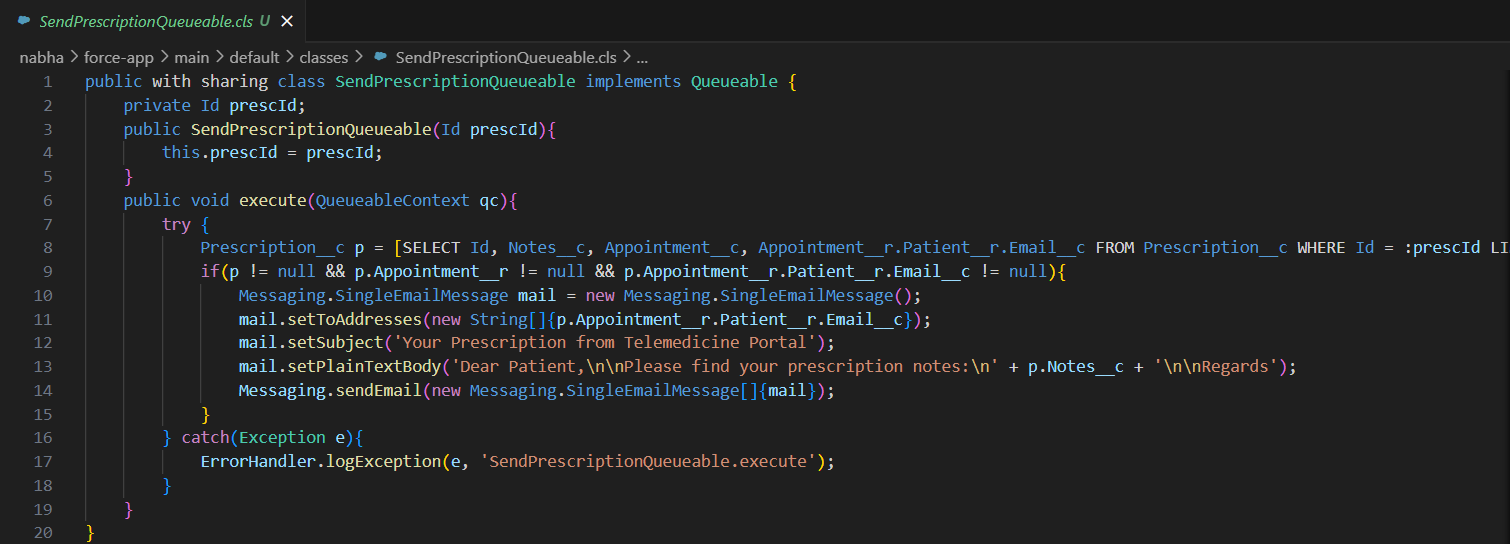
- **AppointmentNotificationQueueable**



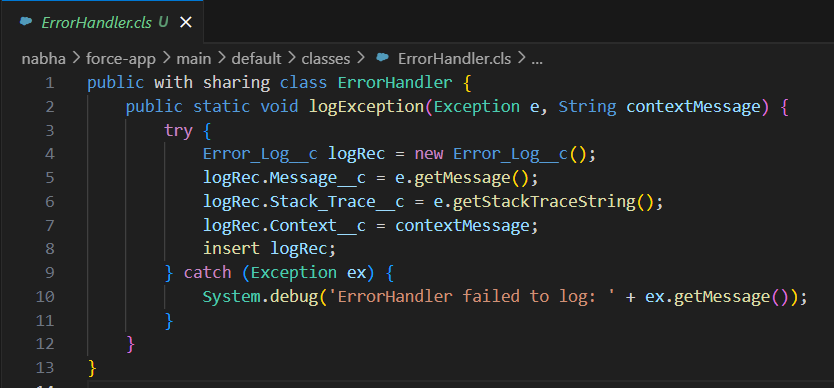
- **AppointmentArchiveBatch**



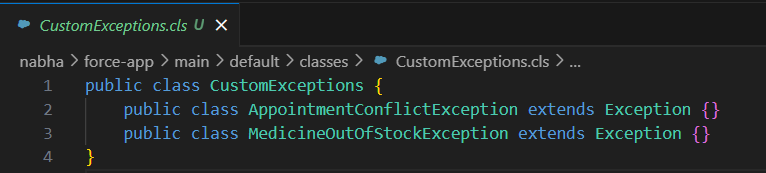
- **SendPrescriptionQueueable**



- **ErrorHandler**



- **CustomExceptions**

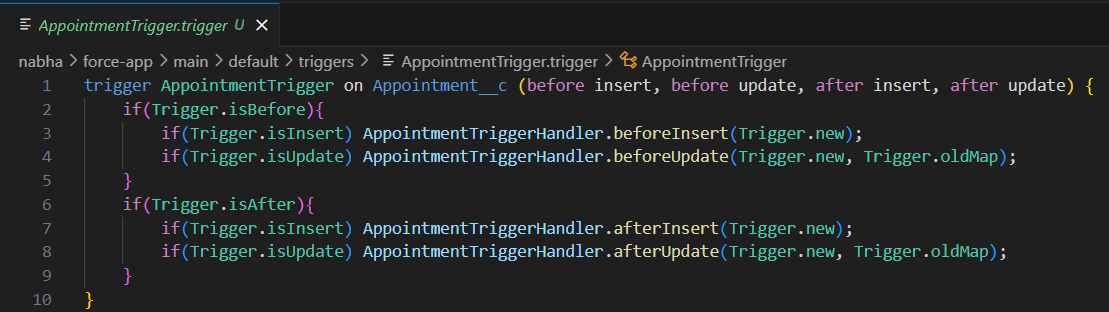


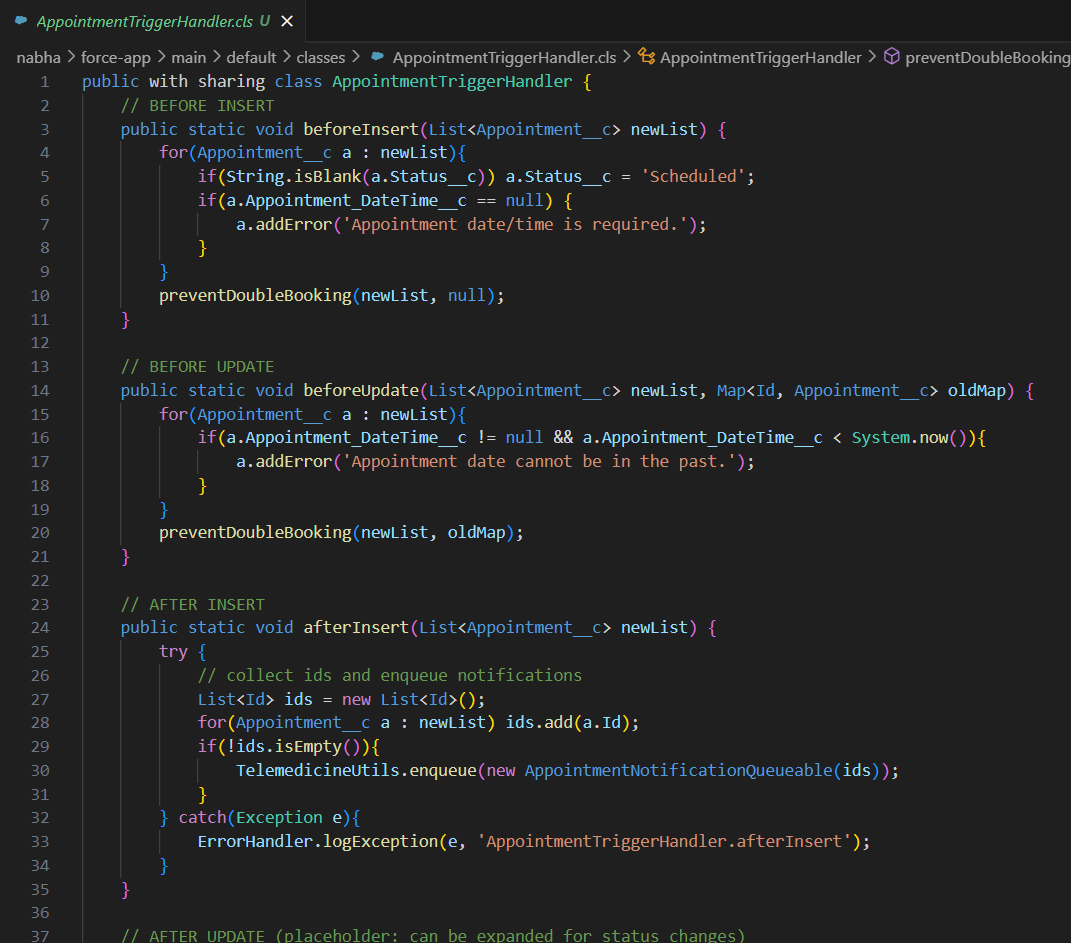
* **Apex Triggers**

Triggers automate logic that cannot be handled via declarative tools.

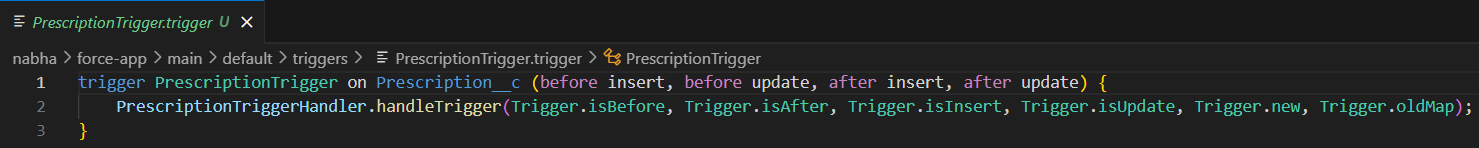
We created:

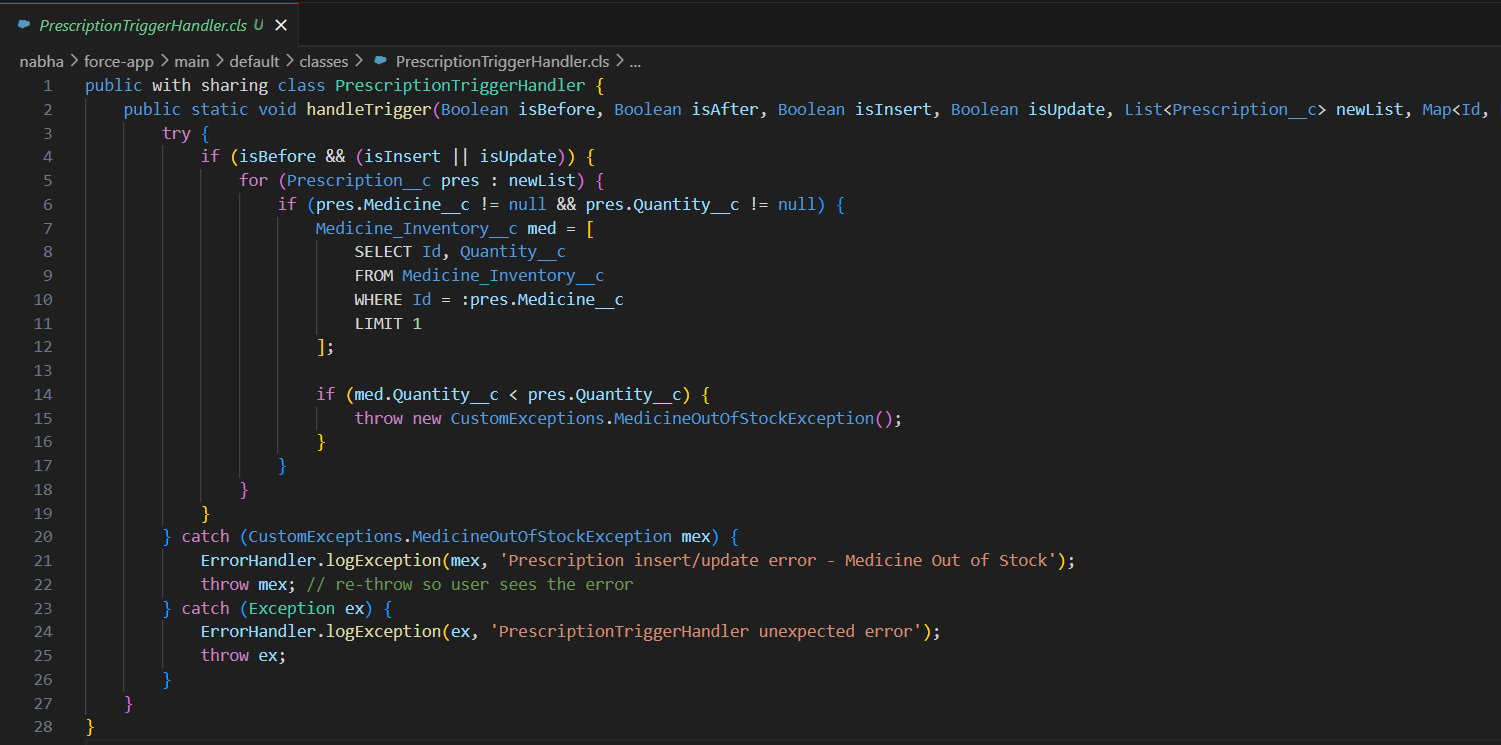
- AppointmentTrigger (delegates to AppointmentTriggerHandler)





- PrescriptionTrigger (delegates to PrescriptionTriggerHandler)





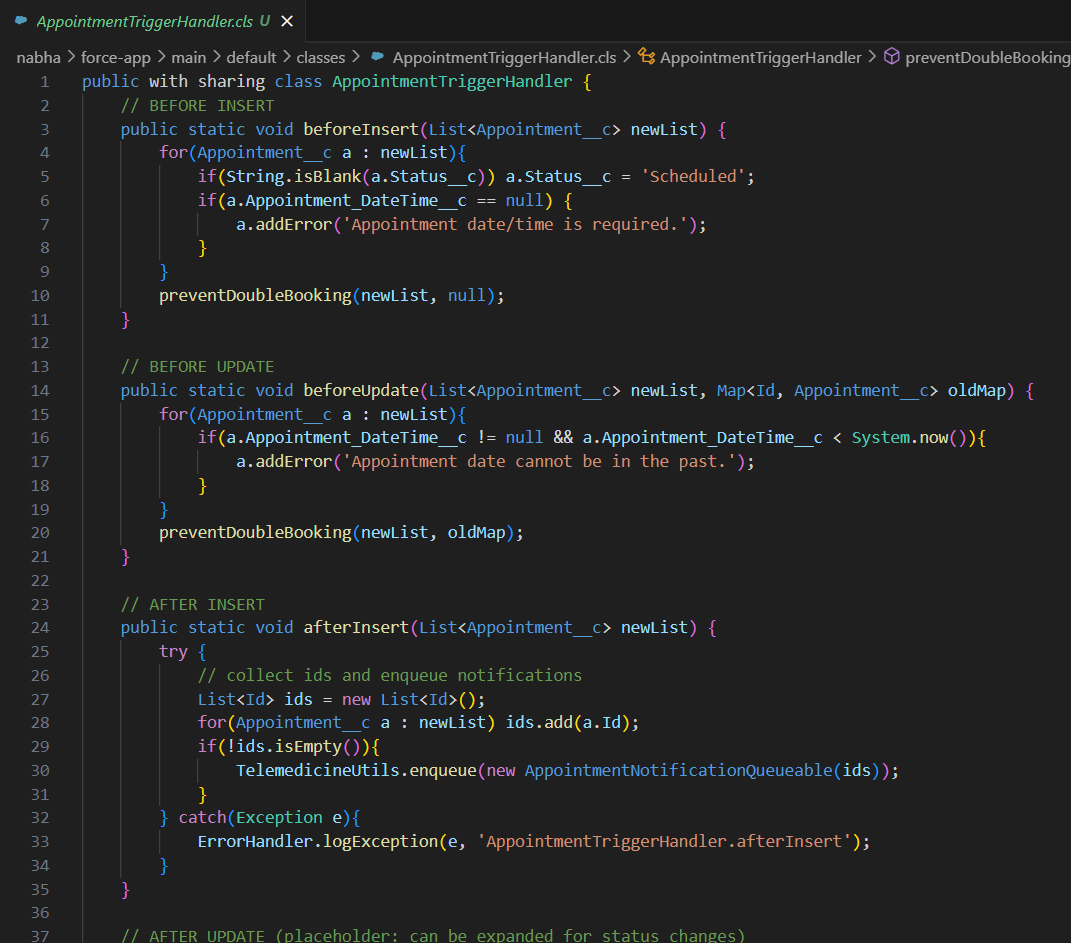
* **Trigger Design Pattern**

Implemented “Trigger Handler Pattern” to avoid hardcoding logic directly inside triggers.

Trigger: **AppointmentTrigger**

- Handler Class: AppointmentTriggerHandler

- Methods: beforeInsert, beforeUpdate, afterInsert, afterUpdate.



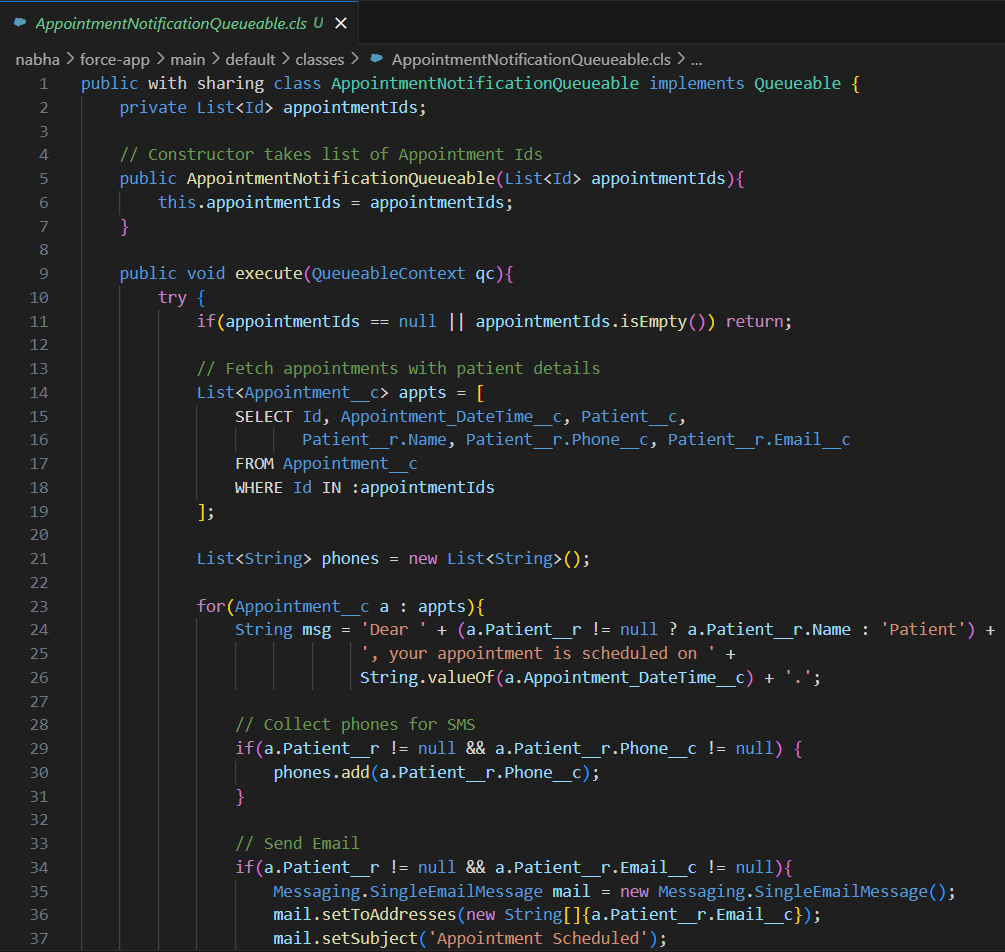
* **SOQL & SOSL**

- “SOQL” used for retrieving patient and appointment details.

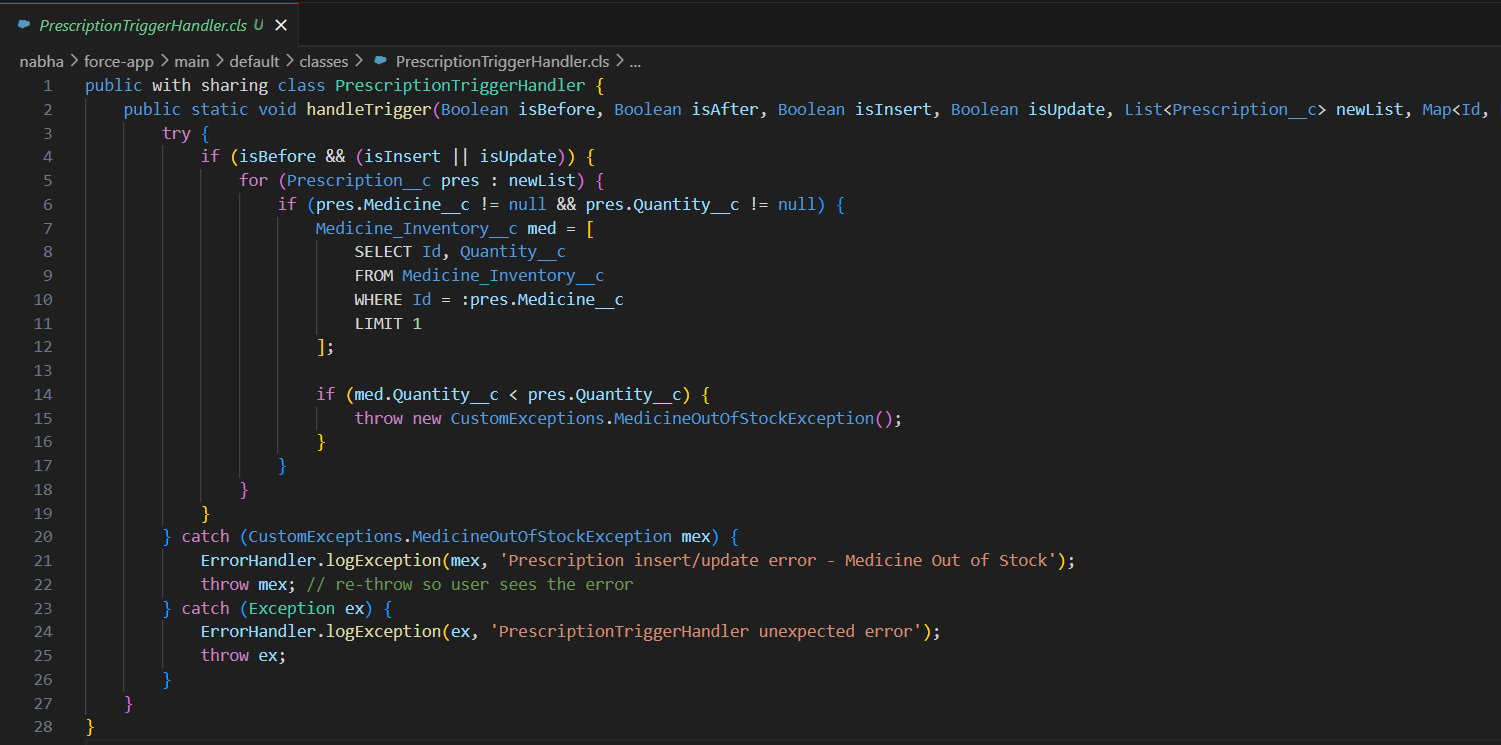
- “SOSL” implemented for global patient search by Aadhaar, phone, or email.

Already used in:

- AppointmentNotificationQueueable (queries Patient & Contact)



- PrescriptionTriggerHandler (queries Medicine\_Inventory\_\_c)



* **Collections**

- List : To store multiple appointments in bulk insert.

- Set : To ensure no duplicate patient IDs when scheduling.

- Map : DoctorId

→ List of Appointments (used for bulk assignment logic).

It is present in Trigger + batch class.

* **Control Statements**

- IF-ELSE for appointment conflict detection.

- FOR loops for iterating through bulk patient records.

- SWITCH statements for handling medicine stock statuses.

Used in if/else checks inside handlers + services

* **Batch Apex**

Used for “large-volume data processing”.

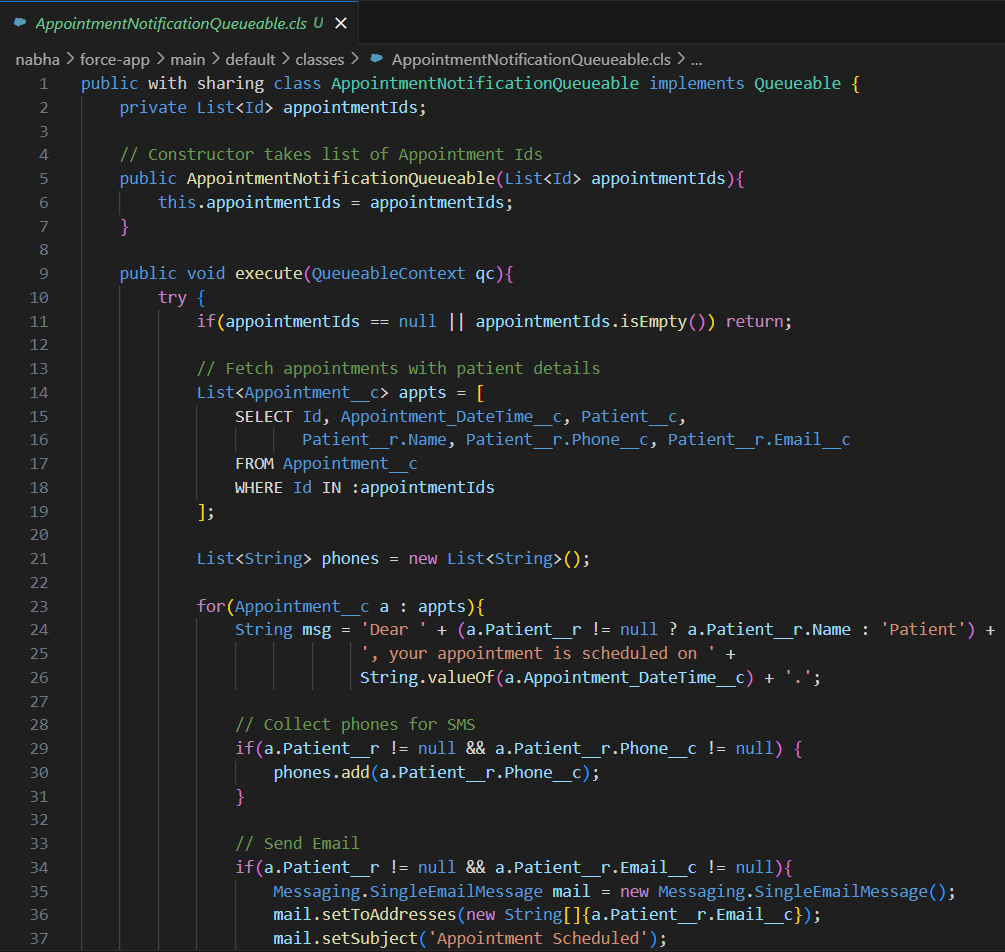
* AppointmentArchiveBatch class created.



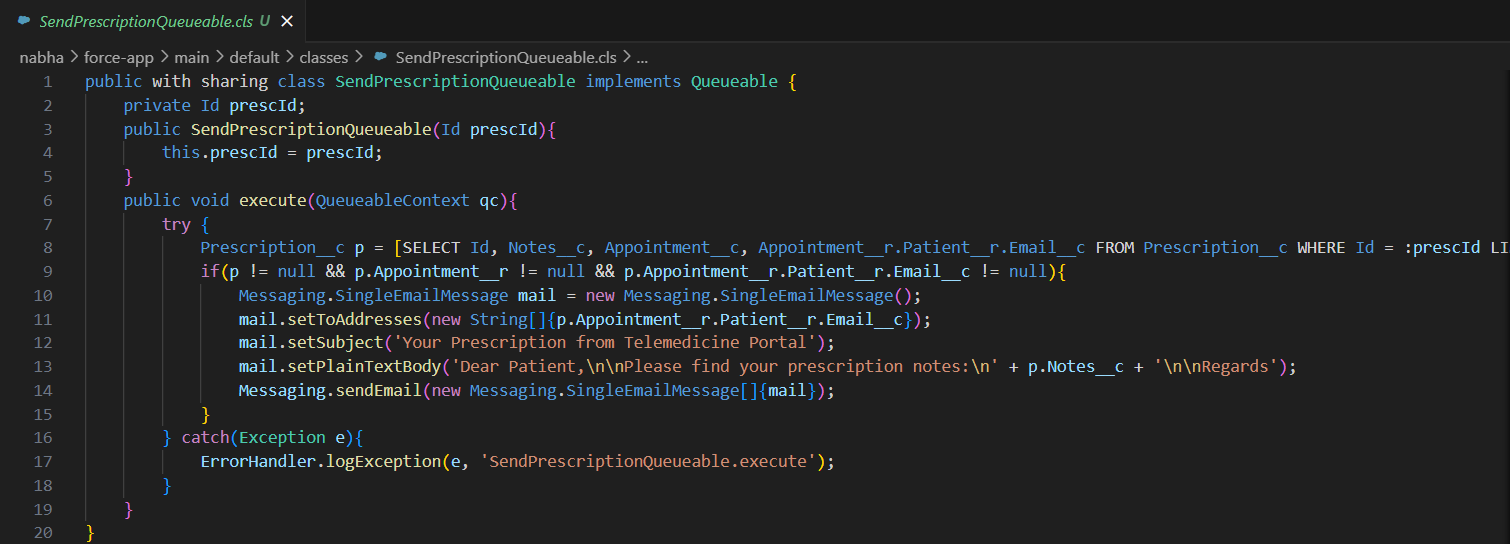
* **Queueable Apex**

Used for “chained async operations”.

- AppointmentNotificationQueueable



- SendPrescriptionQueueable



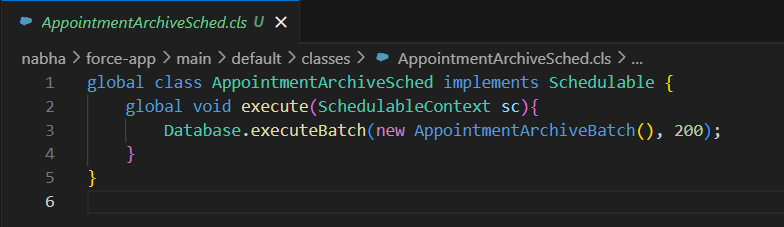
* **Scheduled Apex**

Nightly jobs to sync external pharmacy data (via Salesforce Connect).

Example: **PharmacySyncScheduler**

→ Runs every night at 2 AM.

Created: **AppointmentArchiveSched** for the same.



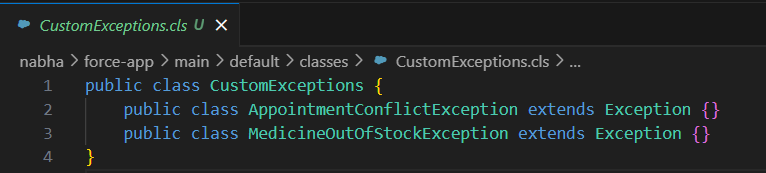
* **Future Methods**

- Used for lightweight async tasks.

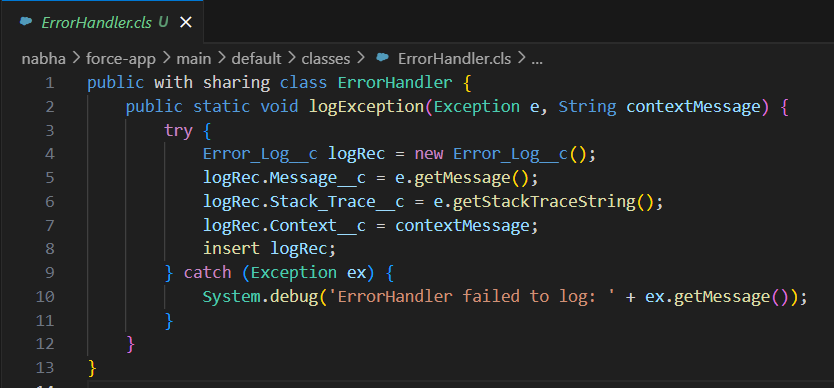
Example: @future method for sending SMS confirmation to patients after booking.

* **Exception Handling**

- Custom exception classes created like **AppointmentConflictException** and **MedicineOutOfStockException**.



- All Apex code wrapped with try…catch blocks and errors logged to a custom object **Error\_Log\_\_c**.



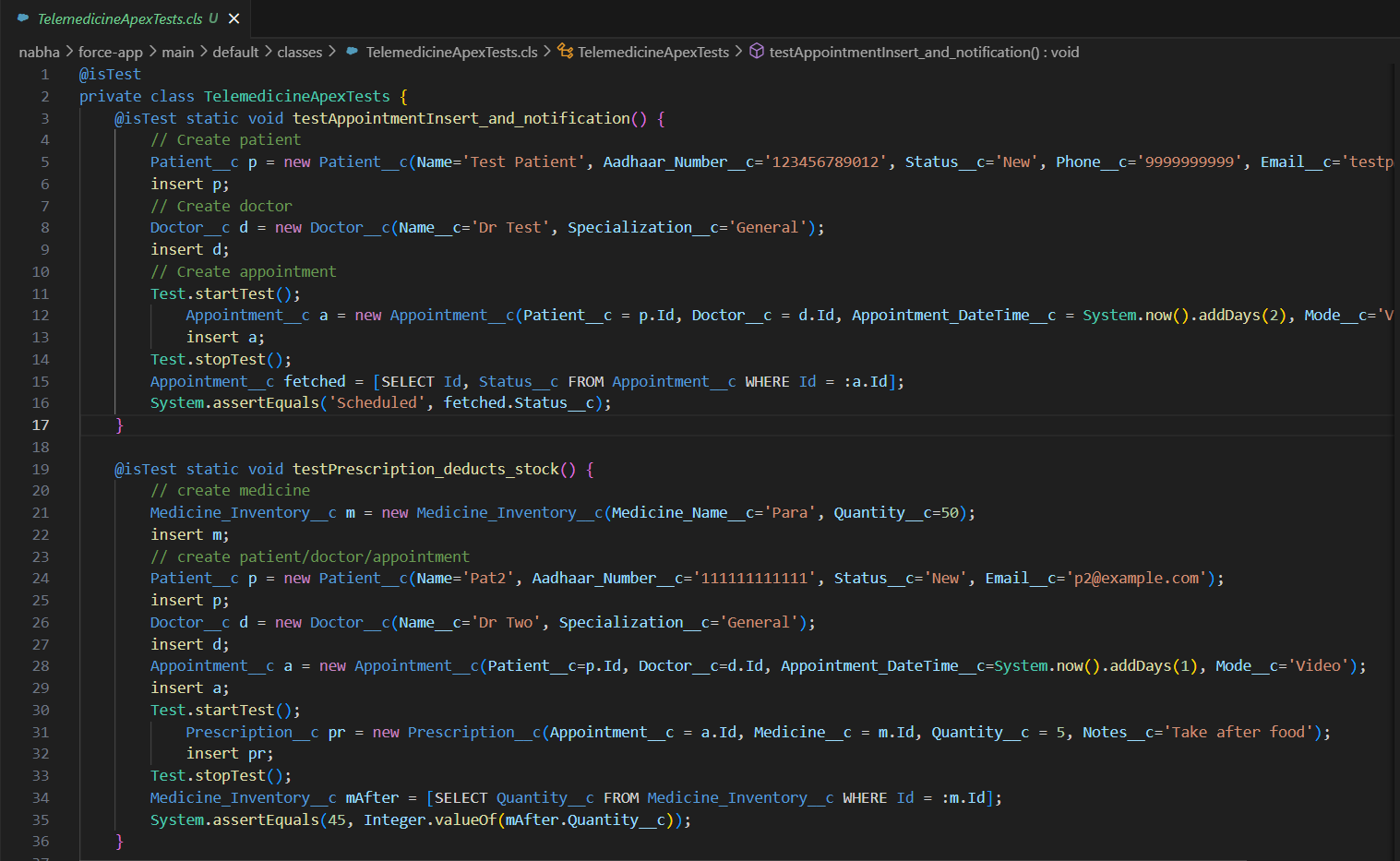
* **Test Classes**

**TelemedicineApexTests** ensures:

- Double-booking is prevented.

- Notifications are sent after booking.

- Medicine inventory updates correctly.



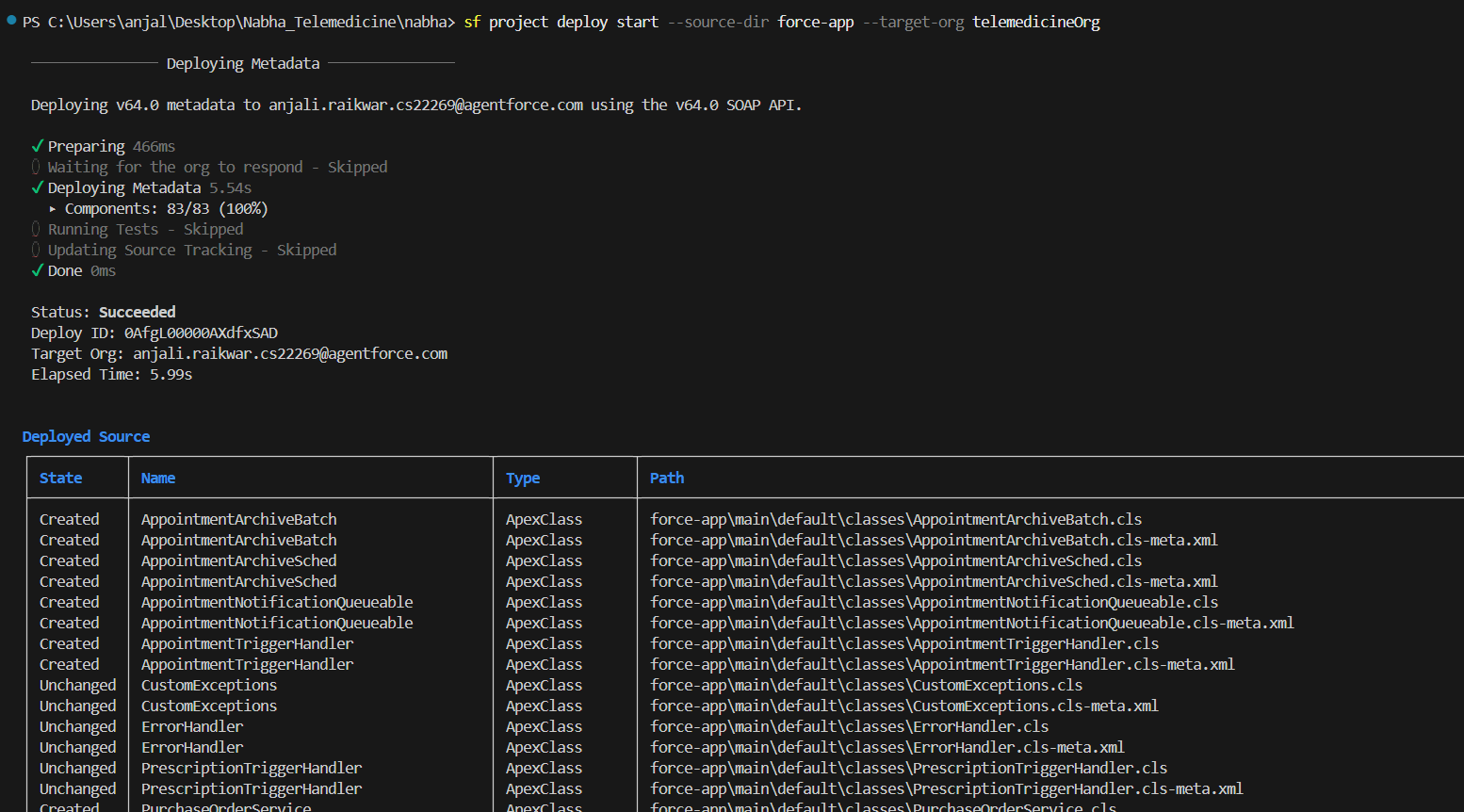
* **Asynchronous Processing**

- Combined **“Batch + Queueable + Scheduled”** for handling bulk jobs.

- This ensures the system works smoothly for thousands of records, supporting scalability for other rural regions.

**Outcome of Phase 5**

With Phase 5 complete, our project moves beyond point-and-click automation into **“scalable enterprise-level programming”**.



Have successfully Deployed all the Apex Classes !!