Phase 5 - Apex Programming

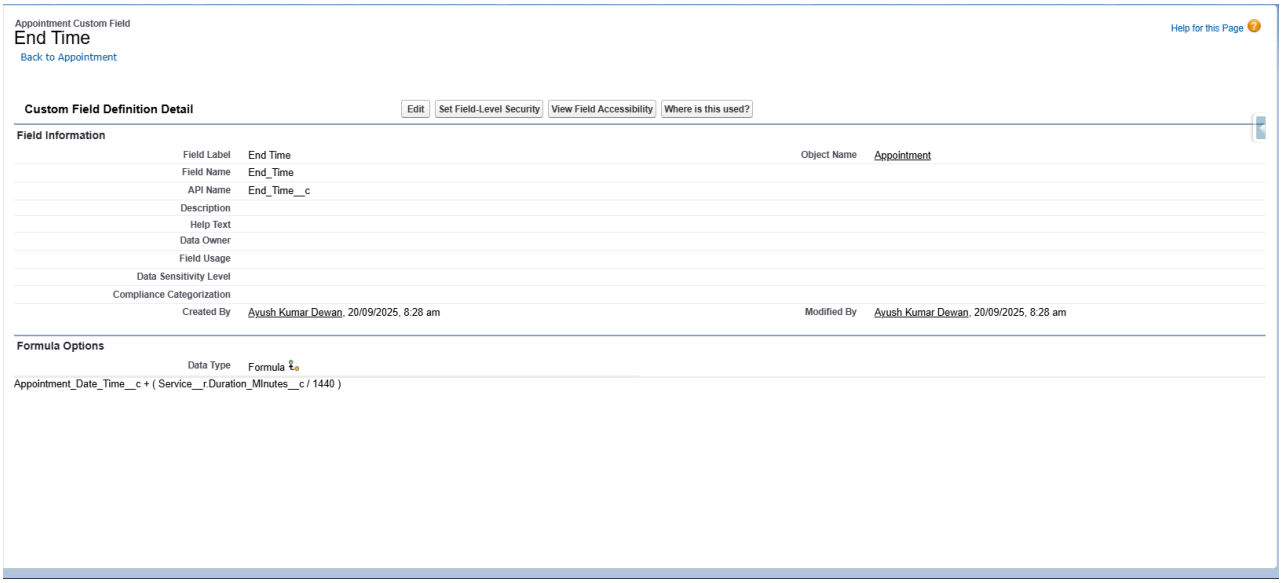
This document provides a detailed, step-by-step summary of all development tasks completed for Phase 5 of the Salon Management System project. The objective of this phase was to use Apex code to solve a complex business requirement that could not be handled by declarative automation alone.

1. Business Requirement

- **Purpose:** The core objective of this phase was to prevent the double-booking of stylists. The system needed a robust, server-side rule to make it impossible for a user to create an appointment for a stylist if that appointment's time slot overlapped with an existing appointment for the same stylist.

2. Prerequisite: Data Model Enhancement

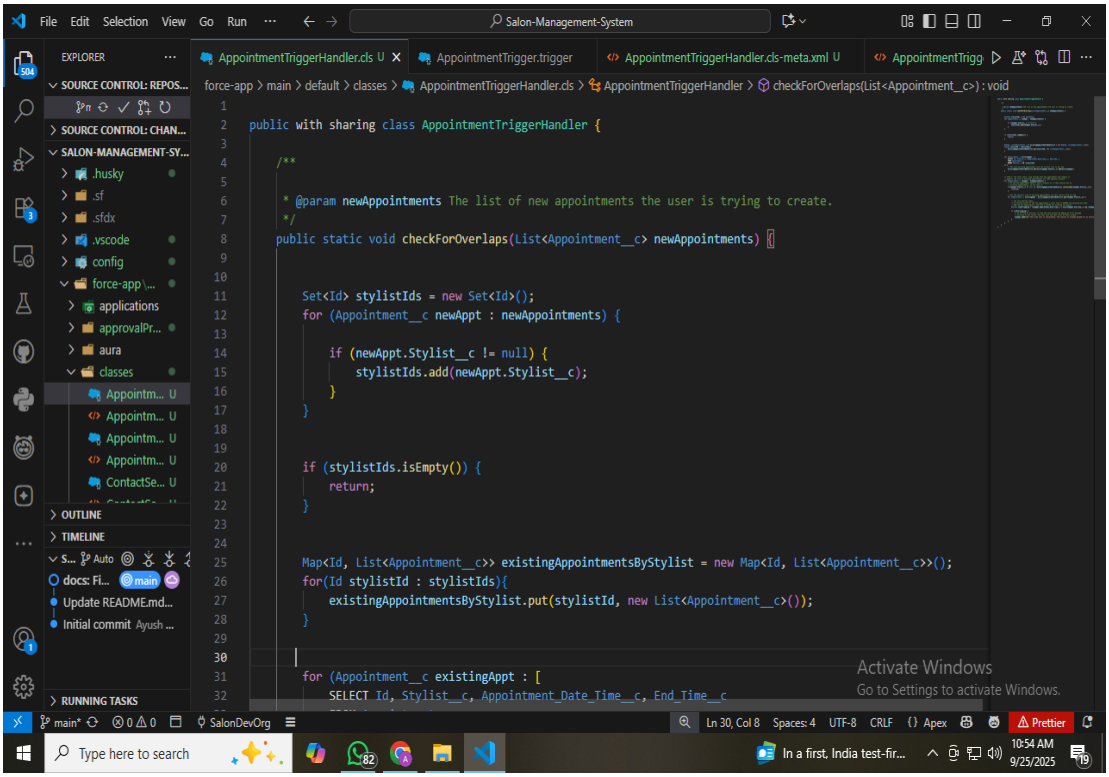
- **Purpose:** Before writing the Apex code, the data model needed to be prepared to support the time-conflict logic. The code needs to know both the start and end time of an appointment to check for overlaps.
- **Steps Followed:**
    1. A new **Formula Field** named `End Time` was created on the `Appointment` object via `Setup > Object Manager`.
    2. The formula was set to return a **Date/Time** value.
    3. The formula `Appointment_Date_Time__c + ( Service__r.Duration_Minutes__c / 1440 )` was entered to automatically calculate the end time by adding the service duration to the start time.



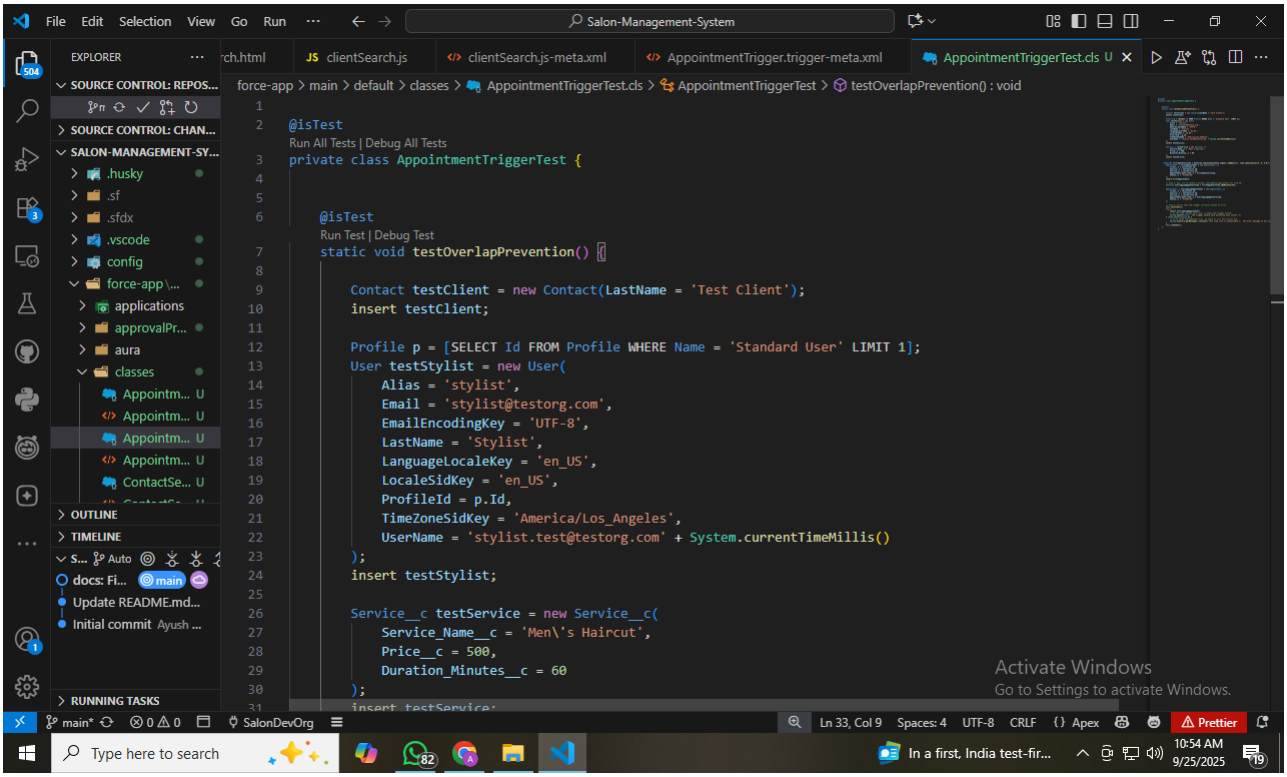
3. Apex Code Development

- **Purpose:** A professional, best-practice "Trigger Handler" pattern was used to build the solution. This separates the logic from the trigger, making the code cleaner, more reusable, and easier to test. All code was written locally in a VS Code project.

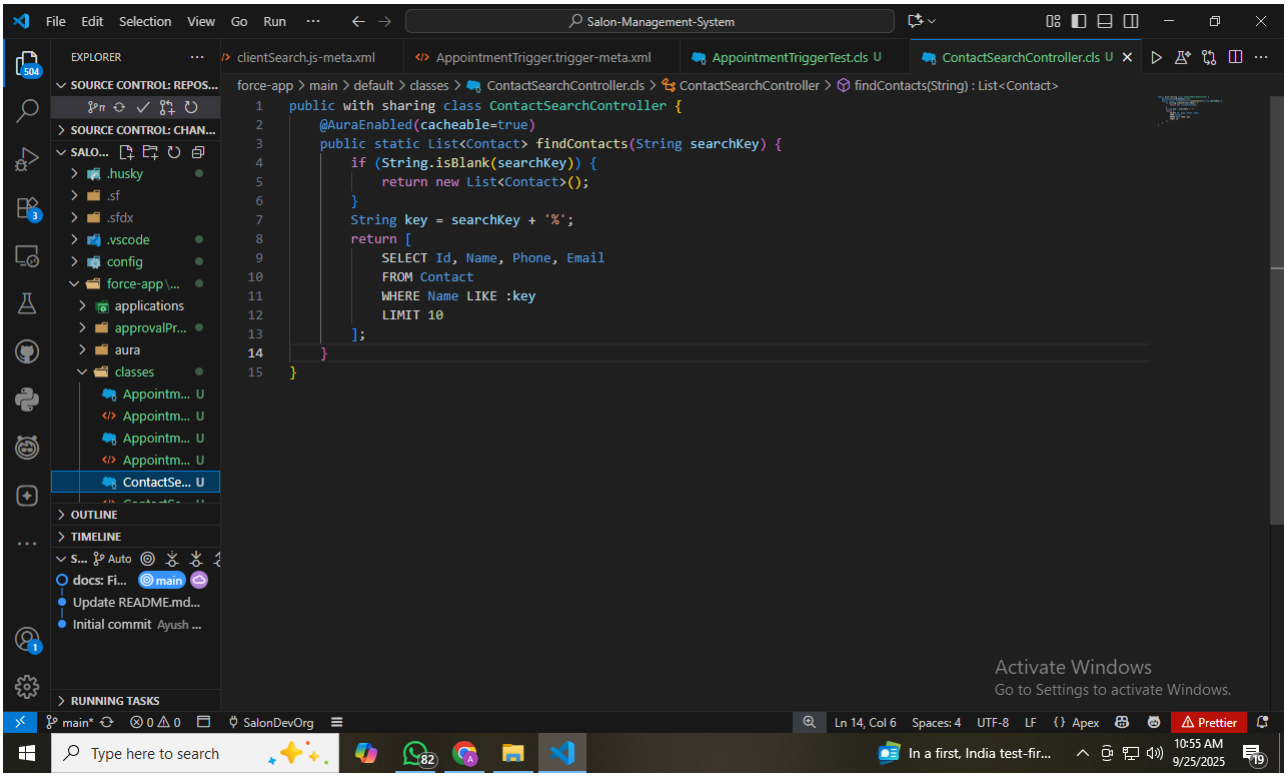
- **Steps Followed:**
    1. **Apex Handler Class Created (`AppointmentTriggerHandler.cls`):**
        - In VS Code, a new file was created in the `force-app/classes/` directory.
        - The `checkForOverlaps` method was built inside this class. This method contains all the business logic to gather stylist IDs, query existing appointments, loop through the records to find time conflicts, and use the `addError()` method to block the save if an overlap is found.

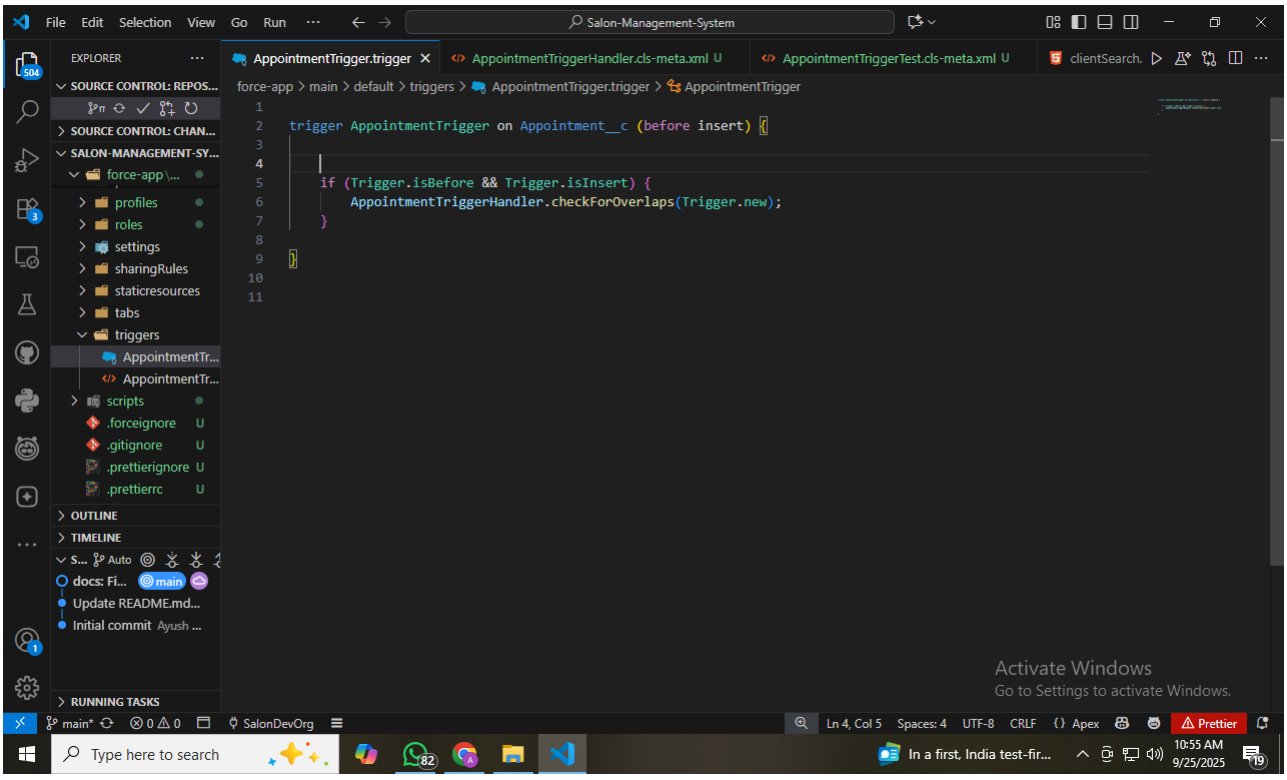


    2. **Apex Trigger Created (`AppointmentTrigger.trigger`):**
        - In VS Code, a new file was created in the `force-app/triggers/` directory.
        - The trigger was configured to fire on the **`before insert`** event of the `Appointment__c` object.
        - The trigger's only responsibility is to call the `checkForOverlaps` method in the handler class, passing it the list of new appointment records (`Trigger.new`).



3. **Apex Test Class Created (`AppointmentTriggerTest.cls`):**
   - In VS Code, a new file was created in the `force-app/classes/` directory to meet Salesforce's code coverage requirements.
   - The test method first creates all necessary test data (a client, a stylist, a service).
   - It then successfully inserts a valid, non-overlapping appointment.
   - Finally, it attempts to insert a second, overlapping appointment inside a `try/catch` block, asserting that the system correctly throws a `DmlException` containing the specific custom error message from the handler class.





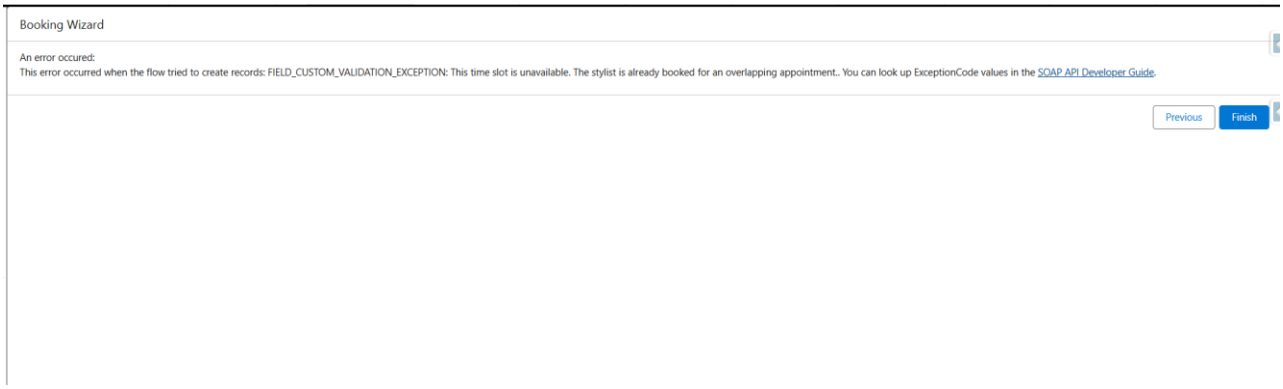
4. Deployment and Debugging

- **Purpose:** To move the locally written code to the Salesforce org and ensure it is working correctly.
- **Steps Followled:**

1. The new Apex files were deployed from VS Code to the Salesforce org using the **`SF: Deploy Source to Org`** command.
2. Initial deployment errors were successfully debugged, including fixing typos in custom field API names (e.g., `Service_c` to `Service__c`) and correcting the syntax for `Datetime` creation in the test class.

5. Flow Integration (Fault Path)

- **Purpose:** To create a seamless user experience by integrating the Apex error message directly into the `Booking Wizard` Screen Flow.
- **Steps Followed:**
  1. A new **"Error Screen"** was added to the `Booking Wizard` flow.
  2. This screen was configured with a `Display Text` component to show the `{!$Flow.FaultMessage}` global variable, which automatically captures the error message from the trigger.
  3. A **Fault Path** (a red, dotted line) was created, connecting the `Create Records` element to the new "Error Screen."
- **Result:** When the Apex trigger blocks a save, the flow no longer shows a generic error. Instead, it automatically navigates the user to the Error Screen, which displays the trigger's specific, user-friendly message (e.g., "This time slot is unavailable...").



6. End-to-End Testing

- **Purpose:** To fully test the feature from the perspective of the end-user.
- **Steps Followed:**
  1. Logged in as the **Receptionist** user.
  2. Used the `Booking Wizard` to successfully create a valid, non-overlapping appointment.
  3. Used the `Booking Wizard` again to attempt to create an overlapping appointment for the same stylist.
  4. Successfully verified that the trigger blocked the save and that the custom error message was displayed correctly on the flow's Error Screen.

7. Version Control

- **Purpose:** To save a permanent record of the new code to the project's central repository.
- **Steps Followed:** After successful deployment and testing, all new Apex code files were saved to the project's GitHub repository using `git add`, `git commit`, and `git push`.