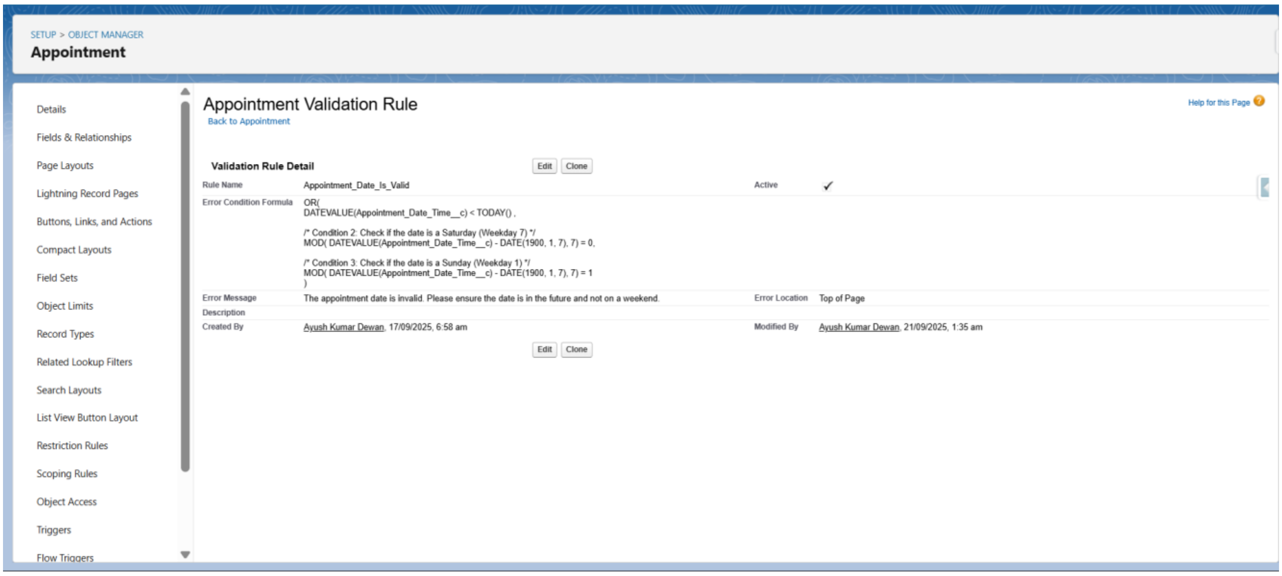
Phase 4 - Process Automation

This document provides a detailed, step-by-step summary of all automation tasks completed for Phase 4 of the Salon Management System project. The objective of this phase was to build the "brains" of the application to automate key business processes, enforce rules, and guide users.

1. Validation Rules

- **Purpose:** To ensure data quality and prevent user error by blocking invalid data *before* a record is saved. This is the first line of defense for maintaining a clean and reliable schedule.
- **Steps Followed:**
    1. A new validation rule named `Appointment_Date_Is_Valid` was created on the `Appointment` object via `Setup > Object Manager`.
    2. A time-zone safe formula was written to check for three invalid conditions: if the appointment date is in the past, if it is a Saturday, or if it is a Sunday.
    3. A user-friendly error message was configured to display at the top of the page, clearly explaining the booking error to the user.



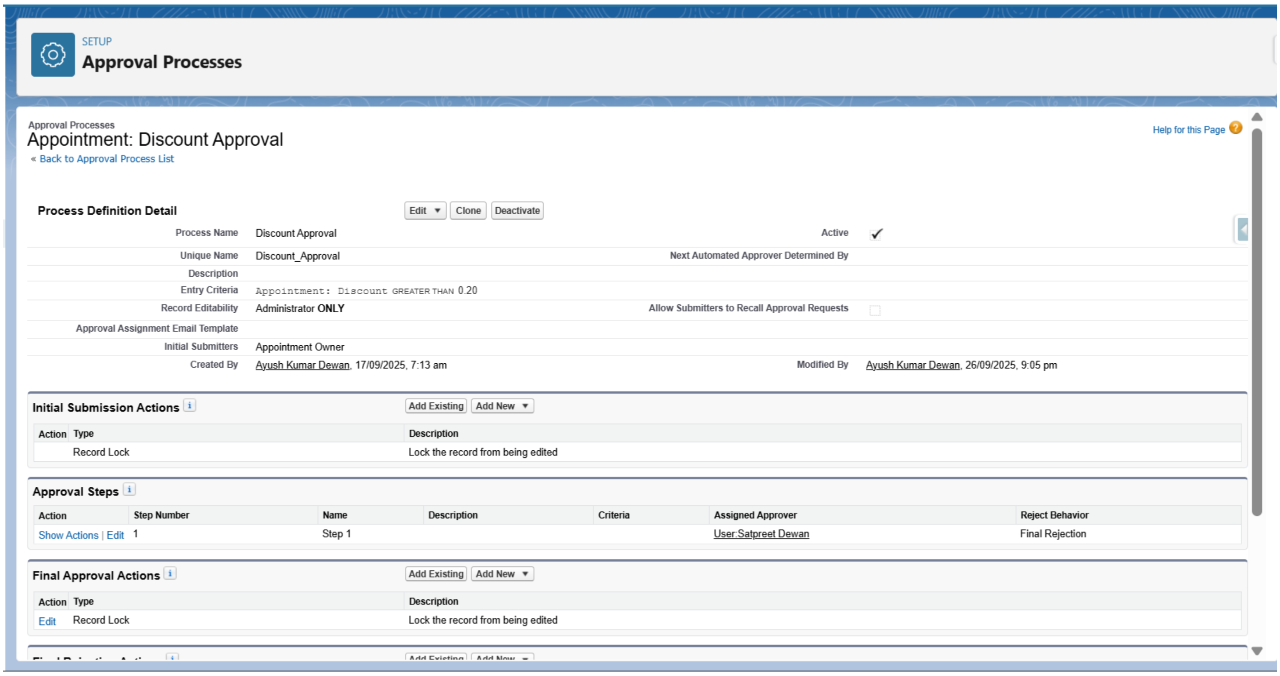
2. & 3. Workflow Rules & Process Builder

- **Purpose:** To understand the role of Salesforce's legacy automation tools.
- **Steps Followed:** As a modern best practice, these tools were **intentionally not used**. All automation was instead created using the more powerful and efficient **Flow Builder**, demonstrating that the project is built on a modern, future-proof platform.

4. Approval Process

- **Purpose:** To automate a formal review and approval workflow for appointments that meet specific criteria, ensuring management oversight on discounts.
- **Steps Followed:**

1. **Prerequisite:** A `Discount` (Percent) field was first added to the `Appointment` object.
2. A new `Discount Approval` process was created on the `Appointment` object using the Jump Start Wizard.
3. **Entry Criteria:** The process was configured to trigger automatically if an `Appointment`'s `Discount` is greater than 20%.
4. **Approver:** The approver was set to be the user's `Manager` (based on the Role Hierarchy), making the process dynamic and scalable.
5. **Deployment:** The process was **Activated**, and the "Submit for Approval" button and "Approval History" related list were automatically added to the `Appointment` page layout for user visibility.

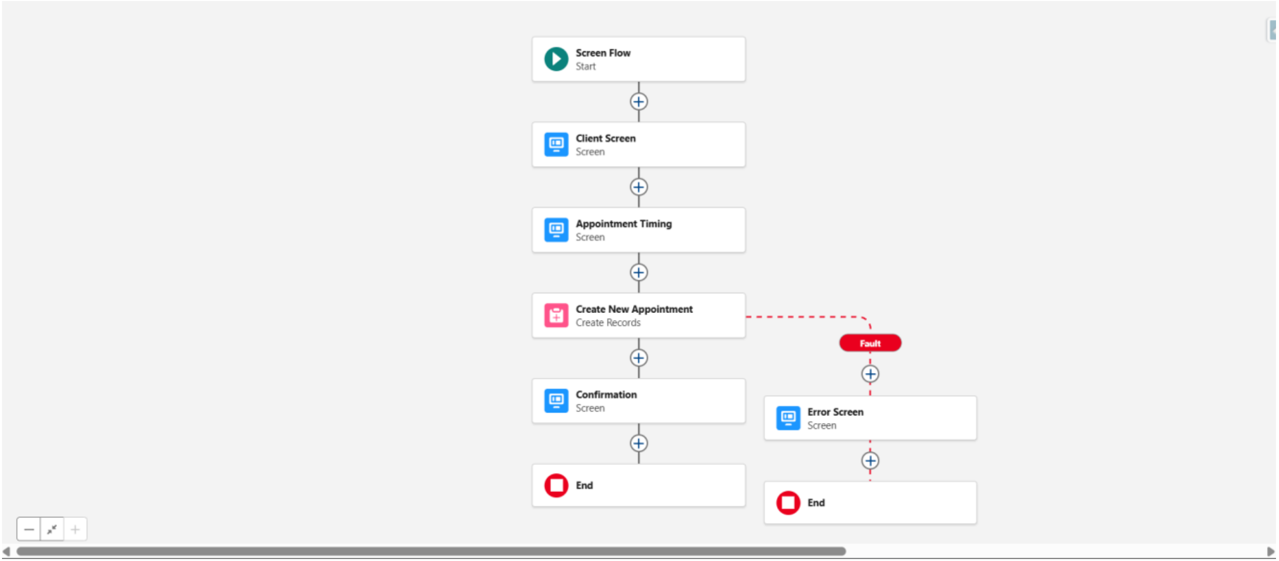


5. Flow Builder

Flow Builder was the primary tool for all automation. Four distinct flows were built, demonstrating mastery of Screen, Record-Triggered, and Scheduled flows.

Flow #1: `Booking Wizard` (Screen Flow)

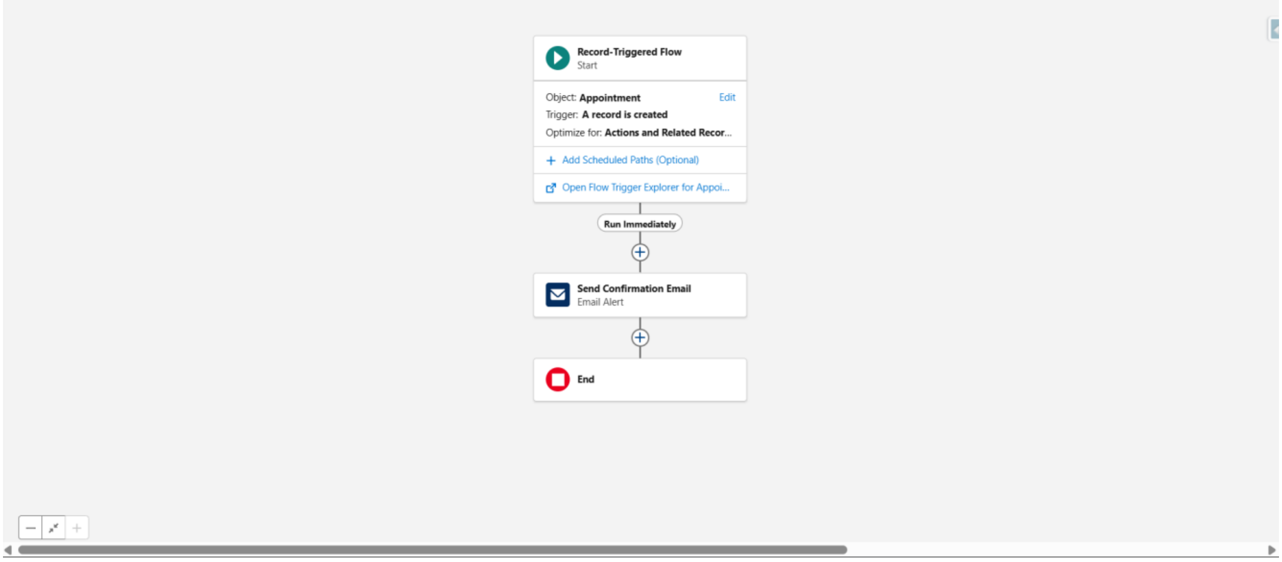
- **Purpose:** To create a step-by-step wizard for the receptionist, making the booking process fast, intuitive, and error-proof.
- **Steps Followed:**
  1. A **Screen Flow** was created from scratch.
  2. **Screen 1 ("Client Information"):** A screen was built with a `Lookup` component configured to search for existing `Contact` records.
  3. **Screen 2 ("Appointment Details"):** A second screen was built with `Lookup` components for `Service__c` and `Stylist__c`, and a `Date Time` component. These components were made **Required** to prevent errors.



4. **Create Record Step:** A `Create Records` element was configured to create a new `Appointment` record, mapping field values from the screen components (e.g., `Client__c` was set from `{!LookupClient.recordId}`).
5. **Error Handling:** A final "Error Screen" was added and connected to the `Create Records` element using a **Fault Path** to display custom Apex error messages.
6. **Deployment:** A custom **List Button** (`New Appointment`) was created to launch the flow. This button was then added to the `Appointment` object's List View layout.
7. **Permissions:** Extensive debugging was performed, and the `Receptionist Profile` was granted all necessary object, field, and flow access permissions.

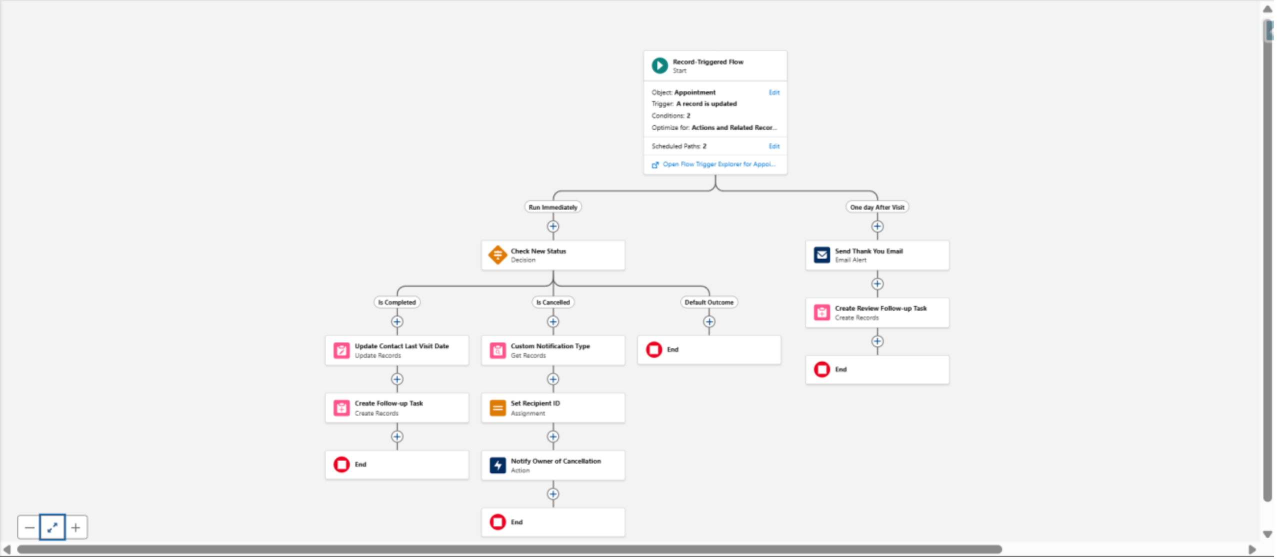
Flow #2: `New Appointment Confirmation` (Record-Triggered Flow)

- **Purpose:** To automate client communication by sending an instant confirmation email when a new appointment is booked.
- **Steps Followed:**
  1. **Prerequisites:** A `Classic Email Template` and a corresponding `Email Alert` were created. Advanced merge fields (`{!Appointment__c.Client__r.Name}`) were used to pull data from related records.
  2. A **Record-Triggered Flow** was created.
  3. **Trigger:** The flow was configured to run *after* a new `Appointment` record is `created` and optimized for **`Actions and Related Records`** to ensure merge fields work correctly.
  4. **Action:** An `Action` element was added to call the pre-configured Email Alert. The flow was then **Activated**.



Flow #3: `Appointment - Status Change Trigger` (Record-Triggered Flow)

- **Purpose:** To automate all the business processes that happen after an appointment is completed or cancelled.
- **Steps Followed:**
  1. **Prerequisites:** A `Last Visit Date` field was created on the `Contact` object, and a `Custom Notification Type` was built.
  2. A **Record-Triggered Flow** was created to run when an `Appointment` is `updated`.
  3. **Start Logic:** The Start element was configured to run if the `Status` equals "Completed" OR "Cancelled" to accommodate the scheduled path limitation.
  4. **Decision Element:** A `Decision` element was used to create two main paths: "Is Completed" and "Is Cancelled."
  5. **"Is Completed" Path:** An `Update Records` element automatically updates the related `Contact`'s `Last` Visit `Date`, and a `Create Records` element automatically creates a new `Task` for rebooking.
  6. **"Is Cancelled" Path:** An `Action` element sends a `Custom Notification` to the Salon Owner.
  7. **Scheduled Path:** A scheduled path, "One day After Visit," was added to send a "Thank You" email and create a "Follow up for review" `Task` 24 hours after the appointment.



Flow #4: `Daily Appointment Reminders` (Scheduled-Triggered Flow)

- **Purpose:** To create a reliable, automated daily process for reminding clients of their appointments for the next day.
- **Steps Followed:**
    1. A **Scheduled-Triggered Flow** was created.
    2. **Schedule:** The flow was scheduled to run every morning at 9:00 AM.
    3. **Start Logic:** It is configured to get all `Appointment` records where the `Status` is "Scheduled".
    4. **Decision Element:** A `Decision` element was added with a time-zone safe formula (`DATEVALUE({!$Record.Appointment_Date_Time__c})`) to check if the appointment date is equal to tomorrow's date (`TODAY() + 1`).
    5. **Actions:** For each matching appointment, the flow performs two actions: it sends the `Appointment Confirmation` **Email Alert** and creates a completed **Task** with the subject "SMS Reminder Sent" as a "mock" SMS log.