Problem Statement

Managing accommodations for diverse groups such as tourists, students, and working professionals is often challenging due to fragmented platforms and lack of personalization.

- **Tourists & Travelers** need short-term stays with flexibility.
- **Students & Professionals** require long-term, affordable, and reliable housing like PGs, hostels, or rentals.
  Traditional property management systems lack centralized communication, efficient booking handling, transparent ratings/reviews, and host–guest coordination.
  This results in:
- Inefficient property discovery
- Lack of trust between host and guest
- Manual management of payments & complaints
- Limited visibility of guest/host performance

StayEase CRM solves these issues by creating a **Salesforce-powered centralized platform** for property management, booking, guest handling, and payments.

Industry Type

- **B2C (Business-to-Consumer):**
  StayEase directly connects property hosts with end-users (tourists, students, working professionals).

Target Users

1. **Hosts** – Individuals or businesses listing their properties.
2. **Guests** – Tourists, students, or professionals booking stays.
3. **Admins** – Oversee operations, resolve disputes, and manage CRM functionalities.

Key Objectives

- Centralize property listings, bookings, payments, and complaints.
- Provide a **trust-building mechanism** via guest ratings and host performance.
- Offer different stay types: **short-term (Airbnb-like)** and **long-term (student/professional rentals)**.
- Enhance **customer satisfaction** with transparent processes and automation

Value Proposition

- For **Guests** → Easy property discovery, secure payments, and transparent ratings.
- For **Hosts** → Better visibility, guest management, and simplified bookings.
- For **Admin/Organization** → A scalable CRM system on Salesforce that tracks all activities in one place.

🔹 Phase 1: Problem Understanding & Scope

Introduction

Accommodation management is a common challenge for multiple groups:

- **Tourists & Travellers** – need short-term stays for leisure, business trips, or events.
- **Students & Working Professionals** – require long-term affordable rentals such as PGs, hostels, or shared flats.

The **existing market** has several limitations:

- **Unverified Listings** – Many platforms allow fake or outdated property details.
- **Broker Dependence** – Manual, broker-driven processes inflate costs.
- **Communication Gaps** – Hosts and guests lack structured communication channels.
- **Low Transparency** – Guests face uncertainty regarding booking status, payments, and reviews.
-

Proposed Solution

StayEase CRM is a Salesforce-based CRM project designed to create a structured and trustworthy **accommodation and rental platform**. It takes inspiration from Airbnb's short-term rental model while extending to long-term housing for students and professionals.

Key Features of the Solution:

- Centralized database of verified properties.
- Seamless booking flow for **short-term** and **long-term** stays.
- Role-based data access for **Admin, Hosts, Guests, and Managers**.
- Automation for booking status, payments, and complaint resolution.
- Reports and dashboards to monitor occupancy, revenue, and service quality.

Objectives

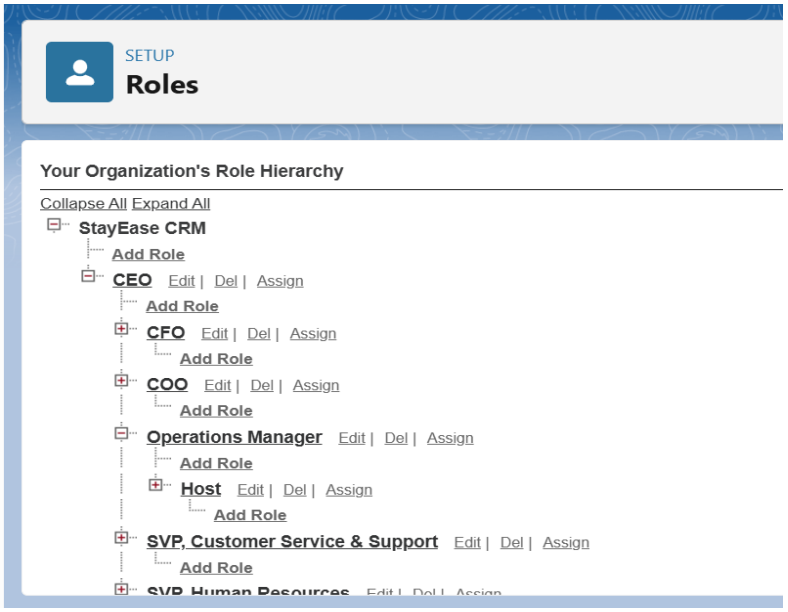
1. Create a **unified accommodation platform** with transparent workflows.
2. Support **short-term bookings** for tourists and **long-term rentals** for students/professionals.
3. Provide **role-based security** for data integrity.
4. Use Salesforce automation (Flows, Validation Rules, Approvals) to improve efficiency.
5. Deliver **business insights** through dashboards and reports.

🔹 Phase 2: Org Setup & Security

To implement the foundation of StayEase CRM, **roles, profiles, org-wide defaults, and permissions** are configured.

2.1 Roles (Role Hierarchy)

- **CEO** → Top-level access; visibility into all records, reports, and dashboards.
- **Operations Manager** → Oversees all bookings, complaints, and escalations across the org.
- **Host** → Can only manage their own properties, bookings, and related complaints.
- **Guest (Traveller/Student/Professional)** → Can book properties and raise complaints related to their own bookings.
- **System Administrator** → Full access, controls setup and configurations (you).



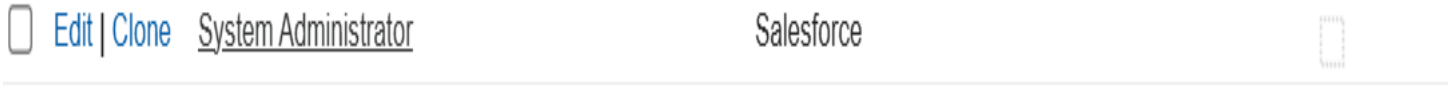
2.2 Profiles

Profiles define **what a user can do** inside Salesforce.

- **Host Profile**
  - Can create, view, and edit only their properties.
  - Can view bookings related to their properties.
  - Limited access to complaints (only related ones)

- **Guest Profile**
  - Can create and manage only their own bookings.
  - Can raise complaints for their own bookings/properties.
  - Read-only access to property listings.

- **Operations Manager Profile**
  - Full access to all bookings and complaints.
  - Can intervene/escalate complaints.
  - Cannot perform system-level configurations.
- **CEO Profile**
  - Read-only access to all data.
  - Access to dashboards and reports for strategic decision-making.
- **System Admin Profile**
  - Full CRUD (Create, Read, Update, Delete) permissions on all objects.
  - Handles automation, security, and backend setup.



2.3 Org-Wide Defaults (OWD)

Default record-level security settings:

- **Property__c** → Private (Hosts can only see their properties).
- **Booking__c** → Private (Guests can see their own bookings, Hosts can see bookings related to their properties).
- **Complaint__c** → Private (Guests see their complaints; Ops Manager sees all).
- **Payment__c** → Private (Linked to bookings; visible only to Guest, Host, and Ops Manager).

2.4 Sharing Rules

- **Hosts** get automatic access to bookings made on their properties.
- **Ops Managers** get access to all complaints and escalated bookings.
- **CEO** gets read-only access to all objects for reporting.

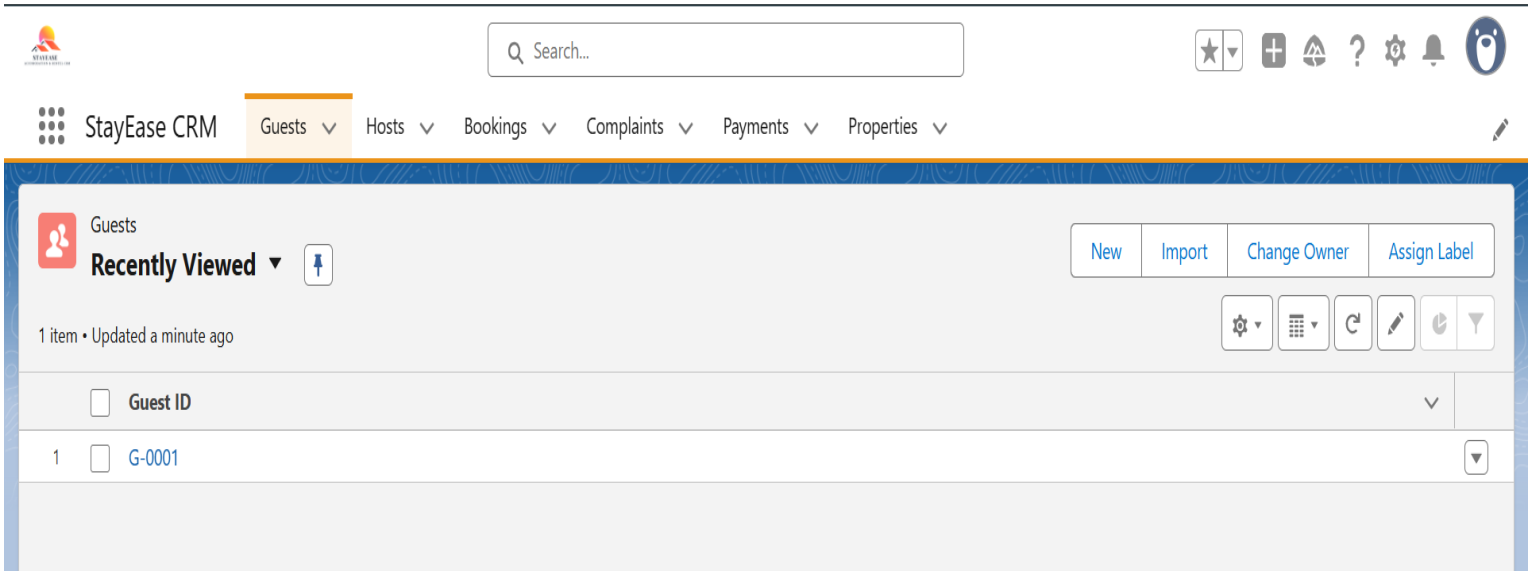
2.5 Permission Sets

Additional flexibility beyond profiles:

- **Host_Extra_Access** → If needed, allow hosts to see limited guest details (like name & contact for bookings).
- **Ops_Manager_Analytics** → Access to advanced dashboards and reports.

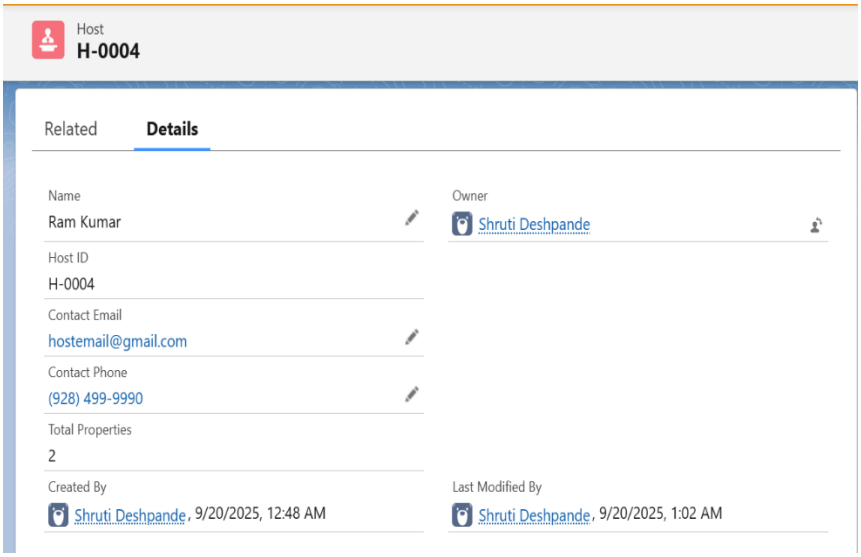
❖ **Outcome of Phase 2:**
At this stage, the org is structured with a secure role hierarchy, profiles, permissions, and OWD rules. Each actor (Admin, CEO, Operations Manager, Host, Guest) has **only the data visibility and access they need**, ensuring data privacy, integrity, and smooth system operation.

❖ Phase 3 Objects & Relationships



1. Host Object

Purpose: Represents the property owner.



Fields:

- Name (Text) – Name of the owner.
- Host id – Auto Number (H - {0000})
- Contact Email – Email field type
- Phone Number – Phone field type
- Total Properties – Number field (will auto update on the addition or deletion of properties associated with the host)

Relationship : Master-Detail relationship with property. Host – master & Property – detail

2. Property

Purpose: Represents the accommodation available for booking.

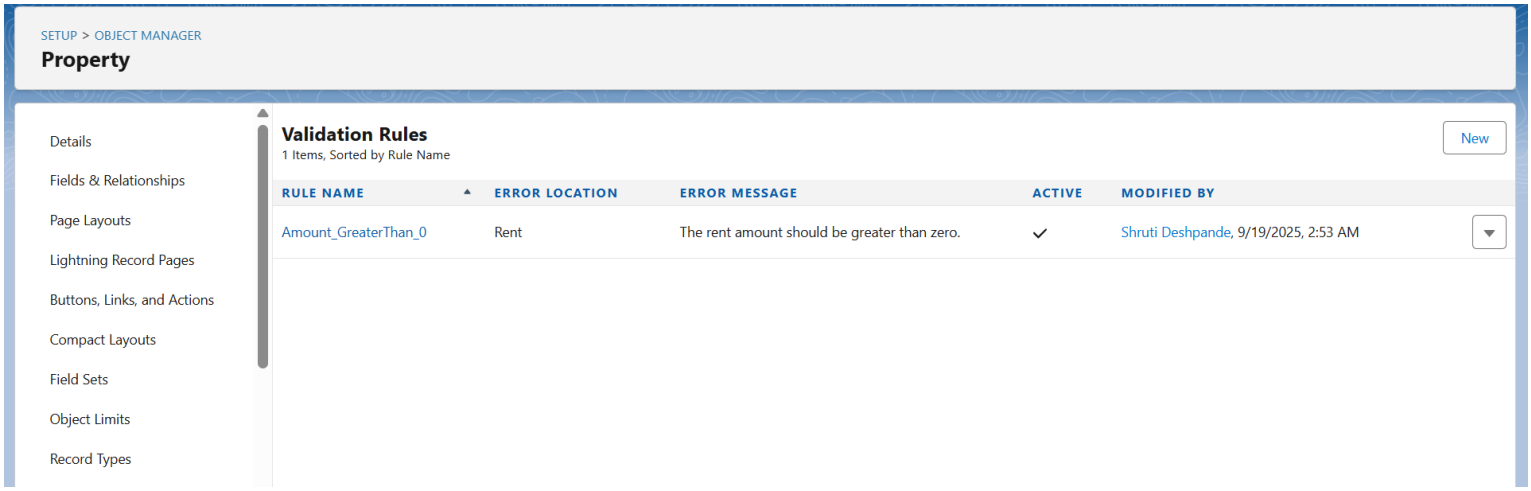
Fields:

- Property ID – Auto Number (P-{0000}).
- Name (Text) – Name of the property.
- Type (Picklist – villa , resort , hostel, PG, flat)
- Location (Text).
- Rent (Currency).
- Rent type (Picklist field – Monthly rent , pre night rent)
- Availability (Checkbox).
- Amenities (Picklist Multiselect – Wifi , AC, Parking , kitchen, cook, semi/ fully furnished)
- Max Occupancy (Number).
- Booking Ratings Sum (Number, hidden).
- Booking Ratings Count (Number, hidden).
- Average Rating (Formula).

Relationship:

- Master-Detail with **Host** (Host → Property).
- Lookup with **Booking** (Booking → Property).
- Lookup with **Complaint** (Complaint → Property).

Validation Rules:

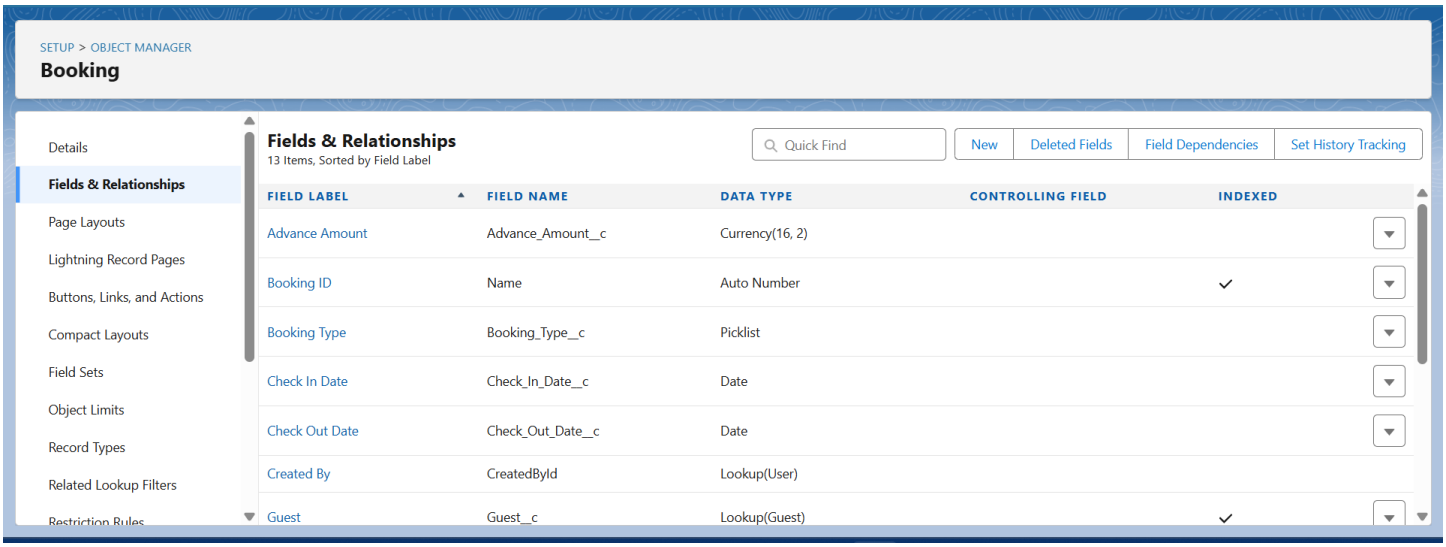


3. Booking Object

Purpose: Represents a reservation made by a Guest.

Fields:

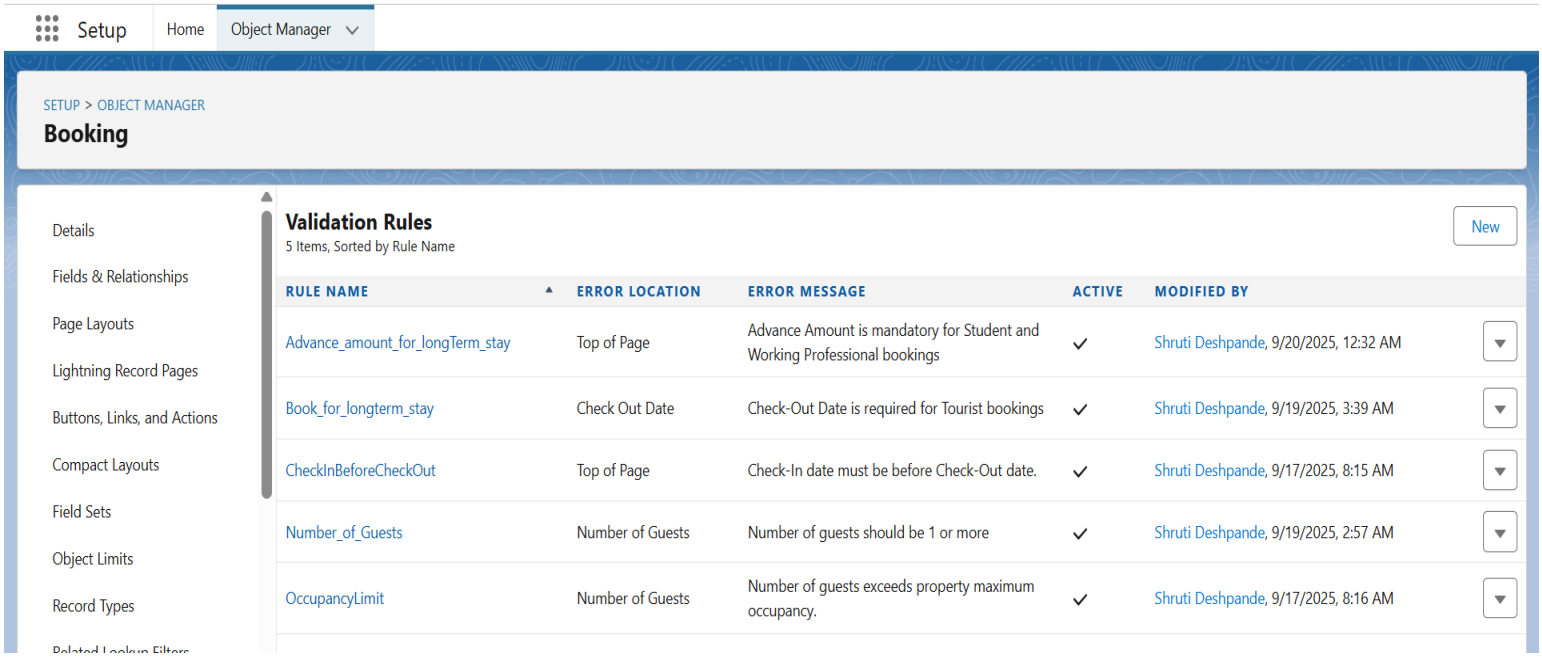
- Booking ID – Auto Number (B-{0000}).
- Booking Type (Picklist)
- Guest (Lookup → Guest).
- Guest Rating (Number)
- Number of guests (Number)
- Total Rent (Currency – total rent to be paid)
- Property (Master-Detail → Property).
- Check-in Date (Date).
- Check-out Date (Date).
- Advance amount (Currency – only for students and working professionals)
- Status (Picklist – Confirmed, Cancelled, Completed).



Relationship:

- Lookup with **Guest**.
- Master-Detail with **Property**.
- Master-Detail with **Payment** (Booking → Payment).

Validation Rules:

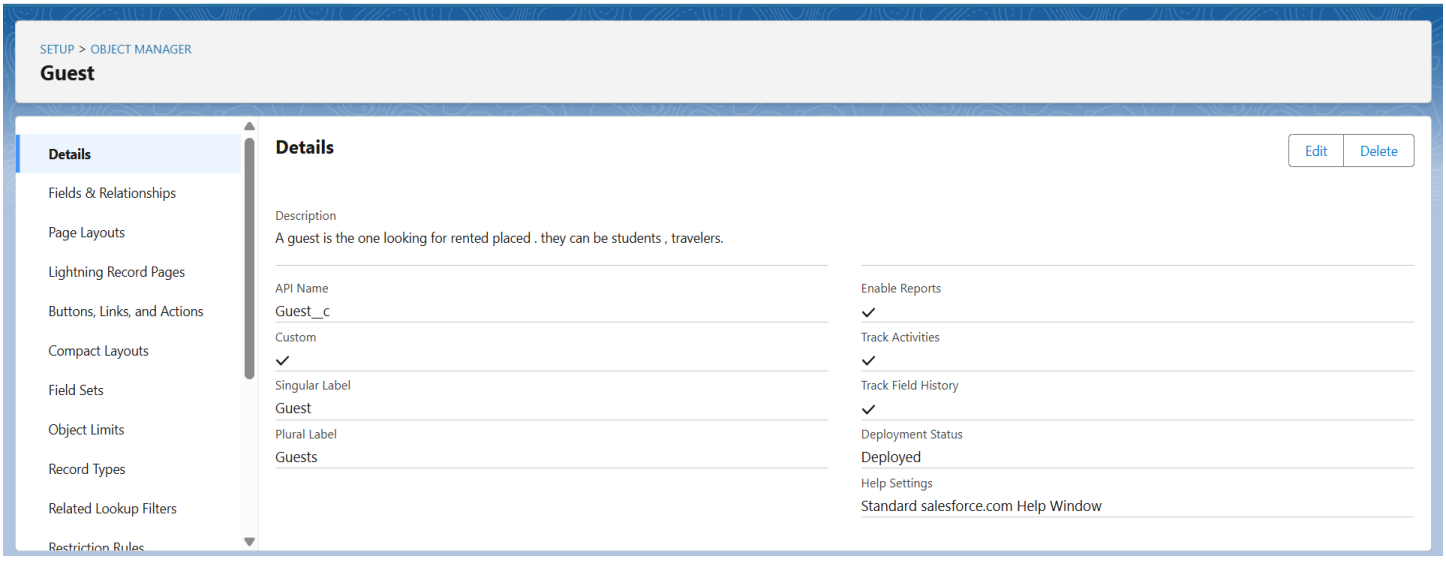


4. Guest Object

Purpose: Represents the customer who books a property.

Fields:

- Full Name (Text) – Guest's full name.
- Guest ID – Auto Number (G-{0000}).
- Email – Email field type.
- Phone – Phone field type.
- Guest Type – Picklist (Tourist, Student/Working Professional).



Relationship:

- Lookup relationship with **Booking** (Booking → Guest).

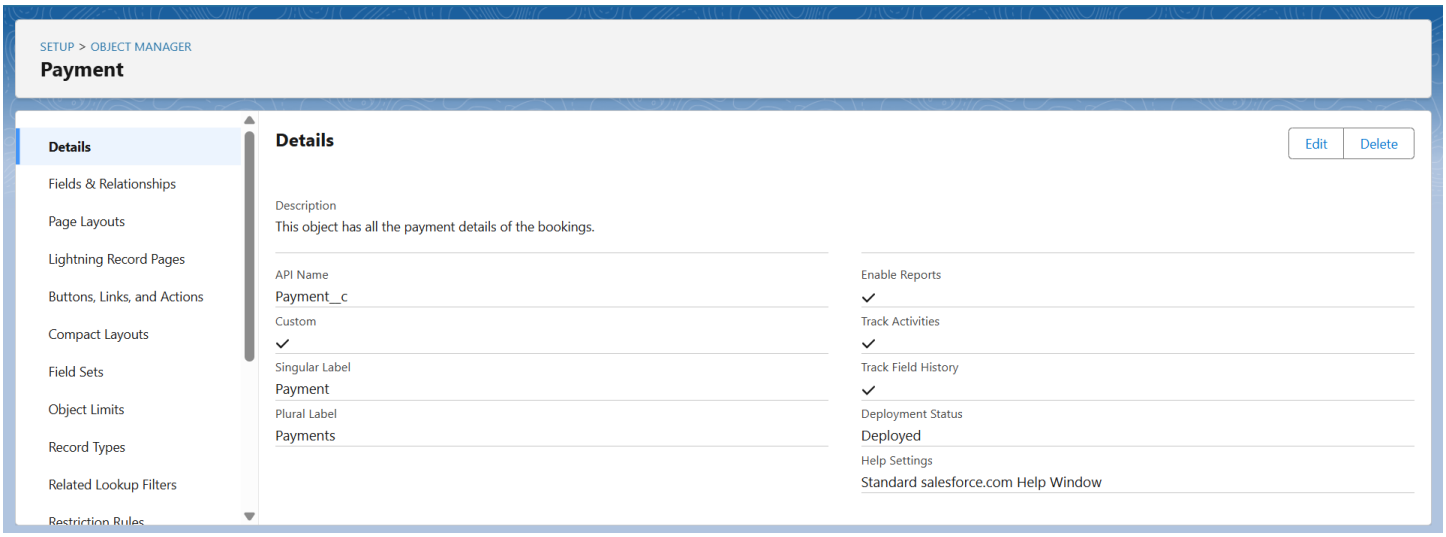
Validation Rules:

- None specific at Guest level.

5. Payment

Purpose: Represents payments made for a booking.
Fields:

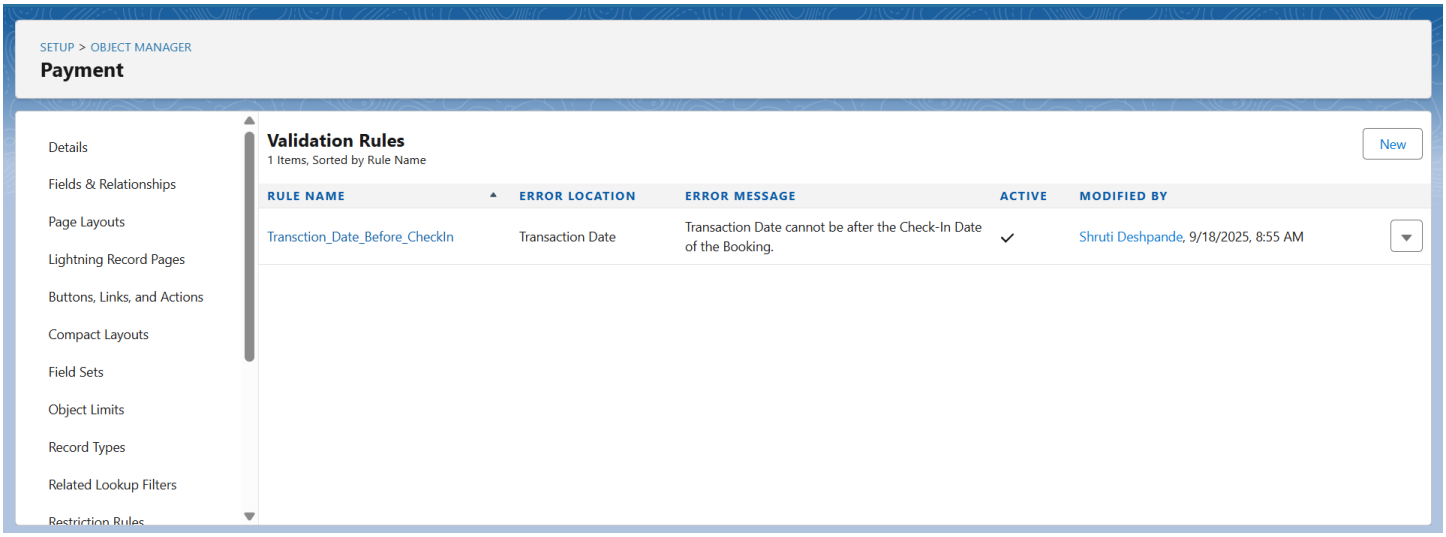
- Payment ID – Auto Number (Pay-{0000}).
- Booking (Master-Detail → Booking).
- Amount (Currency).
- Transaction Date (Date).
- Payment Status (Picklist – Pending, Completed, Failed).



Relationship:

- Master-Detail with **Booking** (Booking → Payment).

Validation Rules:



5. Complaint

Purpose: Represents complaints raised by guests.
Fields:

- Complaint ID – Auto Number (C-{0000}).
- Guest (Lookup → Guest).
- Property (Lookup → Property).
- Complaint Type (Picklist – Maintenance, Cleanliness, Noise, Other).
- Status (Picklist – New, In Progress, Resolved, Closed).
- Description (Long Text Area).

Relationship:

- Lookup with **Guest**.
- Lookup with **Property**.

Validation Rules:

- None specific.

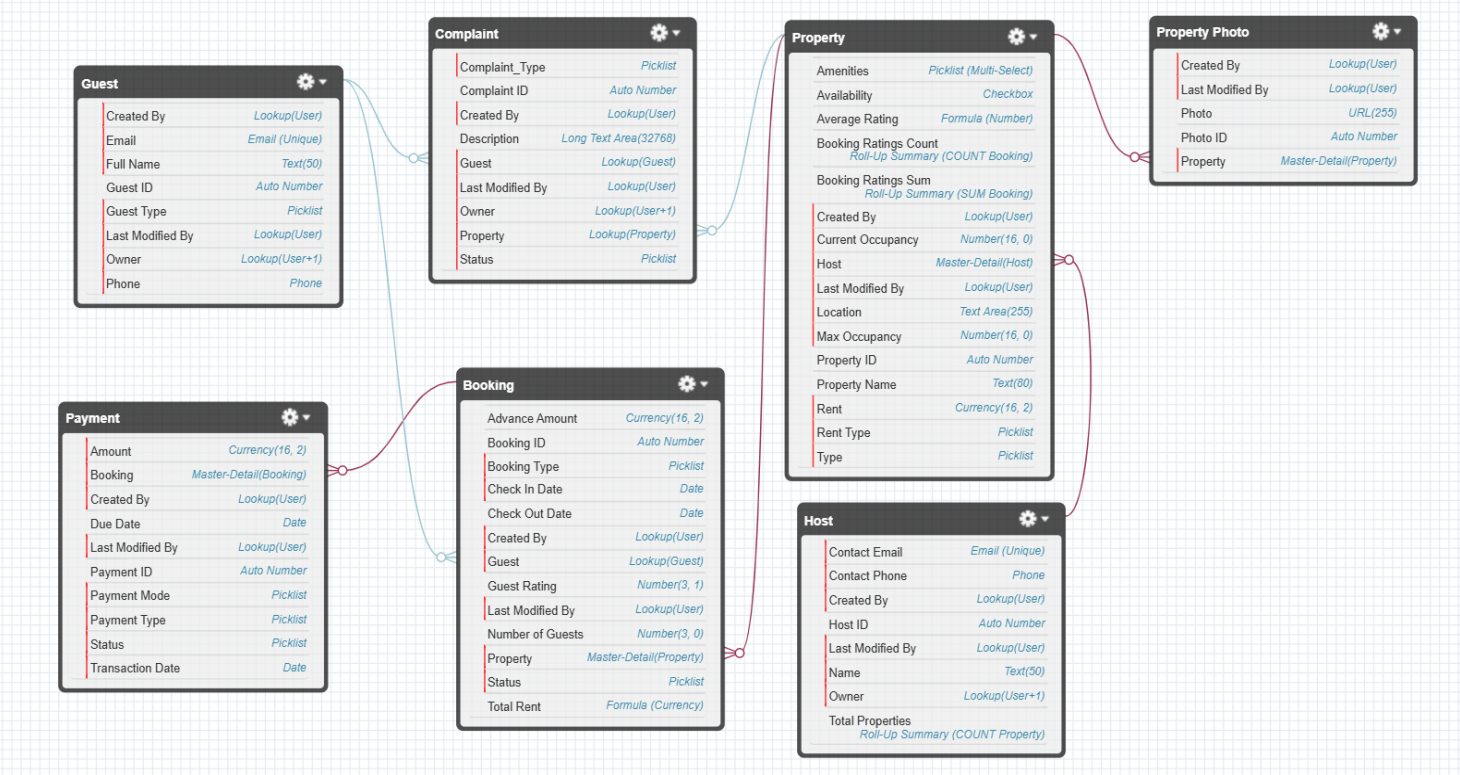


fig : Schema Diagram (Form Schema Builder)

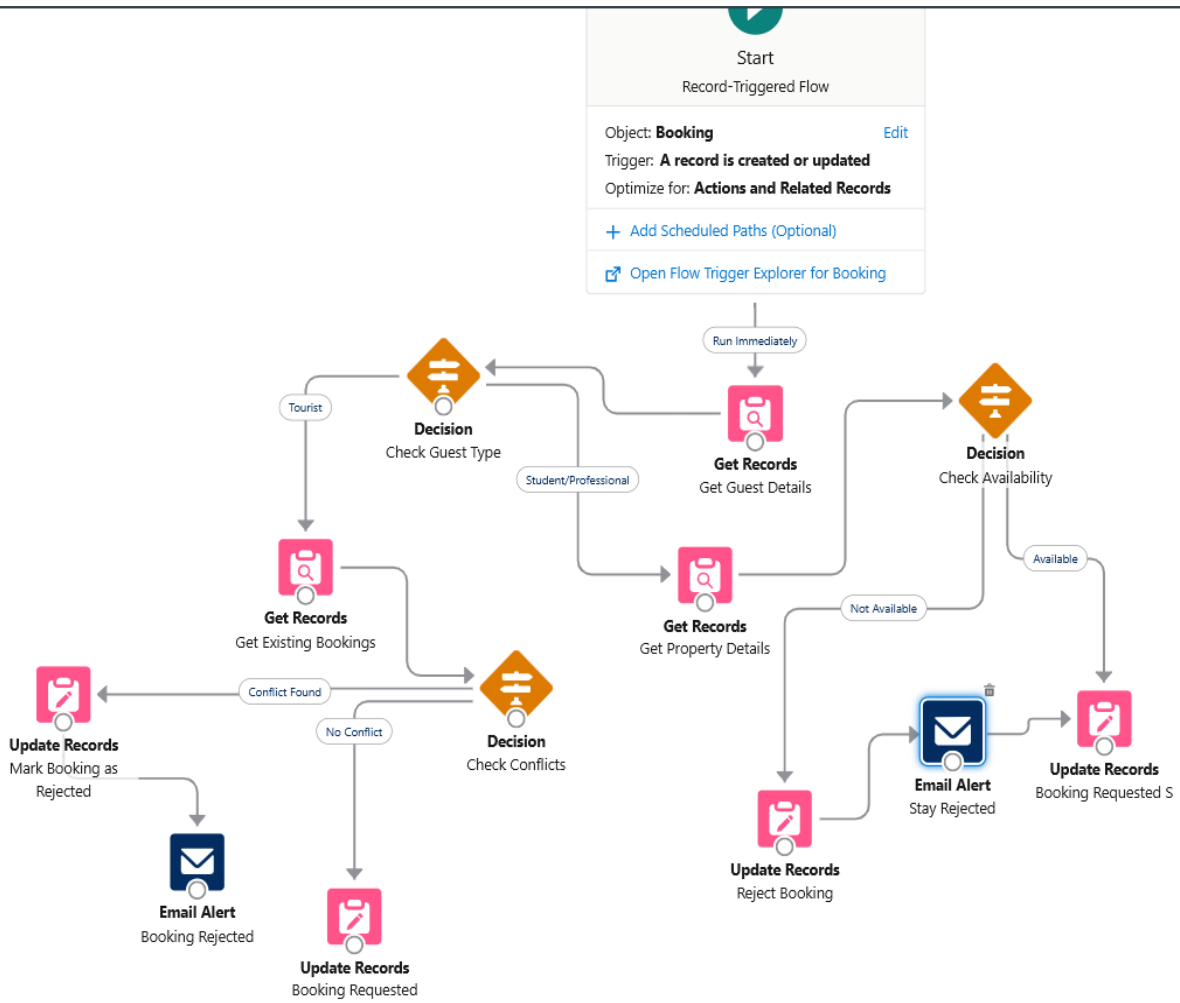
Phase 4 Process Automation (Admin)

Flow 1: Booking Availability Check

Type: Record-Triggered Flow (on Booking)
Trigger: After insert or update

Logic:

- Get Guest Type (Tourist vs Student/Professional).
- If Guest is **Tourist**:
  - Check if selected property already has overlapping bookings.
  - If overlap → Set Status = *Rejected*.
  - If no overlap → Set Status = *Requested (Confirmed only after payment)*.
- If Guest is **Student/Professional**:
  - Check Max Occupancy.
  - If Current Occupancy < Max Occupancy → Confirm.
  - Else → Reject.

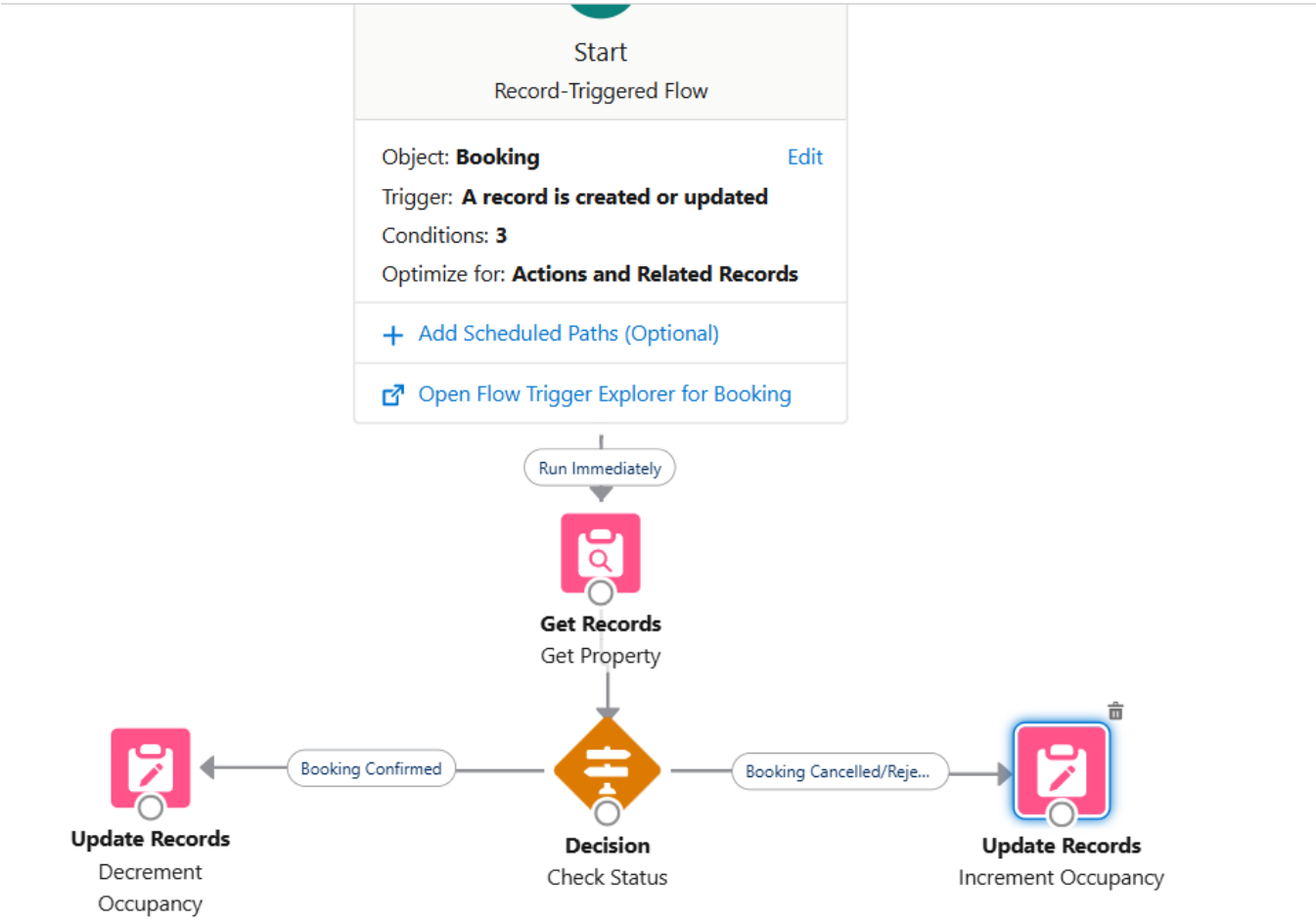


Flow 2: Occupancy Management

Type: Record-Triggered Flow (on Booking, After Save)

Logic:

- On *Confirmed* Booking → Increment Current_Occupancy__c of Property by number of guests.
- On *Cancelled* Booking → Decrement Current_Occupancy__c of Property.



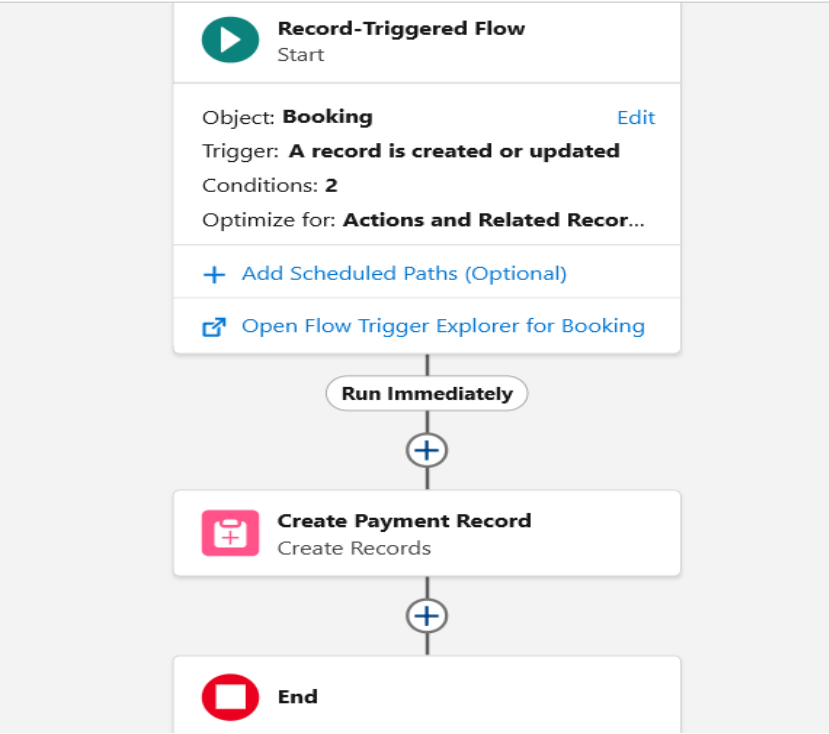
Flow 3: Payment Creation & Notifications

Type: Record-Triggered Flow (on Booking, After Save)

Logic:

- When Booking status = *Confirmed*:
  - Automatically create a **Payment__c** record.
  - Map fields:
    - Booking → Payment.Booking__c
    - Guest → Payment.Guest__c
    - Amount → Booking.Amount__c
    - Transaction Date = TODAY()

- Send **Email Notifications**: (To be implemented)
  - If Confirmed → Send *Confirmation Email Template* (separate templates for Tourist and Student/Professional).
  - If Rejected → Send *Rejection Email Template*.



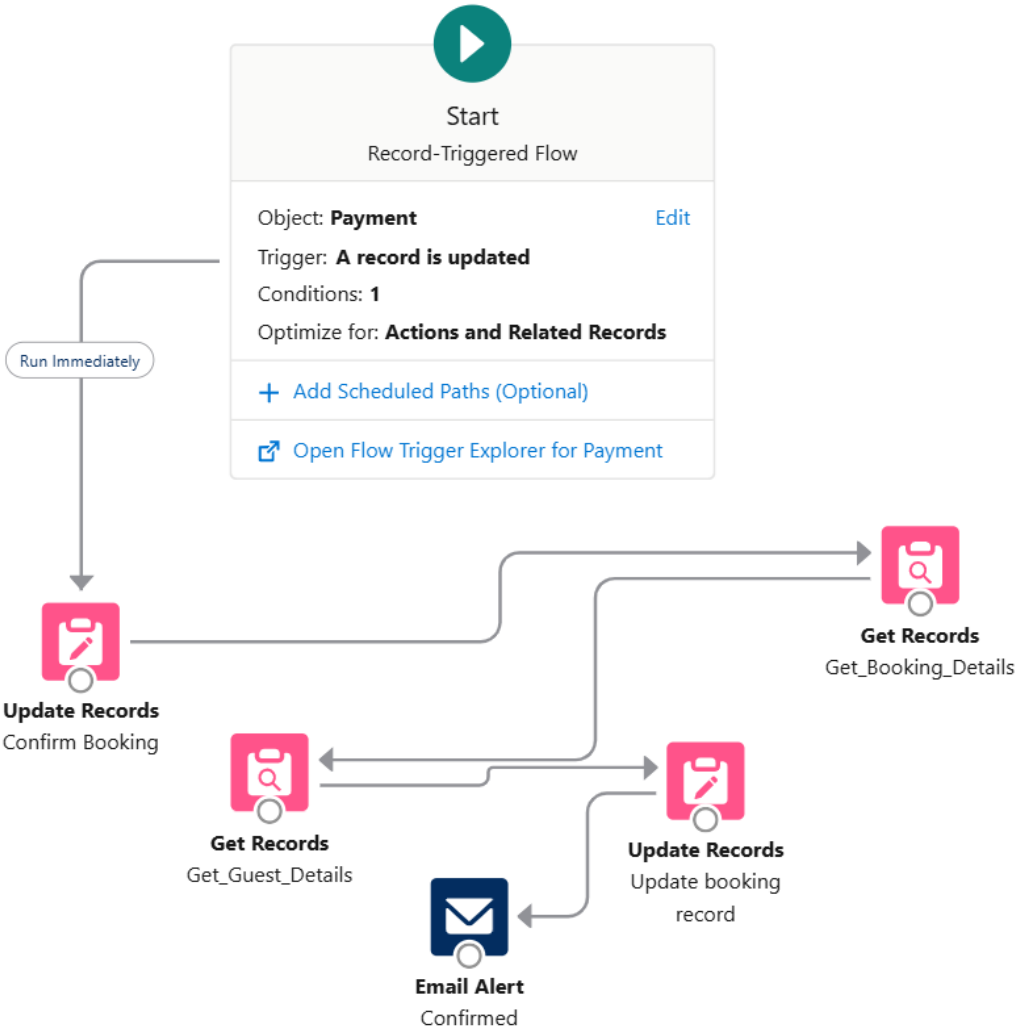
Summary :

- **Phase 3** introduced stricter data model rules, validation for guest types, and default booking behaviors.
- **Phase 4** automated key processes: booking conflict handling, occupancy tracking, auto payment creation, and email notifications.
- Together, these ensure smoother property booking, guest experience, and host management.
- **Flow 3: Payment Creation & Notifications**
- **Flow 3: Payment Creation & Notifications**

- **Flow 4: Confirm Payment**

This is a Record-Triggered Flow.

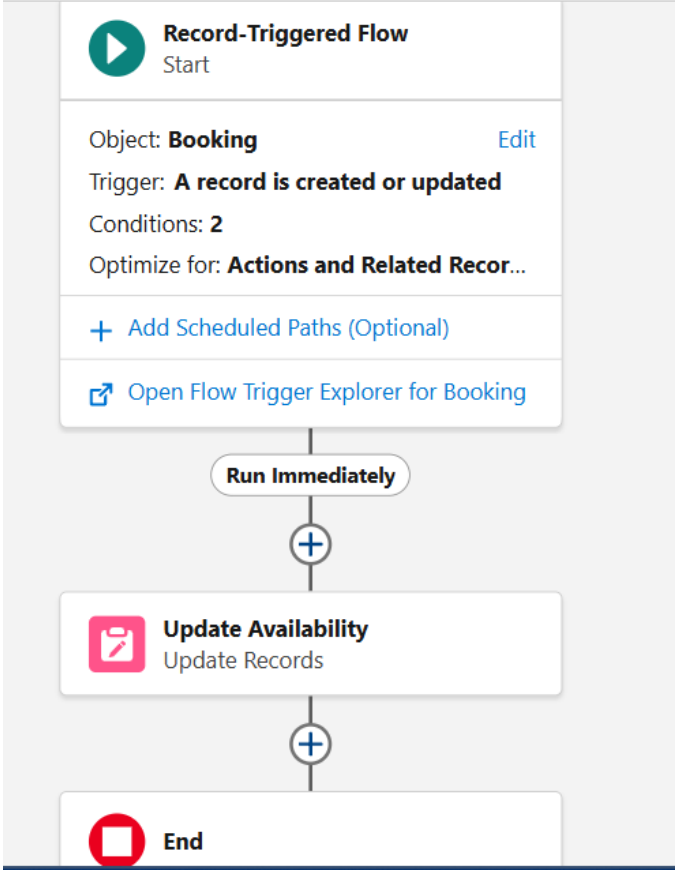
Upon updating a Payment__c record with a status of 'Pending', this flow triggers. It first retrieves the associated Booking__c record and then updates its status to 'Confirmed'. Subsequently, it fetches the Guest__c details from the Booking__c record and updates the Guest_Email__c field in the Booking__c record. Finally, it sends an email alert using the Booking__c.Email_Alerts_for_Booking action.



- **Flow 5: Upadate Availability**

This is a Record-Triggered Flow.

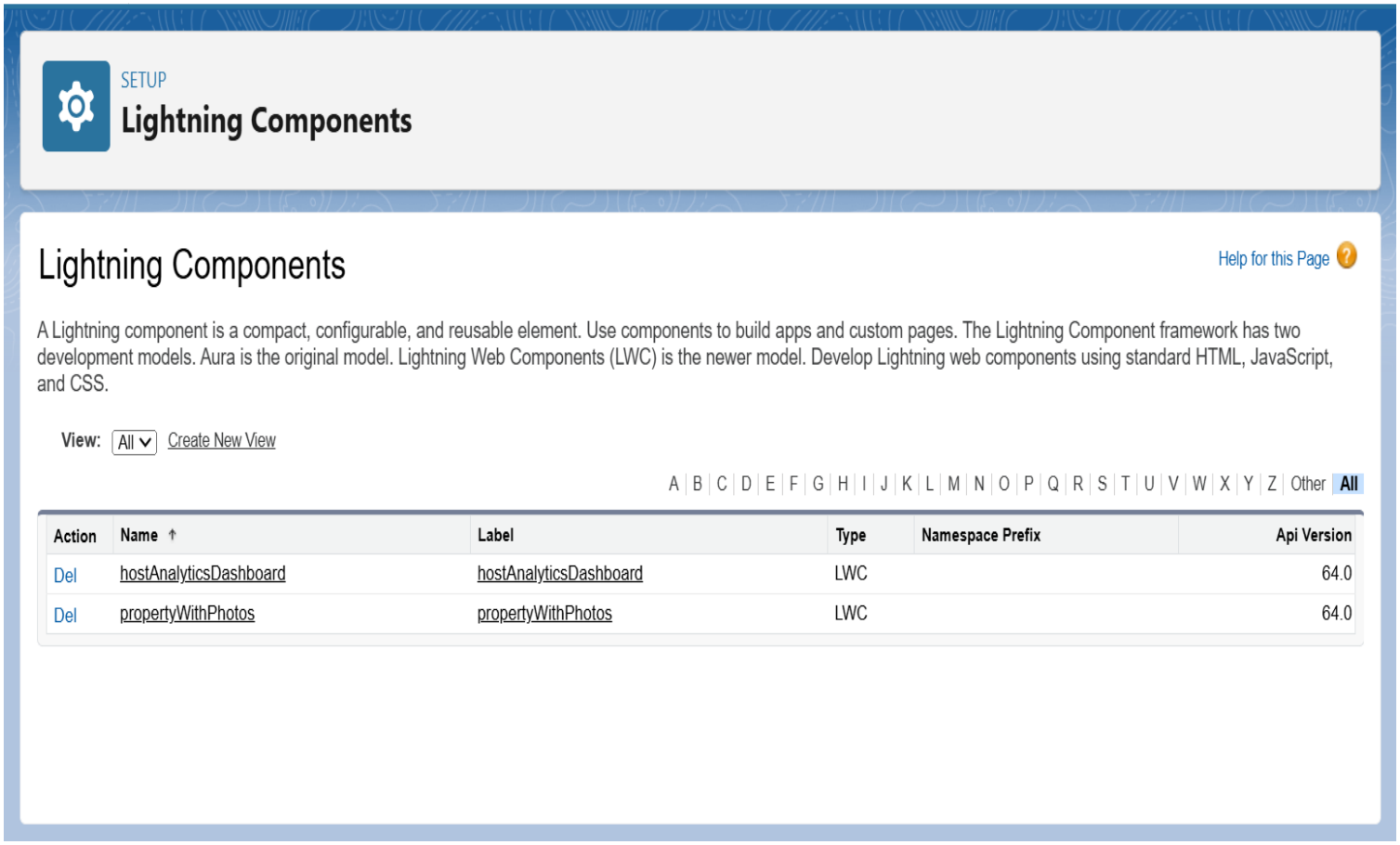
This flow is triggered after a record is created or updated on the Booking__c object. It checks if the Status__c is 'Confirmed' and the Booking_Type__c is 'Tourist'. If true, it updates the Property__c record by setting Availability__c to false if the current availability is true and the Number_of_Guests__c matches the Max_Occupancy__c.

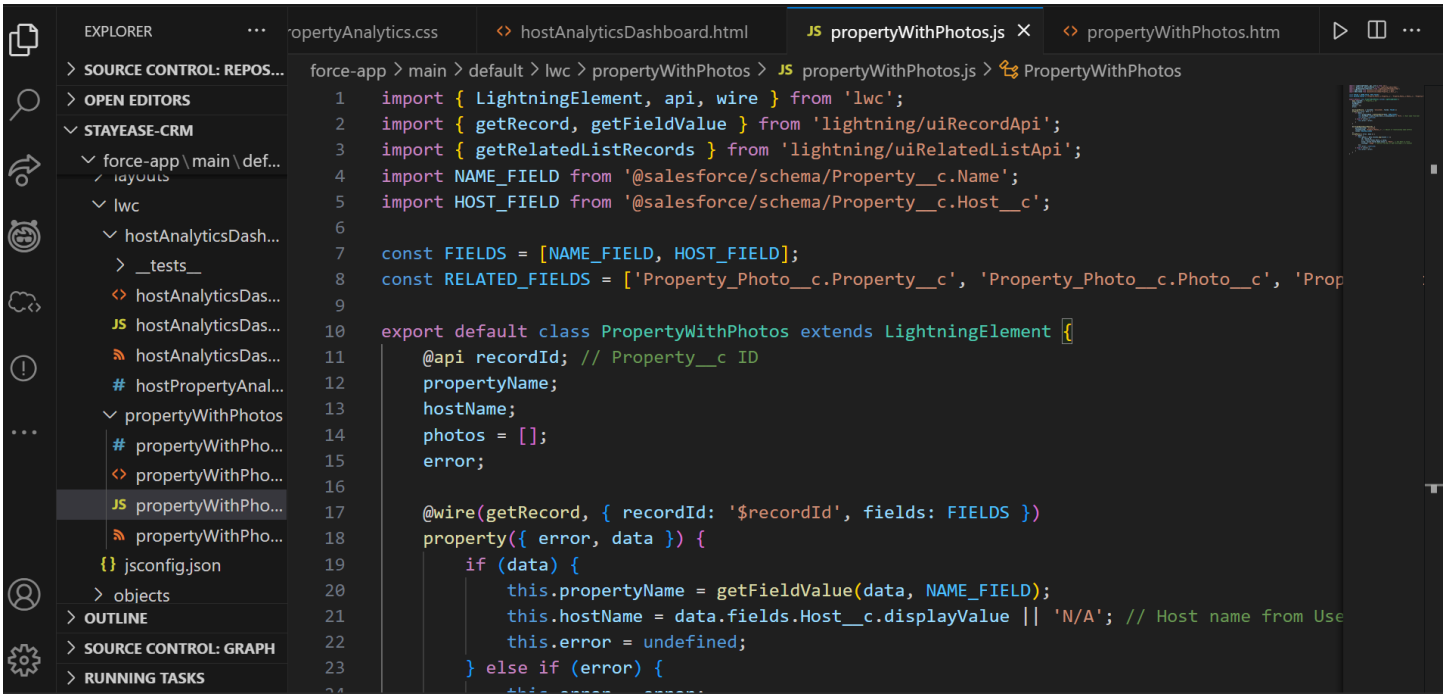


Phase 5 – Apex Enhancements

- Objective: Develop and integrate core features for the StayEase CRM, including the **propertyWithPhotos** Lightning Web Component (LWC) to display property photos and initiate groundwork for host analytics.

- Activities:
- Set up the Salesforce CLI (@salesforce/cli/2.105.6 with Node.js v22.19.0) on a Windows 64-bit system to manage project deployment.
- Created the **propertyWithPhotos LWC** in the force-app/main/default/lwc/propertyWithPhotos directory, including .js, .html, .css, and .js-meta.xml files to display property names, host names, and photos from Property_Photo__c using Photo__c URLs.
- Configured the LWC to leverage the master-detail relationship between Property__c and Property_Photo__c for photo retrieval.
- Initialized the project structure at C:\Users\USER\Desktop\StayEase-CRM and authenticated the DevOrg alias for deployment.



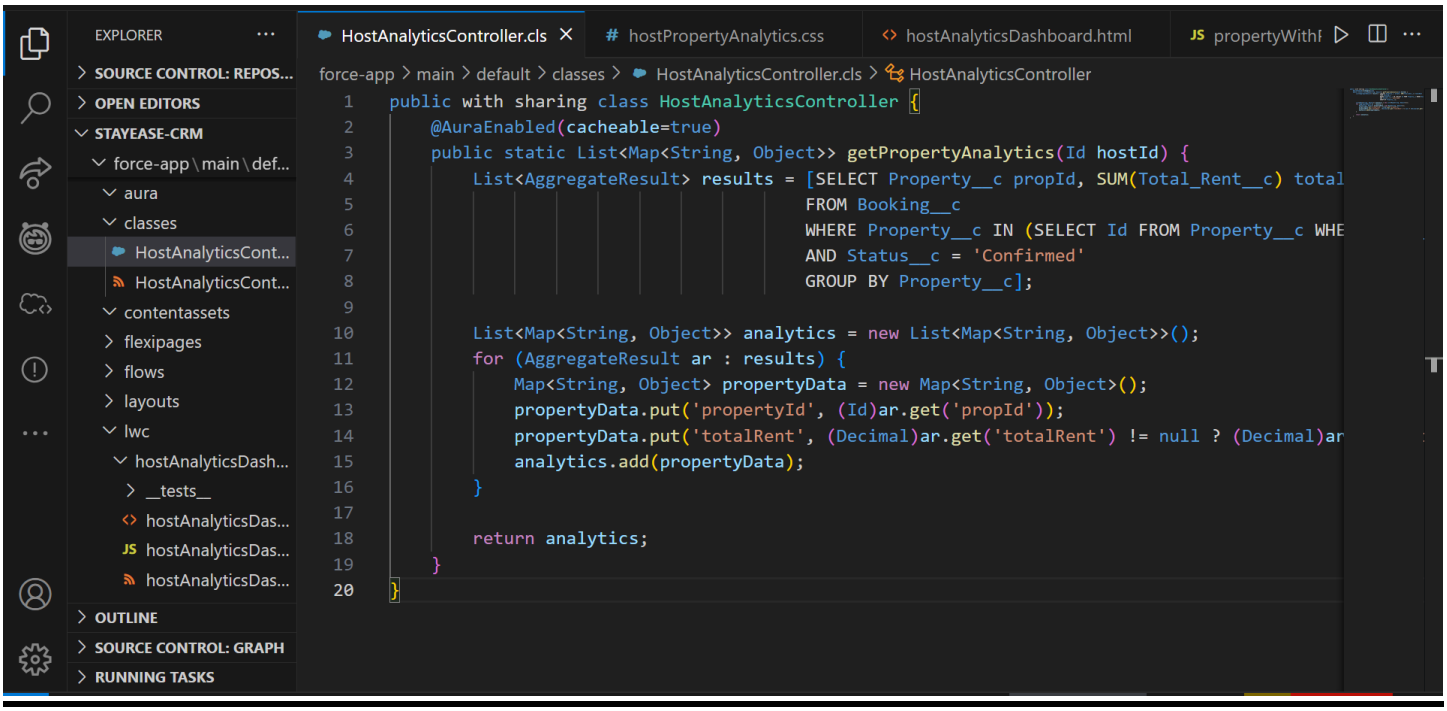


✓ **Outcome**: Successfully developed and partially deployed the propertyWithPhotos LWC. The component was prepared for integration into PropertyBrowsePage, with deployment issues addressed through CLI syntax correction. Further testing and page integration were planned for Phase 6.

Phase 6: User Interface Development

- **Objective**: Enhance the StayEase CRM by adding host analytics via the hostAnalyticsDashboard LWC, integrate propertyWithPhotos into PropertyBrowsePage, and ensure full deployment to the DevOrg.
- **Activities**:

  - Developed the **hostAnalyticsDashboard LWC** to display total rent generated per property for each host, using Apex class HostAnalyticsController to aggregate data from Booking__c records linked via Property__c.Host__c.
  - Created and refined **HostAnalyticsController.cls** to handle property analytics, adjusting SOQL queries to account for the Property__c master-detail relationship on Booking__c and the Host__c lookup on Property__c.
  - Resolved multiple deployment errors, including a missing HostAnalyticsController class, incorrect SOQL field (Host__c on Booking__c), and LWC import syntax, culminating in a successful deployment
  - Attempted to add hostAnalyticsDashboard to the Host User record page and propertyWithPhotos to PropertyBrowsePage using Lightning App Builder, ensuring proper recordId configuration.

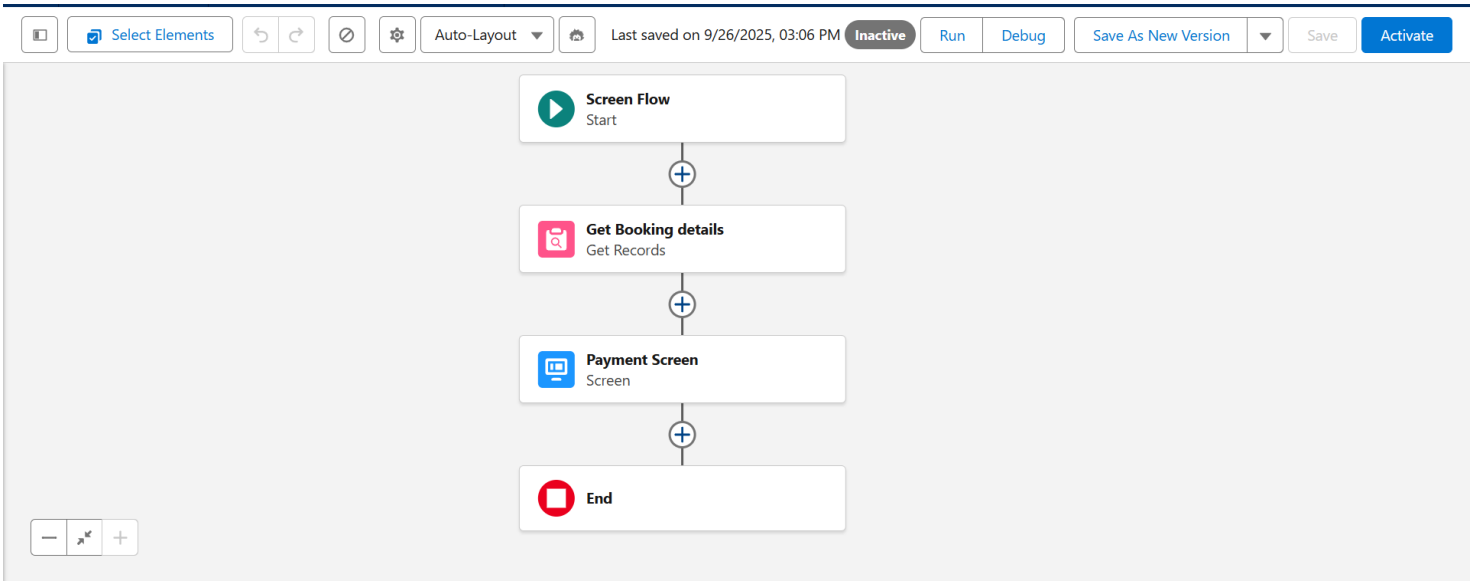


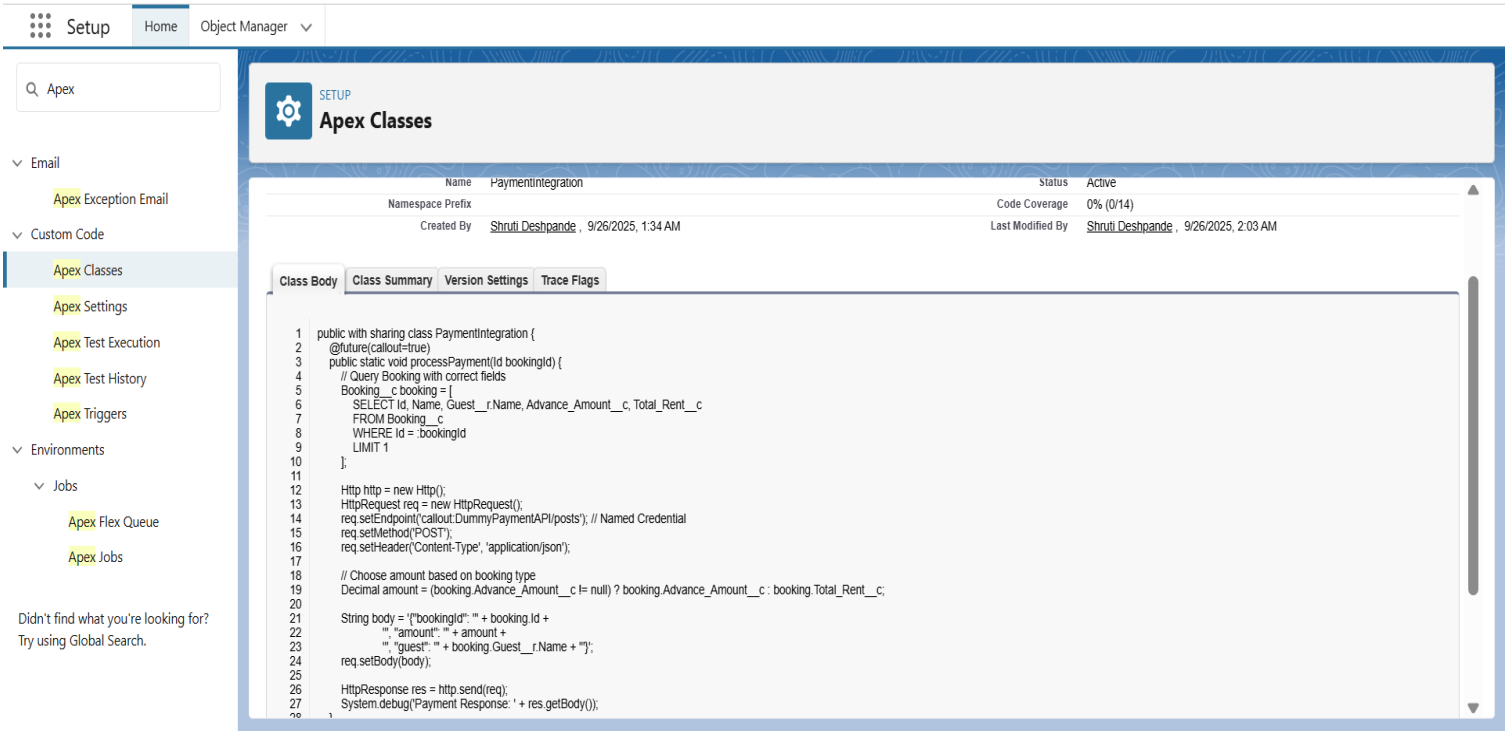
Phase 7 Integration & External Access

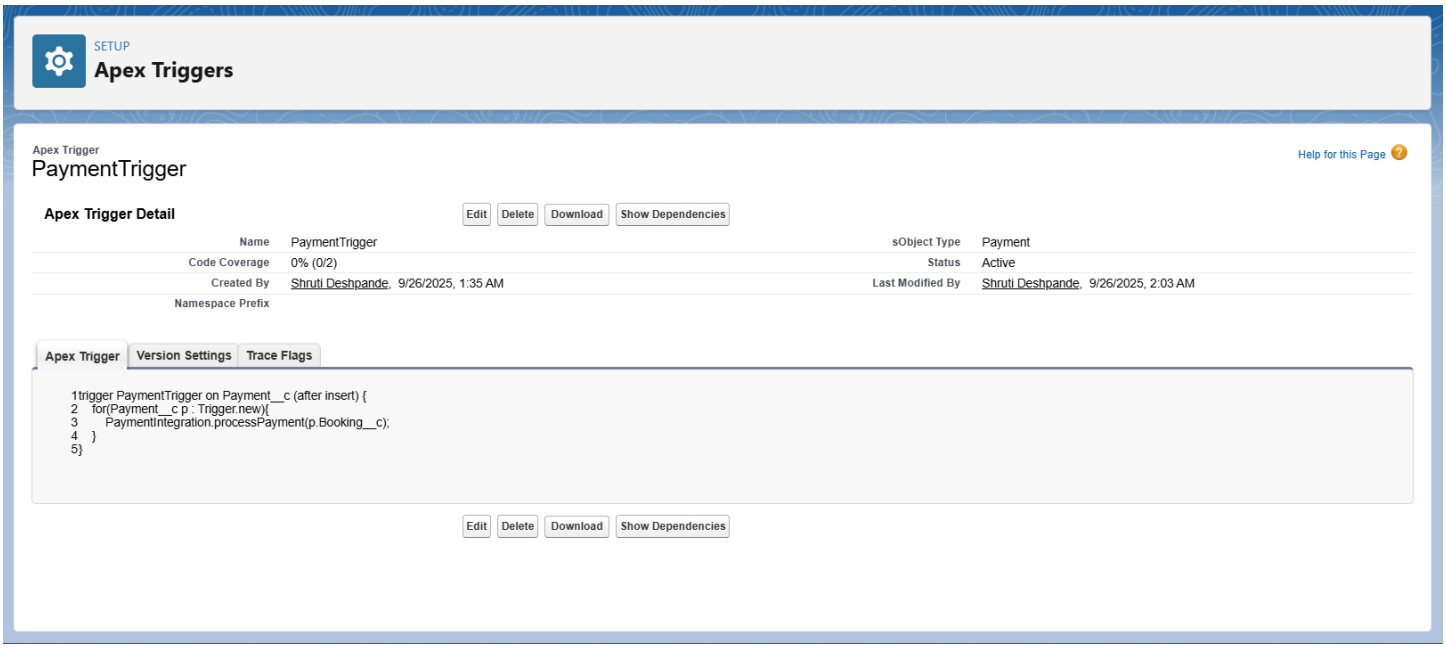
In this phase, a **Screen Flow** was implemented to handle guest payments directly from the Booking record. The flow simplifies the process by automatically fetching booking details, calculating the required amount (either *Advance_Amount__c* for long-term stays or *Total_Rent__c* for short-term/tourist bookings), and guiding the user through a payment process. Guests are prompted to select a payment method (UPI, Card, Net Banking, or Cash) and upon confirmation, a corresponding **Payment__c record** is created and linked to the booking.

For demonstration purposes, a dummy API callout is integrated using Apex to simulate a payment gateway. Once the transaction is recorded, the flow displays a success confirmation to the user. To provide seamless access, a custom **Quick Action** labeled *"Make Payment"* is added on the Booking record page, allowing users to trigger the payment process with a single click.

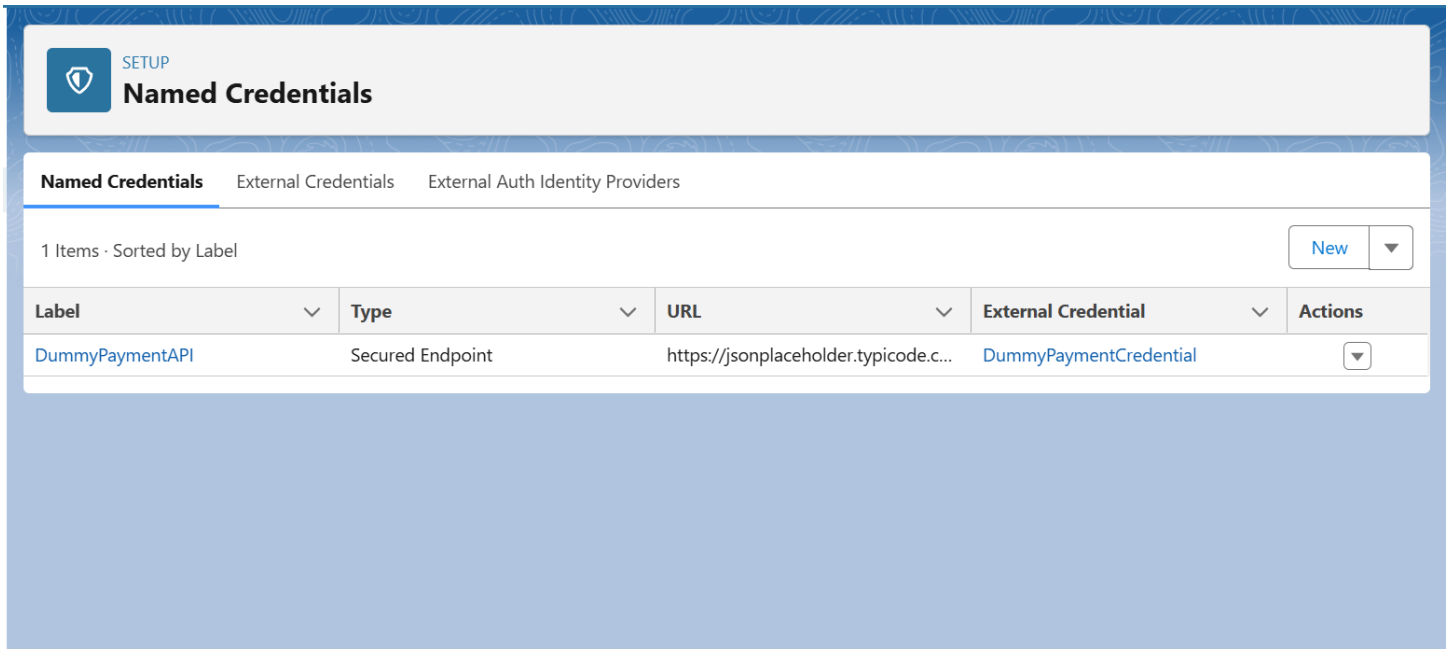
This approach not only reduces manual effort but also provides a user-friendly interface for handling payments, ensuring that long-term stays (students and working professionals) pay advance amounts while short-term tourists generate a complete payment record automatically.







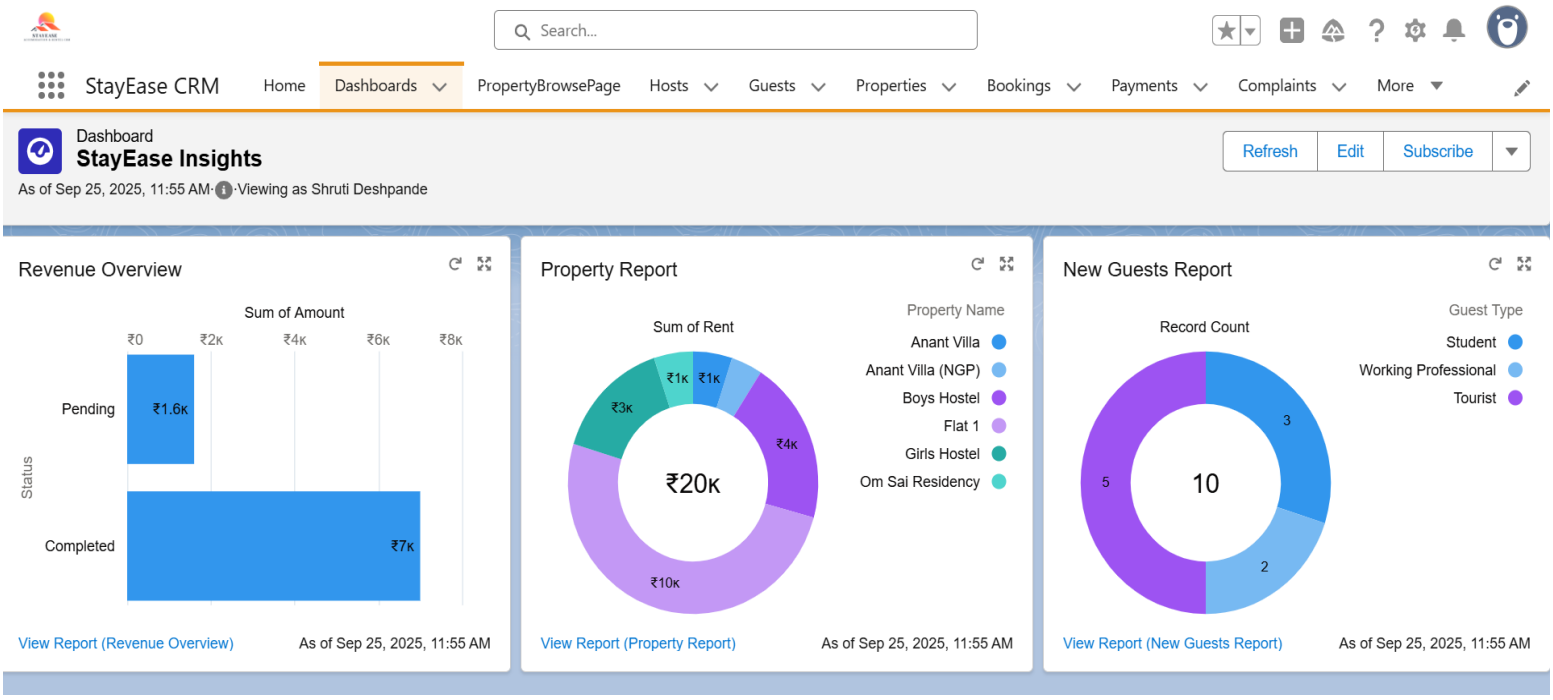
For demonstration purposes, a dummy API callout is integrated using Apex to simulate a payment gateway. To enable this securely, **Named Credentials** and **External Credentials** are configured in Salesforce. These act as a secure authentication layer for external API communication, ensuring that sensitive details like endpoint URLs and authentication methods are managed centrally and not exposed in Apex code.

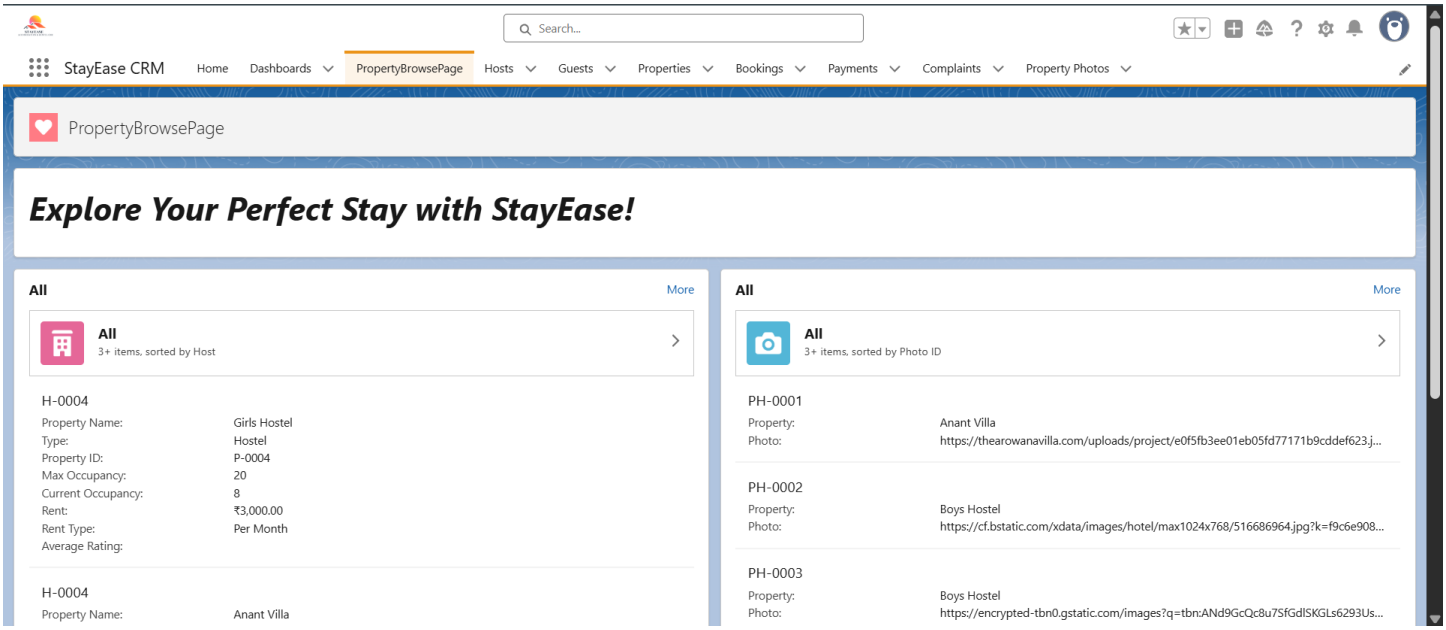

SETUP
Named Credentials
Named Credentials
External Credentials
External Auth Identity Providers
1 Items · Sorted by Label
New
Label
Type
URL
External Credential
Actions
DummyPaymentAPI
Secured Endpoint
https://jsonplaceholder.typicode.c...
DummyPaymentCredential

Phase (6-Advance)

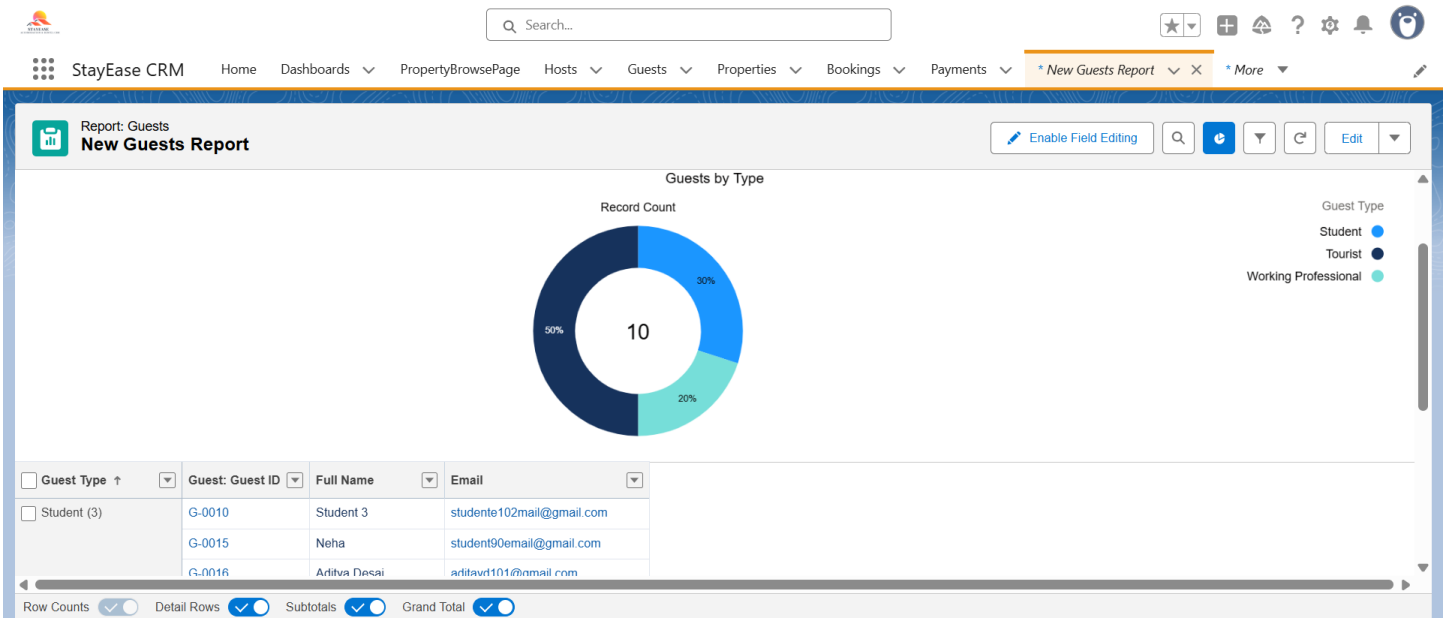
Phase 9: Reporting, Dashboards & Security Review

- **Objective**: Enhance the StayEase CRM with advanced analytics and insights by creating a comprehensive dashboard and integrating it into the user interface, while continuing efforts to incorporate the hostAnalyticsDashboard LWC for host-specific analytics.

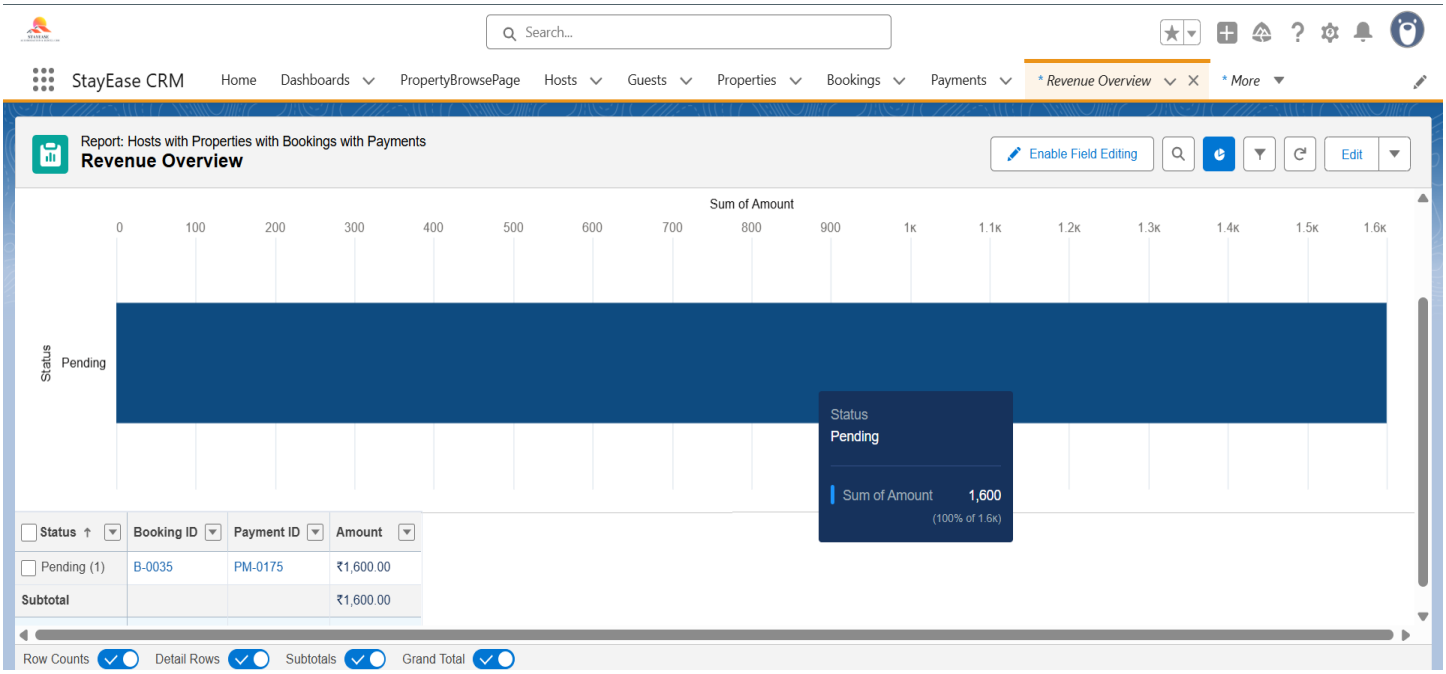




- **Activities**:
  - Designed and implemented the **StayEase Insights Dashboard** to provide a centralized view of key performance metrics, featuring:
    - **Guests by Type**: A visual breakdown categorizing guests (e.g., individual, family, corporate) based on data from the Guest__c object, utilizing custom reports and chart components.

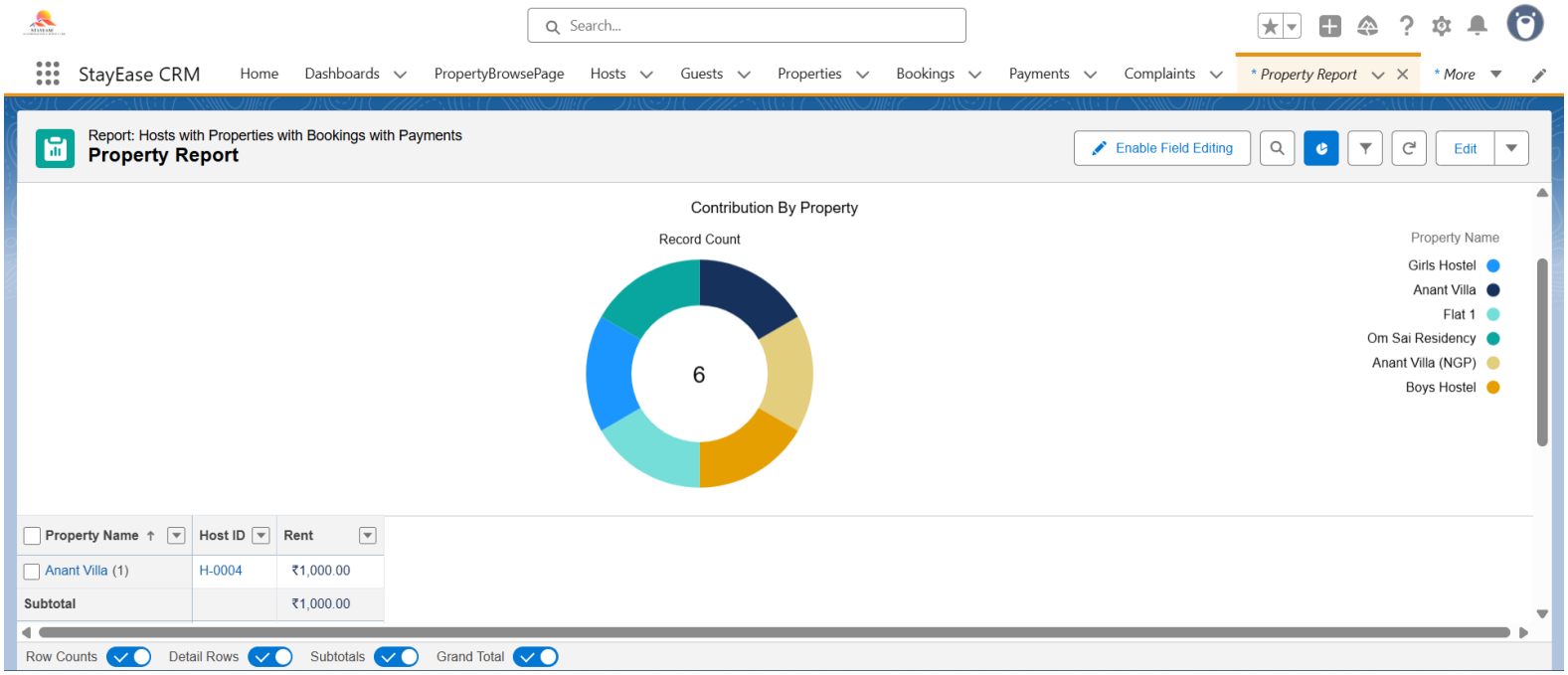


    - **Revenue KPIs**: Key performance indicators showcasing total revenue generated, derived from aggregated Total_Rent__c values across Booking__c records, displayed through gauge or metric charts.



- **Booking Trends**: A time-series analysis of booking activities (e.g., monthly or quarterly trends) using Booking__c data with CreatedDate and Status__c filters, presented in a line or bar chart.

- Integrated the StayEase Insights Dashboard into the **standard Lightning Home Page** via Lightning App Builder, ensuring it is accessible to all users upon logging into the org .
- Continued efforts to enhance host-specific analytics by attempting to add the hostAnalyticsDashboard LWC to the Host User record page, aiming to display total rent generated per property associated with each host.



- **Challenges:**

  - Encountered visibility issues with hostAnalyticsDashboard on the Host User page despite successful deployment potentially due to unconfigured page layout, missing test data (e.g., Property__c records with Host__c set and linked Booking__c records), or insufficient profile permissions.

➢ Future Enhancements

While the current implementation focuses on the core booking and payment functionalities, several enhancements are planned to further improve efficiency and user experience:

- **Rent Due Alerts** – Automated reminders to guests when their rent payment date is approaching, using scheduled flows or Salesforce notifications.
- **Recurring Payments** – Support for auto-debit or recurring payment setups for long-term stays, reducing manual intervention.
- **Refund Management** – A streamlined process to handle booking cancellations and partial or full refunds through the same flow.
- **Multi-Currency Support** – Extending payment records to handle transactions in multiple currencies for international guests.
- **Mobile-Friendly Payment Access** – Embedding the payment flow in a mobile app or community portal to allow guests to pay conveniently from anywhere.
- **Analytics Dashboard** – Enhanced dashboards for hosts and admins to track revenue trends, overdue payments, and collection efficiency.

These enhancements will make the system more robust, user-centric, and adaptable for future needs.

Phase 10: Final Presentation & Demo Day

• Demo Walkthrough : Demo Video

• Handoff Documentation : Google DOC - Link

• LinkedIn/Portfolio Project Showcase : LinkedIn