

# 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- **Unverified Listings** – Many platforms allow fake or outdated property details.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**

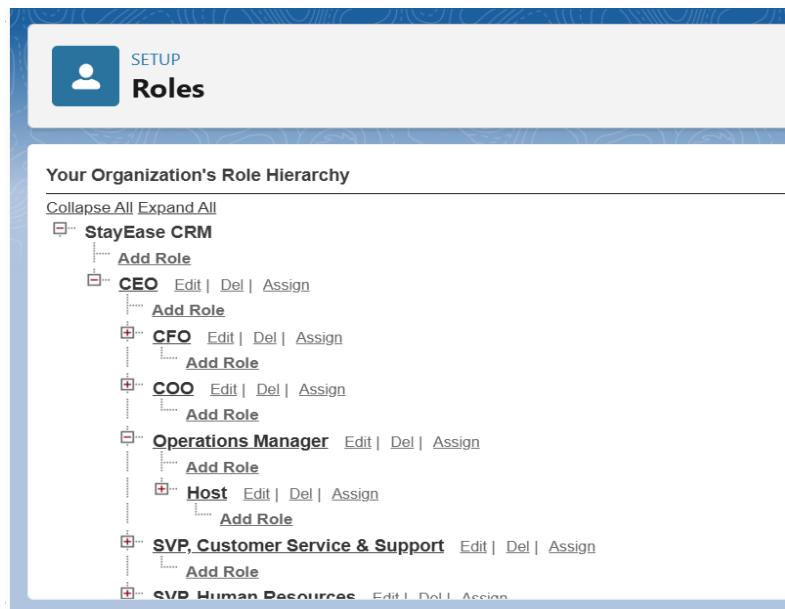
1. Create a **unified accommodation platform** with transparent workflows.
2. Support **short-term bookings** for tourists and **long-term rentals** for students/professionals.
3. Provide **role-based security** for data integrity.
4. Use Salesforce automation (Flows, Validation Rules, Approvals) to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

[Edit](#) | [Del](#) | ... [Host Profile](#)

Salesforce Platform



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

[Edit](#) | [Del](#) | ... [Guest Profile](#)

Salesforce Platform



- **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

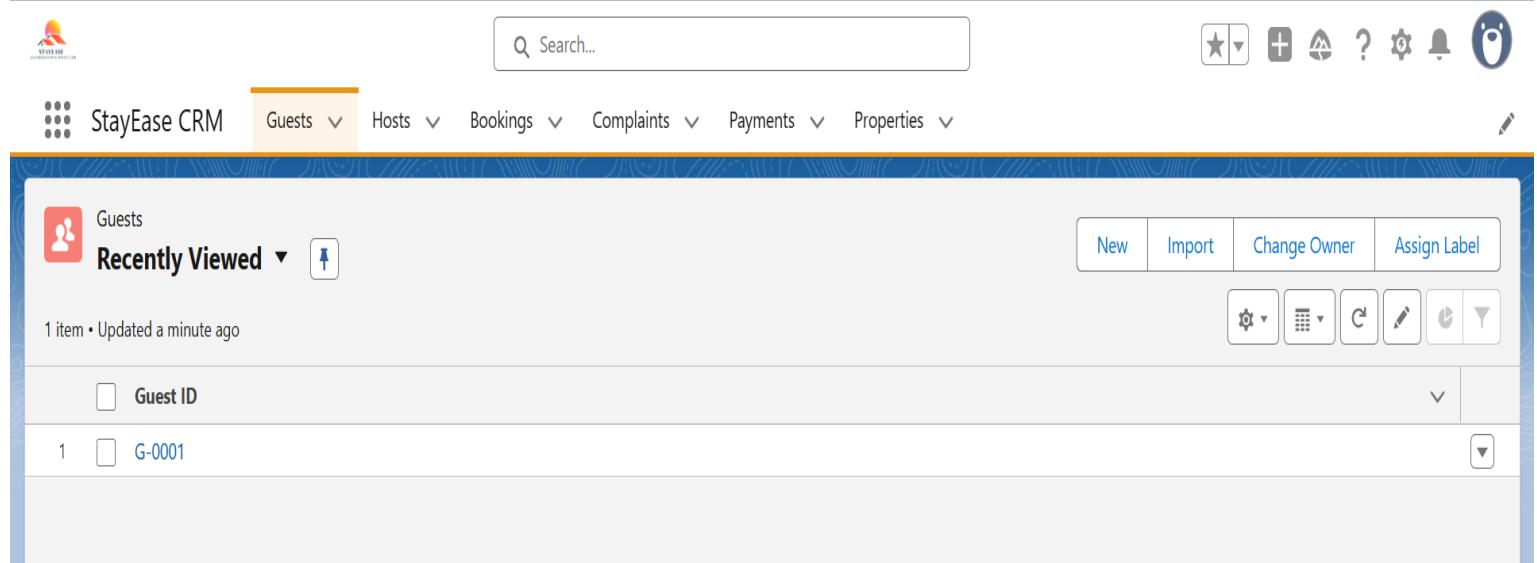
Additional flexibility beyond profiles:

- **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



StayEase CRM

Guests ▾ Hosts ▾ Bookings ▾ Complaints ▾ Payments ▾ Properties ▾

Guests

Recently Viewed ▾

1 item • Updated a minute ago

Guest ID

1 G-0001

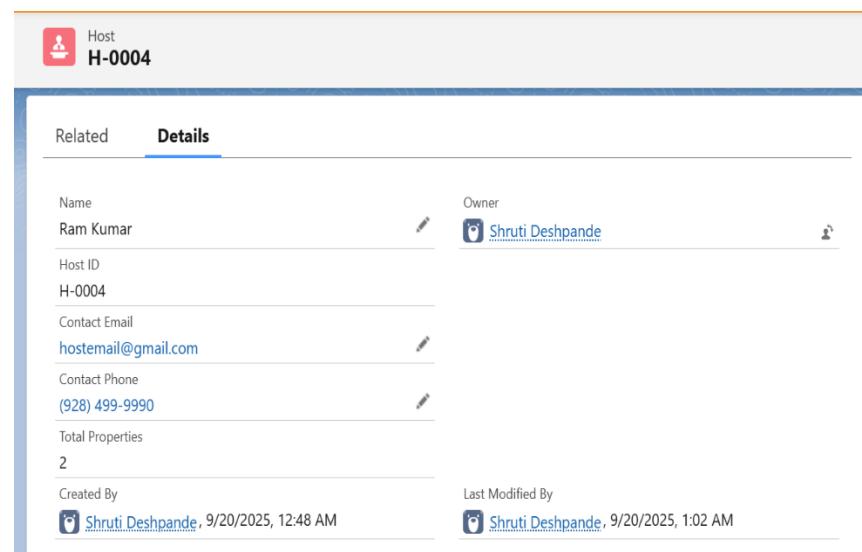
### 1. Host Object

**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



Host  
H-0004

Related Details

Name: Ram Kumar

Host ID: H-0004

Contact Email: hostemail@gmail.com

Contact Phone: (928) 499-9990

Total Properties: 2

Created By: Shruti Deshpande, 9/20/2025, 12:48 AM

Last Modified By: Shruti Deshpande, 9/20/2025, 1:02 AM

## 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:

The screenshot shows the Salesforce Object Manager interface for the 'Property' object. On the left, there is a sidebar with various configuration links: Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, and Record Types. The main area is titled 'Validation Rules' and shows a single item: 'Amount\_GreaterThan\_0'. The table has columns: RULE NAME, ERROR LOCATION, ERROR MESSAGE, ACTIVE, and MODIFIED BY. The row details are: Rule Name is 'Amount\_GreaterThan\_0', Error Location is 'Rent', Error Message is 'The rent amount should be greater than zero.', Active status is checked, and Modified By is 'Shruti Deshpande, 9/19/2025, 2:53 AM'. A 'New' button is located in the top right corner of the validation rules section.

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Amount_GreaterThan_0	Rent	The rent amount should be greater than zero.	✓	Shruti Deshpande, 9/19/2025, 2:53 AM

### 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).

Fields & Relationships 13 Items, Sorted by Field Label				
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD
	Advance Amount	Advance_Amount_c	Currency(16, 2)	
	Booking ID	Name	Auto Number	✓
	Booking Type	Booking_Type_c	Picklist	
	Check In Date	Check_In_Date_c	Date	
	Check Out Date	Check_Out_Date_c	Date	
	Created By	CreatedByld	Lookup(User)	
▼	Guest	Guest_c	Lookup(Guest)	✓

#### Relationship:

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

## Validation Rules:

The screenshot shows the Salesforce Setup interface for the Object Manager. The left sidebar lists various configuration options like Details, Fields & Relationships, Page Layouts, etc. The main content area is titled "Validation Rules" under the "Booking" object. It displays a table with five validation rules, each with a "New" button in the top right corner.

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Advance_amount_for_longTerm_stay	Top of Page	Advance Amount is mandatory for Student and Working Professional bookings	✓	Shruti Deshpande, 9/20/2025, 12:32 AM
Book_for_longterm_stay	Check Out Date	Check-Out Date is required for Tourist bookings	✓	Shruti Deshpande, 9/19/2025, 3:39 AM
CheckInBeforeCheckOut	Top of Page	Check-In date must be before Check-Out date.	✓	Shruti Deshpande, 9/17/2025, 8:15 AM
Number_of_Guests	Number of Guests	Number of guests should be 1 or more	✓	Shruti Deshpande, 9/19/2025, 2:57 AM
OccupancyLimit	Number of Guests	Number of guests exceeds property maximum occupancy.	✓	Shruti Deshpande, 9/17/2025, 8:16 AM

## 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).

The screenshot shows the Salesforce Setup interface for the Object Manager. The left sidebar lists various configuration options like Details, Fields & Relationships, Page Layouts, etc. The main content area is titled "Details" under the "Guest" object. It displays a table with fields and their descriptions, along with checkboxes for various settings.

Details	Details
Description	A guest is the one looking for rented placed . they can be students , travelers.
API Name	Guest__c
Custom	✓
Singular Label	Guest
Plural Label	Guests
Enable Reports	✓
Track Activities	✓
Track Field History	✓
Deployment Status	Deployed
Help Settings	Standard salesforce.com Help Window

### 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).

The screenshot shows the 'Payment' object setup page in the Salesforce Object Manager. The left sidebar lists standard object settings like Details, Fields & Relationships, Page Layouts, etc. The main 'Details' tab is selected, showing the API Name as 'Payment\_c', which is custom and checked. Singular and Plural labels are set to 'Payment' and 'Payments' respectively. A description states: 'This object has all the payment details of the bookings.' On the right, there are checkboxes for enabling reports, tracking activities, and field history, all of which are checked. Deployment status is set to 'Deployed'. Help settings point to the standard Salesforce help window.

## Relationship:

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

## Validation Rules:

The screenshot shows the 'Validation Rules' section for the Payment object. It displays a single rule named 'Transction\_Date\_Before\_Checkin'. The rule specifies that the Transaction Date cannot be after the Check-In Date of the Booking. The rule is active and was modified by Shruti Deshpande on September 18, 2025, at 8:55 AM.

Rule Name	Error Location	Error Message	Active	Modified By
Transction_Date_Before_Checkin	Transaction Date	Transaction Date cannot be after the Check-In Date of the Booking.	✓	Shruti Deshpande, 9/18/2025, 8:55 AM

## 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.

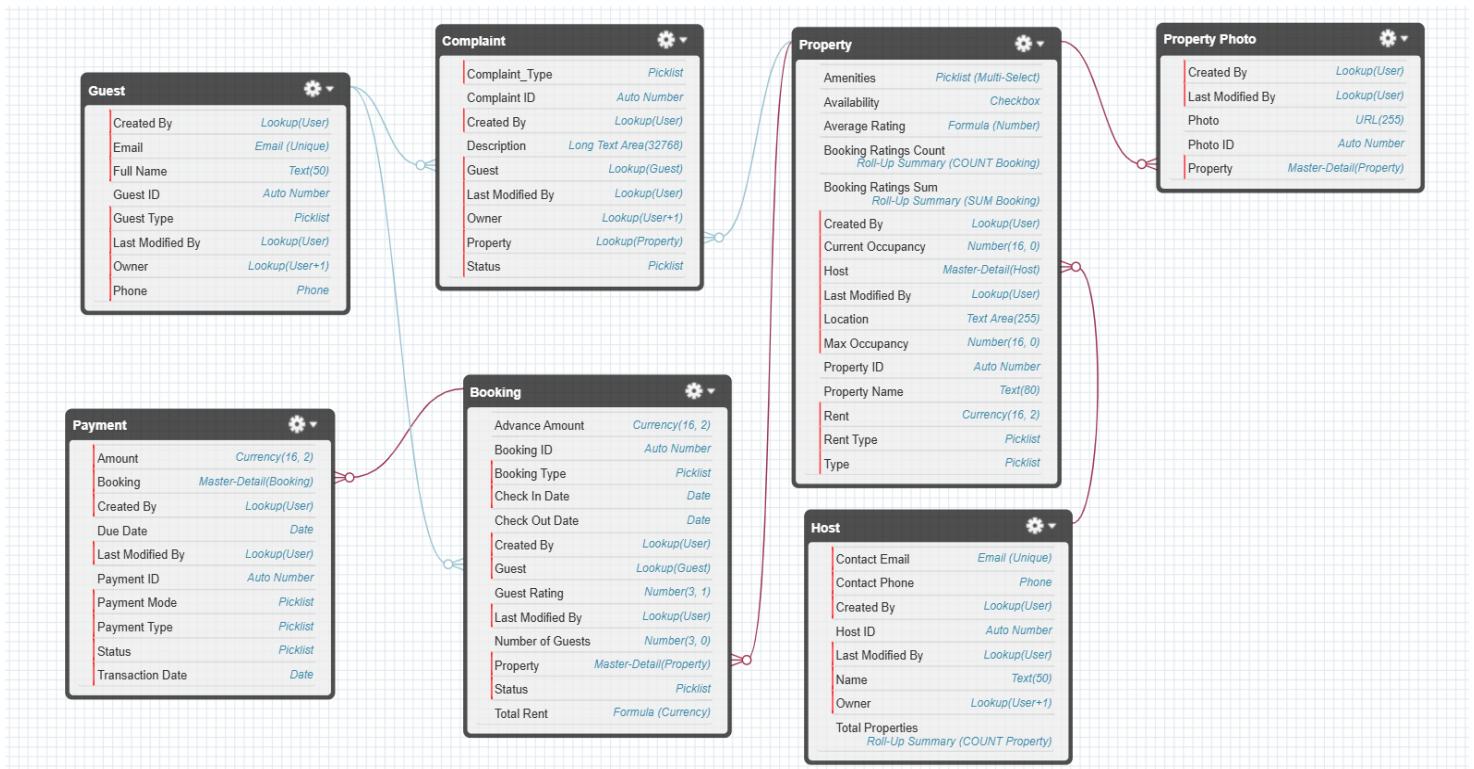


fig : Schema Diagram (Form Schema Builder)

# Phase 4 Process Automation (Admin)

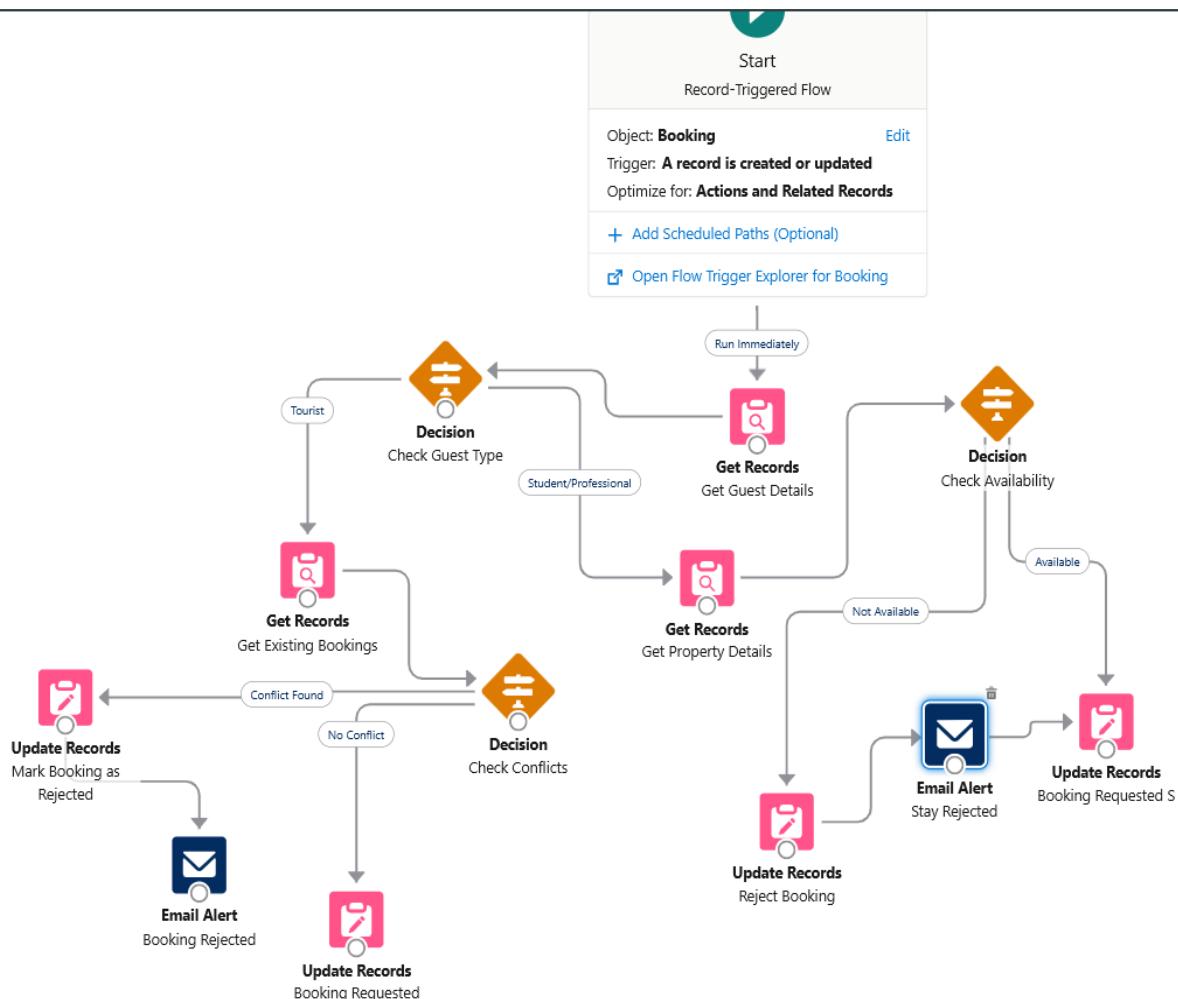
## Flow 1: Booking Availability Check

**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.

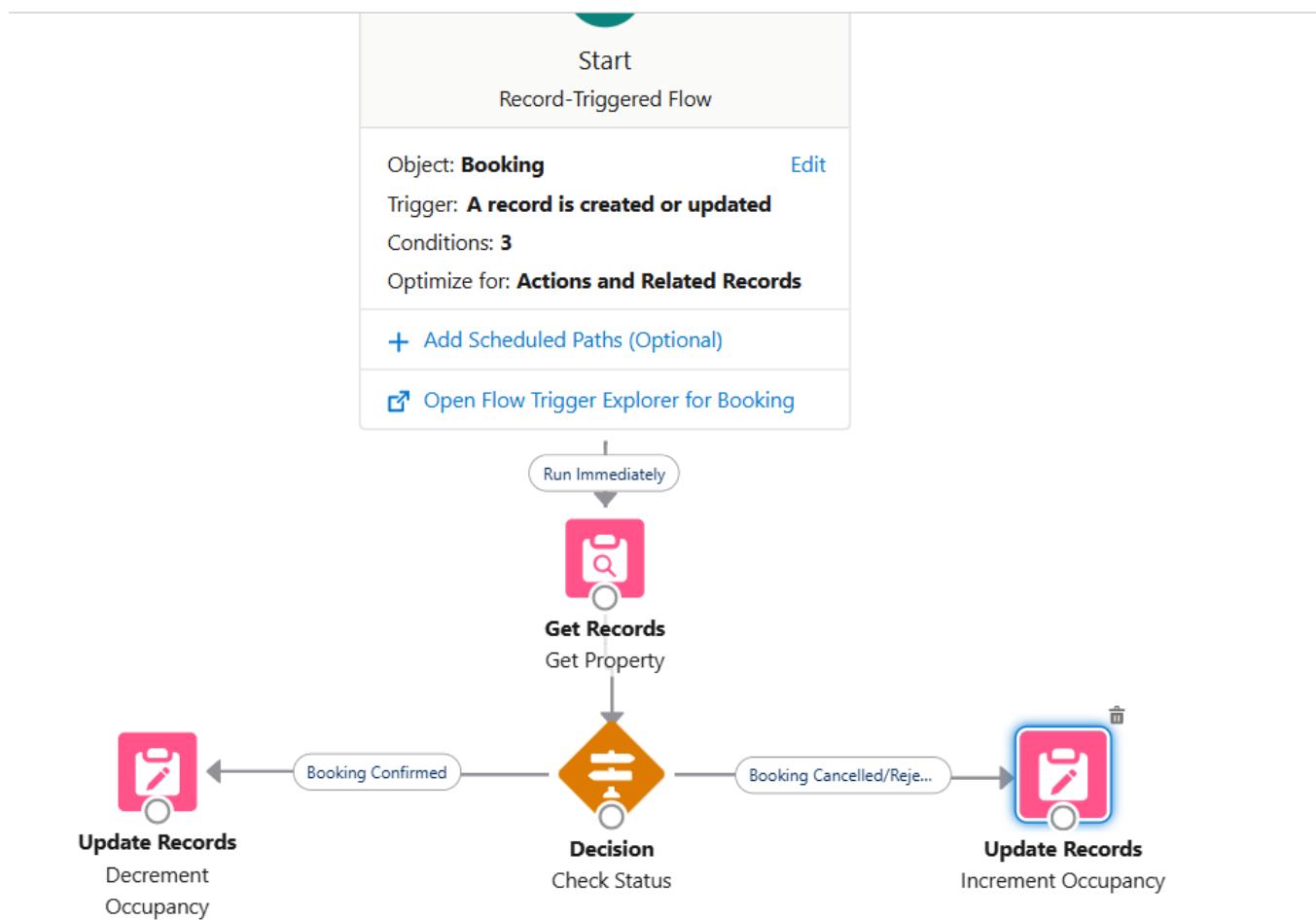


## Flow 2: Occupancy Management

**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.

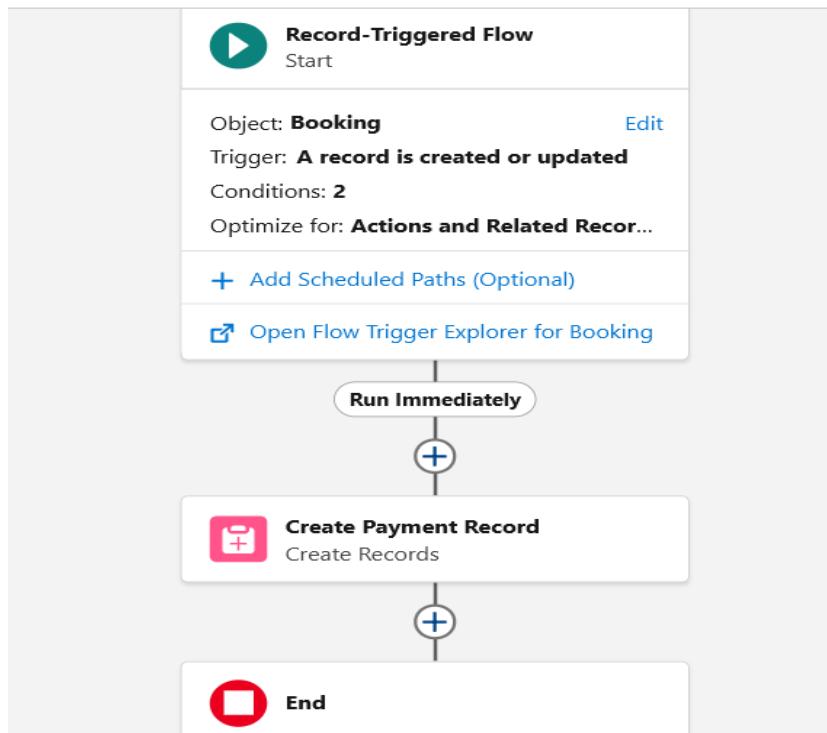


## Flow 3: Payment Creation & Notifications

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*.



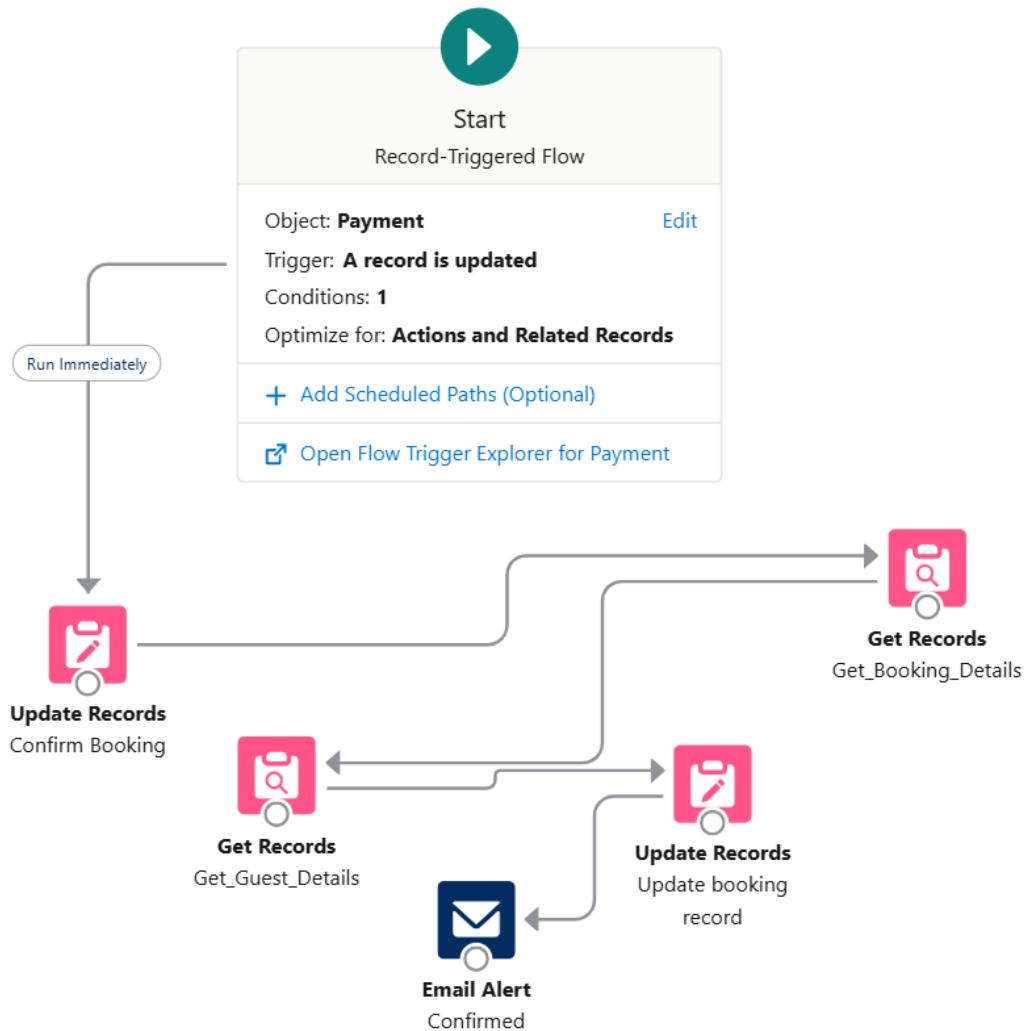
## Summary :

- 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.

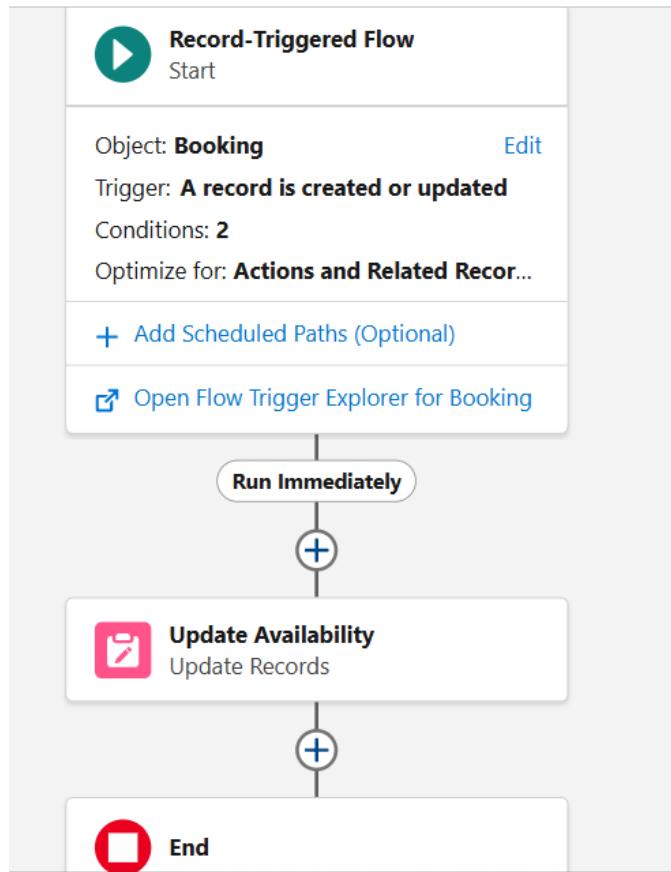
Upon updating a Payment\_\_c record with a status of 'Pending', this flow triggers. It first retrieves the associated Booking\_\_c record and then updates its status to 'Confirmed'. Subsequently, it fetches the Guest\_\_c details from the Booking\_\_c record and updates the Guest\_Email\_\_c field in the Booking\_\_c record. Finally, it sends an email alert using the Booking\_\_c.Email\_Alerts\_for\_Booking action.



- **Flow 5: Update 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.

The screenshot shows the Salesforce Lightning Components page under the SETUP tab. The main title is "Lightning Components". Below it, there's a sub-section titled "Lightning Components" with a "Help for this Page" link. A descriptive text explains that a Lightning component is a compact, configurable, and reusable element used for building apps and custom pages. It mentions two development models: Aura and Lightning Web Components (LWC). The page includes a "View" dropdown set to "All" and a "Create New View" button. At the top, there's a navigation bar with letters A-Z and a "All" button. A table lists the components:

Action	Name ↑	Label	Type	Namespace Prefix	Api Version
Del	hostAnalyticsDashboard	hostAnalyticsDashboard	LWC		64.0
Del	propertyWithPhotos	propertyWithPhotos	LWC		64.0

The screenshot shows the Salesforce Developer Console interface. The left sidebar contains navigation links such as EXPLORER, SOURCE CONTROL: REPOS..., OPEN EDITORS, STAYEASE-CRM, force-app\main\def..., lwc, hostAnalyticsDash..., propertyWithPhotos, jsconfig.json, OUTLINE, SOURCE CONTROL: GRAPH, and RUNNING TASKS. The main area displays the code for the LWC component `propertyWithPhotos.js`. The code imports LightningElement, api, and wire from 'lwc'. It also imports getRecord, getFieldValue, and getRelatedListRecords from 'lightning/uiRecordApi' and 'lightning/uiRelatedListApi' respectively. It defines constants `NAME_FIELD` and `HOST_FIELD` from '@salesforce/schema/Property\_\_c.Name' and '@salesforce/schema/Property\_\_c.Host\_\_c'. The component class `PropertyWithPhotos` extends LightningElement and includes properties `recordId`, `propertyName`, `hostName`, `photos`, and `error`. It uses @wire to call `getRecord` with fields `FIELDS` and handles errors by setting `propertyName` and `hostName` if successful, or setting `error` if there is an issue.

```
import { LightningElement, api, wire } from 'lwc';
import { getRecord, getFieldValue } from 'lightning/uiRecordApi';
import { getRelatedListRecords } from 'lightning/uiRelatedListApi';
import NAME_FIELD from '@salesforce/schema/Property__c.Name';
import HOST_FIELD from '@salesforce/schema/Property__c.Host__c';

const FIELDS = [NAME_FIELD, HOST_FIELD];
const RELATED_FIELDS = ['Property_Photo__c.Property__c', 'Property_Photo__c.Photo__c', 'Prop...'];

export default class PropertyWithPhotos extends LightningElement {
    @api recordId; // Property__c ID
    propertyName;
    hostName;
    photos = [];
    error;

    @wire(getRecord, { recordId: '$recordId', fields: FIELDS })
    property({ error, data }) {
        if (data) {
            this.propertyName = getFieldValue(data, NAME_FIELD);
            this.hostName = data.fields.Host__c.displayValue || 'N/A'; // Host name from Use...
        } else if (error) {
            this.error = error.message;
        }
    }
}
```

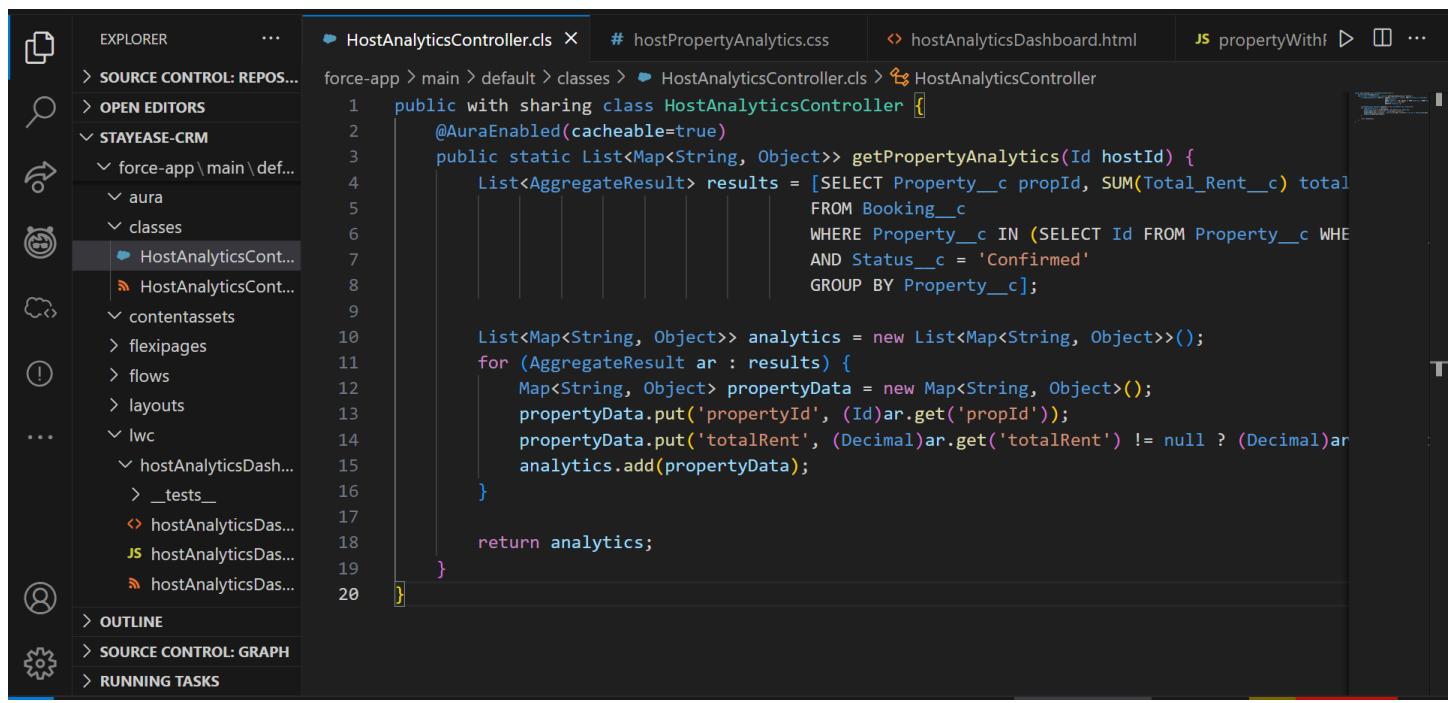
- ✓ **Outcome:** Successfully developed and partially deployed the propertyWithPhotos LWC. The component was prepared for integration into PropertyBrowsePage, with deployment issues addressed through CLI syntax correction. Further testing and page integration were planned for Phase 6.

## 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.



The screenshot shows the Salesforce Developer Console interface. On the left is the Explorer sidebar with various project components like SOURCE CONTROL, OPEN EDITORS, and the current file, HostAnalyticsController.cls. The main area displays the Apex code for the HostAnalyticsController class. The code implements a static method getPropertyAnalytics that performs an aggregate query on the Booking\_\_c object to calculate the total rent for each host. It then iterates over the results to build a map of property ID to total rent, which is returned as a list of maps.

```
public with sharing class HostAnalyticsController {
    @AuraEnabled(cacheable=true)
    public static List<Map<String, Object>> getPropertyAnalytics(Id hostId) {
        List<AggregateResult> results = [SELECT Property__c propId, SUM(Total_Rent__c) total
                                         FROM Booking__c
                                         WHERE Property__c IN (SELECT Id FROM Property__c WHERE Status__c = 'Confirmed'
                                         GROUP BY Property__c);

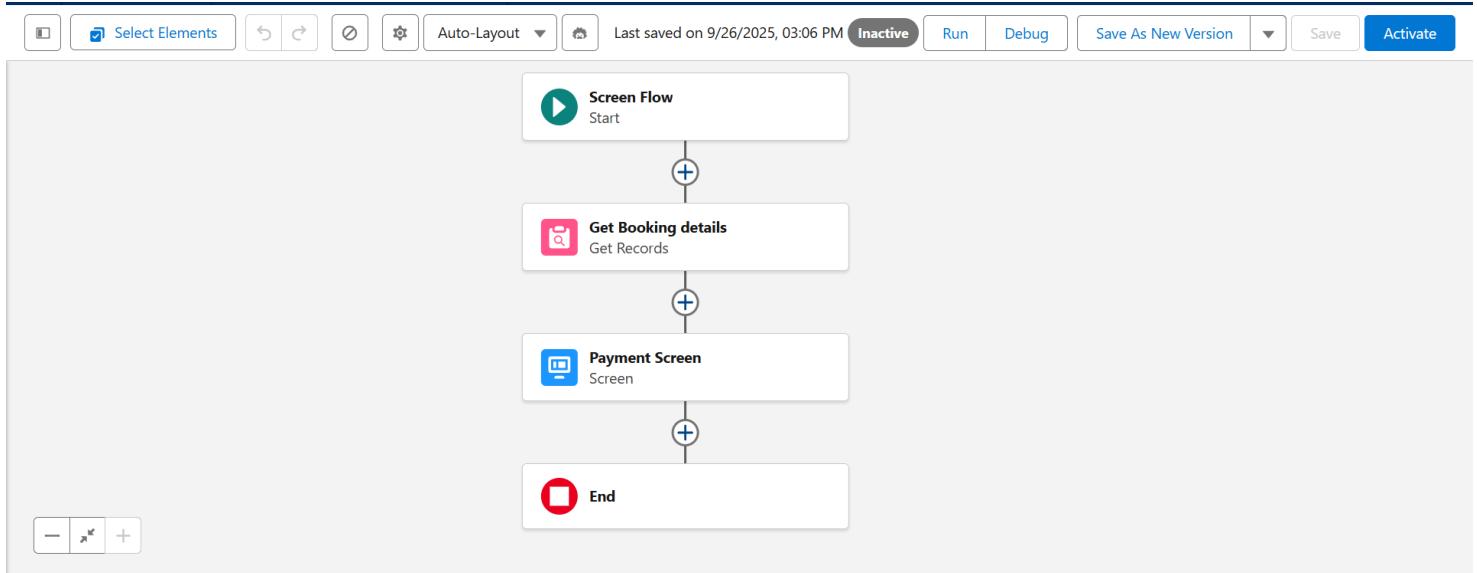
        List<Map<String, Object>> analytics = new List<Map<String, Object>>();
        for (AggregateResult ar : results) {
            Map<String, Object> propertyData = new Map<String, Object>();
            propertyData.put('propertyId', (Id)ar.get('propId'));
            propertyData.put('totalRent', (Decimal)ar.get('totalRent') != null ? (Decimal)ar.get('totalRent') : 0);
            analytics.add(propertyData);
        }
        return analytics;
    }
}
```

## 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.



Q Apex

Email

Apex Exception Email

Custom Code

Apex Classes

Apex Settings

Apex Test Execution

Apex Test History

Apex Triggers

Environments

Jobs

Apex Flex Queue

Apex Jobs

Didn't find what you're looking for?  
Try using Global Search.

## SETUP Apex Classes

Name	PaymentIntegration	Status	Active
Namespace Prefix		Code Coverage	0% (0/14)
Created By	Shruti Deshpande, 9/26/2025, 1:34 AM	Last Modified By	Shruti Deshpande, 9/26/2025, 2:03 AM

Class Body Class Summary Version Settings Trace Flags

```

1 public with sharing class PaymentIntegration {
2     @future(callout=true)
3     public static void processPayment(Id bookingId) {
4         // Query Booking with correct fields
5         Booking__c booking = [
6             SELECT Id, Name, Guest__r.Name, Advance_Amount__c, Total_Rent__c
7             FROM Booking__c
8             WHERE Id = :bookingId
9             LIMIT 1
10        ];
11
12        Http http = new Http();
13        HttpRequest req = new HttpRequest();
14        req.setEndpoint('callout:DummyPaymentAPI/posts'); // Named Credential
15        req.setMethod('POST');
16        req.setHeader('Content-Type', 'application/json');
17
18        // Choose amount based on booking type
19        Decimal amount = (booking.Advance_Amount__c != null) ? booking.Advance_Amount__c : booking.Total_Rent__c;
20
21        String body = '{"bookingId": "' + booking.Id +
22            '", "amount": "' + amount +
23            '", "guest": "' + booking.Guest__r.Name + '"}';
24        req.setBody(body);
25
26        HttpResponse res = http.send(req);
27        System.debug('Payment Response: ' + res.getBody());
28    }

```

SETUP Apex Triggers

Apex Trigger PaymentTrigger

Help for this Page

Apex Trigger Detail

Edit Delete Download Show Dependencies

Name	PaymentTrigger	sObject Type	Payment
Code Coverage	0% (0/2)	Status	Active
Created By	Shruti Deshpande, 9/26/2025, 1:35 AM	Last Modified By	Shruti Deshpande, 9/26/2025, 2:03 AM
Namespace Prefix			

Apex Trigger Version Settings Trace Flags

Edit Delete Download Show Dependencies

```

1 trigger PaymentTrigger on Payment__c (after insert) {
2     for(Payment__c p : Trigger.new){
3         PaymentIntegration.processPayment(p.Booking__c);
4     }
5 }

```

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

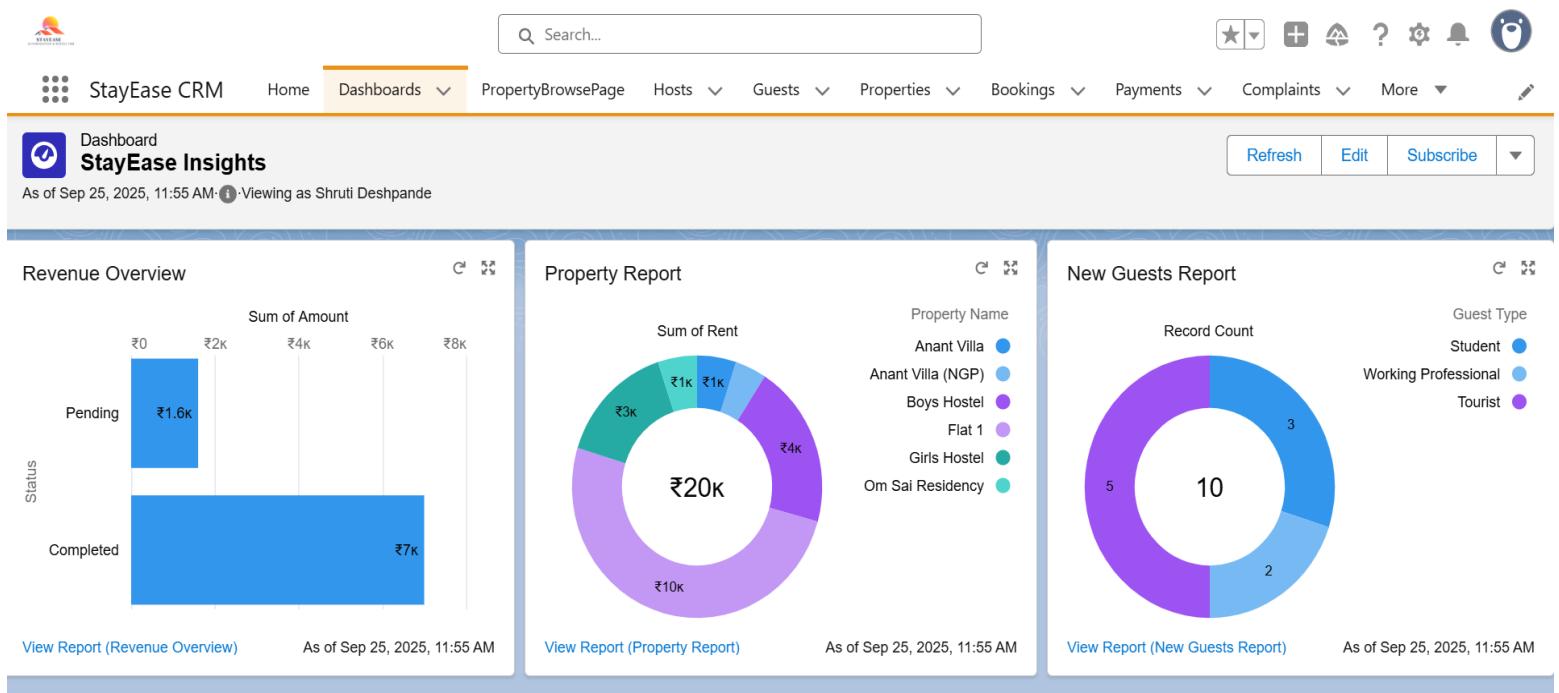


Label	Type	URL	External Credential	Actions
DummyPaymentAPI	Secured Endpoint	https://jsonplaceholder.typicode.c...	DummyPaymentCredential	

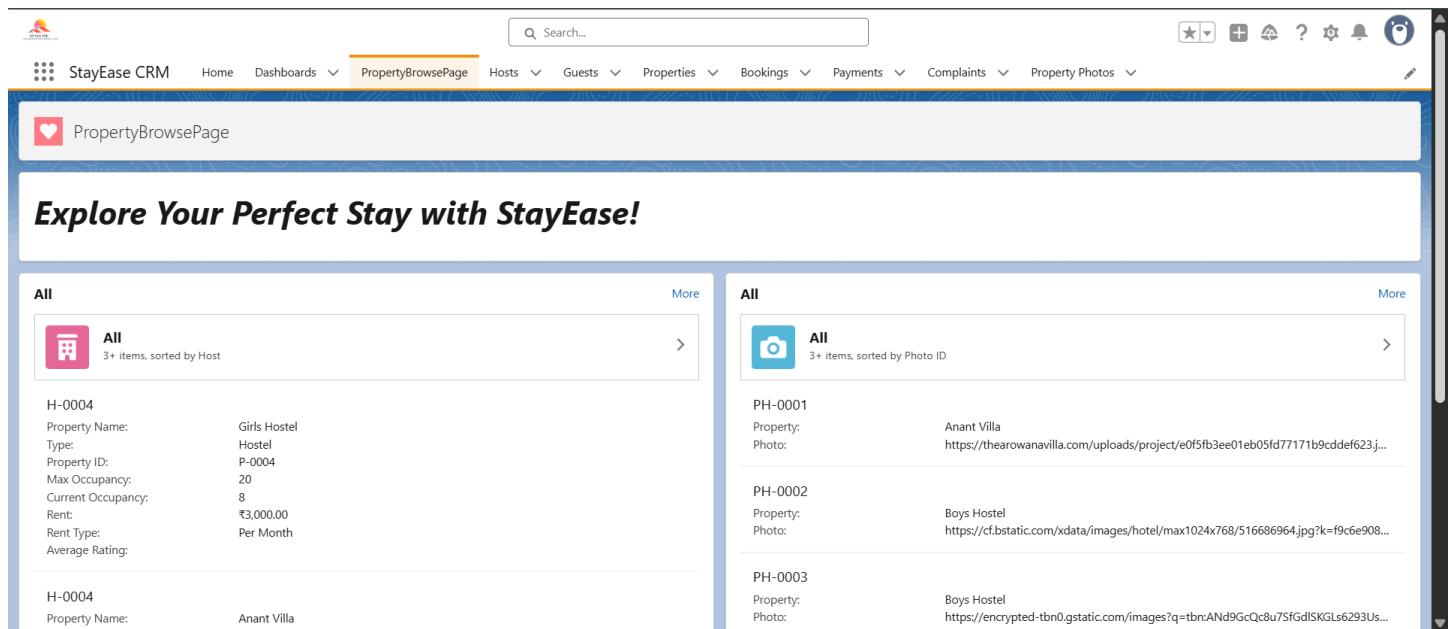
## Phase (6-Advance)

### 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.



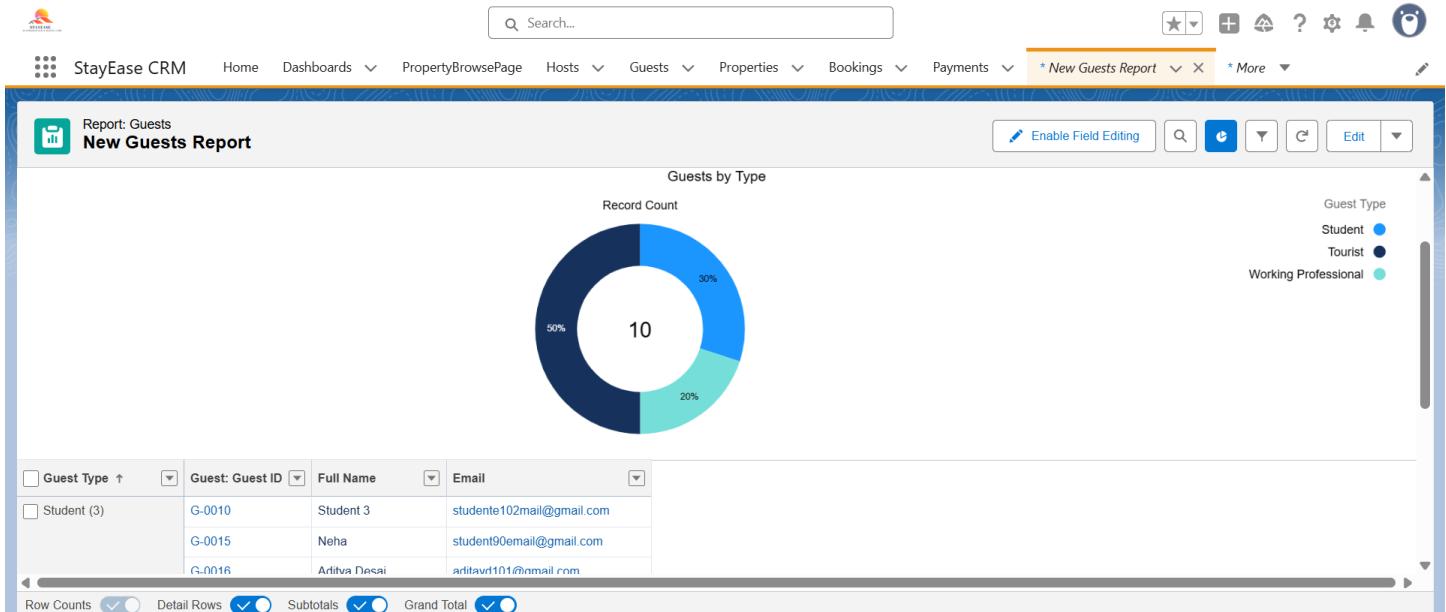
The screenshot shows the StayEase CRM dashboard. At the top, there's a search bar and a navigation bar with links like Home, Dashboards, PropertyBrowsePage, Hosts, Guests, Properties, Bookings, Payments, Complaints, More, Refresh, Edit, and Subscribe. The main area has three cards: 'Revenue Overview' (bar chart showing Pending and Completed amounts), 'Property Report' (donut chart showing rent distribution across properties), and 'New Guests Report' (donut chart showing guest count by type). Each card has a 'View Report' link at the bottom.



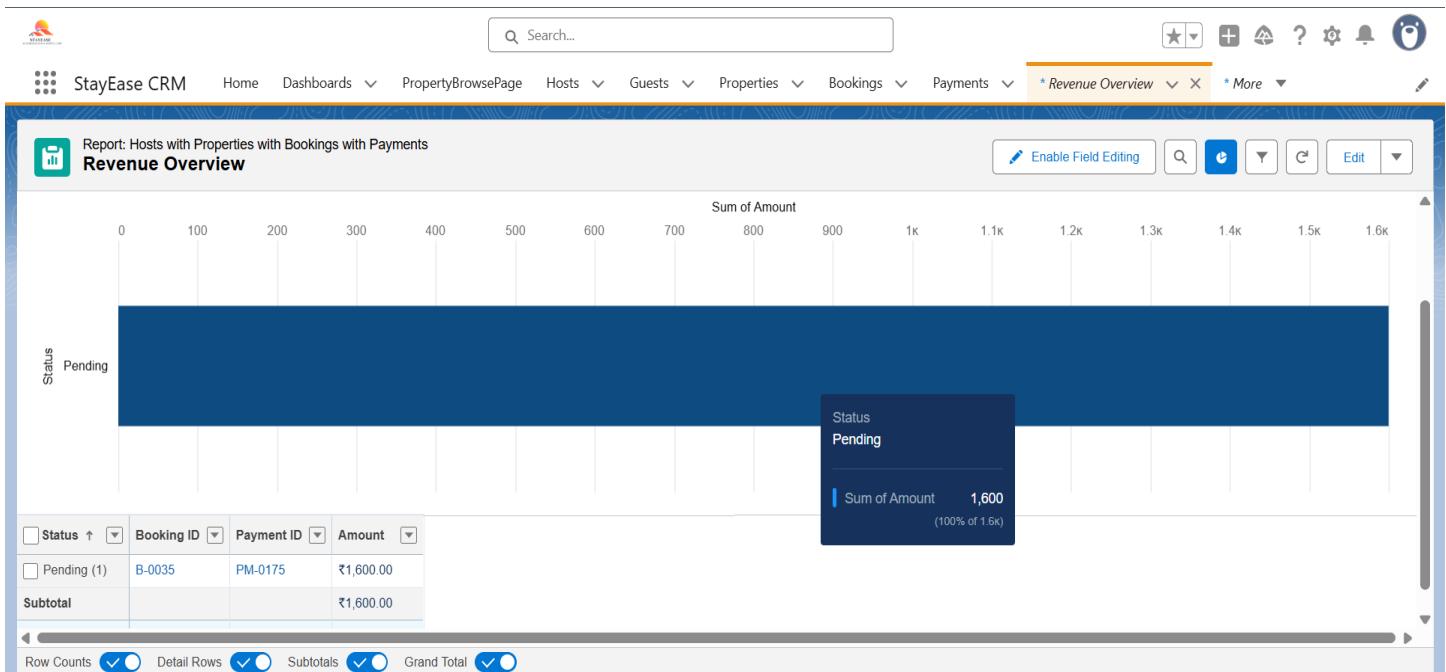
The screenshot shows the StayEase CRM PropertyBrowsePage. The top navigation bar includes a search bar and links for Home, Dashboards, PropertyBrowsePage, Hosts, Guests, Properties, Bookings, Payments, Complaints, and Property Photos. Below the navigation is a header with a heart icon and the text 'Explore Your Perfect Stay with StayEase!'. The main content area displays two columns of property cards. The left column shows a card for 'H-0004' (Girls Hostel) with details: Property Name: Girls Hostel, Type: Hostel, Property ID: P-0004, Max Occupancy: 20, Current Occupancy: 8, Rent: ₹3,000.00, Rent Type: Per Month, and Average Rating: Not specified. The right column shows cards for 'PH-0001' (Anant Villa), 'PH-0002' (Boys Hostel), and 'PH-0003' (Boys Hostel), each with a photo thumbnail and a link to the property's details.

- **Activities:**

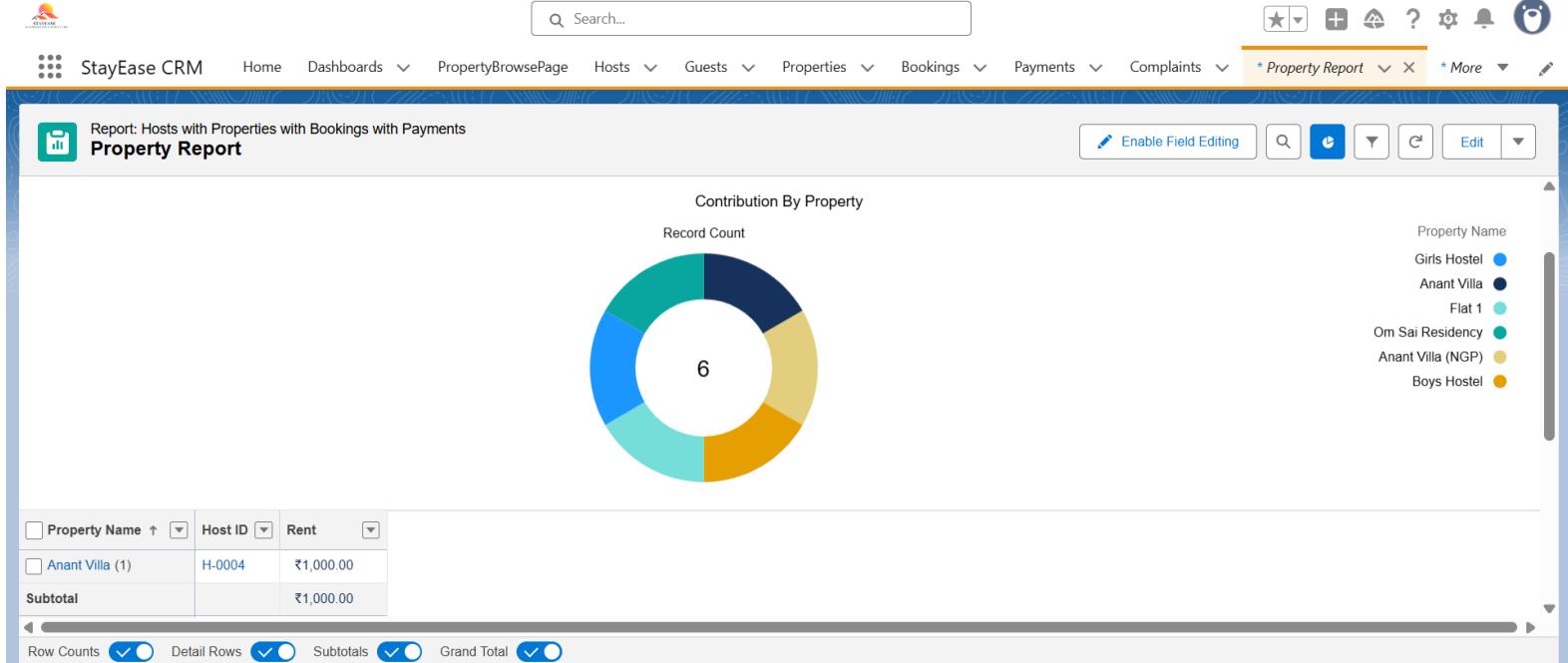
- Designed and implemented the **StayEase Insights Dashboard** to provide a centralized view of key performance metrics, featuring:
  - **Guests by Type:** A visual breakdown categorizing guests (e.g., individual, family, corporate) based on data from the Guest\_\_c object, utilizing custom reports and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