

# **Project Title: CampusStay – A Salesforce-Based Hostel Booking System**

---

## **1. Project Overview**

The **CampusStay** project is designed to help students book hostel rooms on a cot basis using Salesforce. The application streamlines the hostel booking process, ensuring students can easily reserve cots in either Boys' or Girls' hostels on campus. It also provides hostel managers with a tool to monitor bookings, manage room availability, and track payments. The system is built to ensure an efficient, user-friendly experience for both students and managers while automating much of the administrative process.

Through this project, we aim to enhance operational efficiency by automating the booking process, reducing manual effort, and minimizing errors in hostel room allocation and payment tracking.

---

## 2. Objectives

- **Business Goals:**
    - Simplify and automate the hostel booking process for students.
    - Ensure accurate tracking of available and booked cots in hostels.
    - Provide hostel managers with tools to manage availability and payments efficiently.
  - **Specific Outcomes:**
    - A complete booking system that allows students to book cots in on-campus hostels.
    - Automation of payment records once a booking is made.
    - An integrated system for hostel managers to track bookings and availability.
- 

## 3. Salesforce Key Features and Concepts Utilized

1. **Custom Objects:**
  - a. **College, Hostel/PG, Booking, Payment:** These custom objects were created to model the entities in the CampusStay application, such as colleges, hostels, bookings, and payments.
  - b. Each object has specific fields and relationships to capture the necessary data, such as cot availability, booking details, and payment status.
2. **Flows:**
  - a. **Screen Flow:** A visual, step-by-step process guides users through selecting a college, choosing a hostel type, checking cot availability, confirming booking details, and processing payments. Screens like "Booking Confirmation" and "Booking Successful" ensure the user gets feedback at each step.

- b. **Get Hostel Availability:** A record retrieval step in the flow fetches the available cots based on the selected hostel and filters the data by both the **college** and **hostel type**.
- c. **Conditional Logic:** The decision element checks whether enough cots are available and prevents overbooking by showing an error screen if there are not enough cots.

### 3. Apex Class and Triggers:

- a. **CotBookingHandler:** An Apex class that validates the cot availability during bookings, ensuring that users cannot book more cots than are available.
- b. **BookingTrigger:** A trigger that calls the class to run this validation before any new booking is inserted or updated.

### 4. Formula Fields and Roll-Up Summary:

- a. **Available Cots:** A formula field dynamically calculates how many cots are still available by subtracting the booked cots from the total cots.
- b. **Booked Cots:** A roll-up summary field in the **Hostel/PG** object automatically sums up the cots booked from the related **Booking** records, providing real-time data on availability.

### 5. Picklists:

- a. **Picklist Fields:** Used in key fields like **Hostel Type** to streamline data entry and prevent errors.

### 6. Lookup Relationships:

- a. Relationships between objects like **College ↔ Hostel**, **Hostel ↔ Booking**, and **Booking ↔ Payment** are managed through lookup fields, which make navigation and reporting easier while ensuring data integrity.

### 7. Validation Rules:

- a. Automatically enforce business logic, such as ensuring that the booking dates, cot availability, and booking details meet certain criteria before being processed.

### 8. Profiles and Role Hierarchy:

- a. **Profiles:** Control access to specific objects and fields for different users, ensuring that students, hostel managers, and admins have the appropriate permissions.
- b. **Roles:** Define the hierarchy within the system, for example, **Admin** (full control), **Hostel Manager** (can manage hostels and bookings), and **Student** (can only create bookings).

#### 9. **Formula Fields for Payment:**

- a. The **Total Cost** and **Total Amount** fields automatically calculate the final payment amount by multiplying the number of cots booked by the cost per cot. This automation simplifies the payment process for users.
- 

## 4. Detailed Steps to Solution Design

### **Objects and Fields :-**

1. **College**
2. **Hostel/PG**
3. **Booking**
4. **Payment**

## 1. College (Custom Object)

The screenshot shows the CampusStay application interface. At the top, there is a navigation bar with links for Home, Dashboards, Reports, Colleges, Bookings, Hostel/PGs, Payments, and Flows. Below the navigation bar is a search bar labeled "Search...". The main content area displays a list of colleges under the heading "Colleges". A dropdown menu shows "All" selected. There are six items listed, each with a checkbox and a "View" button:

- 1 COEP
- 2 DYP
- 3 MIT
- 4 PCCOE
- 5 VITI
- 6 WIT

At the top right of the list area, there are buttons for New, Import, Change Owner, Printable View, and Assign Label. Below the list, there are more standard UI elements like a refresh icon, a list icon, a search icon, and a settings icon.

The screenshot shows the detail view for a college record named "DYP". The top navigation bar and search bar are identical to the previous screenshot. The main content area has a header "College DYP". Below the header, there are two tabs: "Related" and "Details". The "Details" tab is active, showing the following fields:

College Name	DYP
Location	Akurdi
Created By	Rohan Ekbote, 05/10/2024, 11:33 pm
Last Modified By	Rohan Ekbote, 18/10/2024, 4:15 pm

Next to the details, there is an "Owner" field with a user icon and the name "Rohan Ekbote". On the right side of the screen, there is a "Activity" section with a header "Upcoming & Overdue". It includes a toolbar with icons for calendar, list, message, and email, and a filter section. The activity list is currently empty, displaying the message "No activities to show. Get started by sending an email, scheduling a task, and more.".

**College Name :** The name of the college (e.g., DYP , PCCOE , MIT , COEP ).  
**Location (Text):** The physical location of the college.

**Relationships:**

**One-to-Many:** A college can have multiple hostels.

## 2. Hostel/PG (Custom Object)

This screenshot shows the 'Hostel/PGs' list view in the CampusStay application. The page title is 'Hostel/PGs' and it displays 10 items sorted by Hostel/PG Name. The items listed are:

- 1 COEP Boys
- 2 COEP Girls
- 3 DYP Boys
- 4 DYP Girls
- 5 MIT Boys
- 6 MIT Girls
- 7 VJTI Boys
- 8 VJTI Girls
- 9 WIT Boys
- 10 WIT Girls

The interface includes a search bar, filter buttons (All), and standard list-view controls like New, Import, Change Owner, Printable View, and Assign Label.

This screenshot shows the detail view for the 'VJTI Boys' Hostel/PG record. The page title is 'VJTI Boys'. The 'Details' tab is selected, showing the following fields:

Hostel/PG Name	VJTI Boys
Manager Name	Dharmesh
College	VJTI
Hostel Address	(empty)
Hostel Type	Boys Hostel
Created By	Rohan Ekbote, 21/10/2024, 3:59 pm
Total Cots	100
Cost per Cot	₹10,000
Booked Cots	10
Available Cots	90.00
Owner	Rohan Ekbote
Last Modified By	Rohan Ekbote, 21/10/2024, 4:09 pm

The 'Activity' sidebar shows no upcoming or overdue activities.

- 2.1 Hostel/PG Name : The name of the hostel (e.g., Boys Hostel or Girls Hostel).
- 2.2 College (Lookup to College): Links the hostel to a specific college.
- 2.3 Total Cots (Number): The total number of cots available in the hostel.
- 2.4 Available Cots (Formula, Number): Calculated as Total Cots\_c - Booked Cots\_c. Shows how many cots are still available for booking.
- 2.5 Cost per Cot (Currency): The price per cot for booking.
- 2.6 Booked Cots (Roll-Up Summary, Number): Automatically sums the total booked cots from related Booking records. This prevents overbooking.
- 2.7 Manager Name (Text): The manager responsible for the hostel.

2.8 Hostel Address (Text): The address or location of the hostel.

Relationships:

One-to-Many: Each hostel is linked to a college, and it can have multiple bookings.

### 3. Booking (Custom Object)

The screenshot shows two screenshots of the Lightning Platform interface for the 'CampusStay' application.

**Screenshot 1: Bookings List View**

This screenshot shows the 'Bookings' list view. The URL is [https://smartinternz-d7-dev-ed.develop.lightning.force.com/lightning/o/Booking\\_\\_c/list?filterName=00BdL00000H4iFCUAZ](https://smartinternz-d7-dev-ed.develop.lightning.force.com/lightning/o/Booking__c/list?filterName=00BdL00000H4iFCUAZ). The page displays a table of bookings with columns: Booking Name, Hostel, Hostel Type, Cots Booked, Booking Date, and Total Cost. The data is as follows:

Booking Name	Hostel	Hostel Type	Cots Booked	Booking Date	Total Cost
Anil	VITI Boys	Boys Hostel	10	21/10/2024	₹1,00,000.00
Cinna					
Rohan					
Shivam					

**Screenshot 2: Single Booking Detail View**

This screenshot shows the detail view for a specific booking. The URL is [https://smartinternz-d7-dev-ed.develop.lightning.force.com/lightning/r/Booking\\_\\_c/a02dL000006LjR0OAK/view](https://smartinternz-d7-dev-ed.develop.lightning.force.com/lightning/r/Booking__c/a02dL000006LjR0OAK/view). The page displays the following details:

Booking Name	Hostel
Shivam	VITI Boys

Other fields shown include:

- Student Name: Shivam
- College: VITI
- Arrival Date: 22/10/2024
- Departure Date: 31/01/2025
- Room Preferences: 1st Floor and Near Window
- Created By: Rohan Ekbote, 21/10/2024, 4:09 pm
- Last Modified By: Rohan Ekbote, 21/10/2024, 4:09 pm

A separate section titled 'Payments (1)' shows one payment entry:

Payment Name
Shivam

#### 3.1 Booking Name

#### 3.2 Student Name

3.3 College (Lookup to College): The selected college for the booking.

3.4 Hostel Type (Picklist): Two options - Boys Hostel and Girls Hostel.

3.5 Hostel (Lookup to Hostel/PG): Automatically filled based on the selected hostel type.

3.6 Cots Booked (Number): The number of cots the student wants to book.

3.7 Total Cost (Formula, Currency): Booked Cots\_c \* Hostel.Cost per Cot\_c.

Calculates the total cost based on the number of cots booked.

3.8 Booking Date (Date): The date when the booking is made.

3.9 Arrival Date (Date): The date the student plans to arrive at the hostel.

3.10 Departure Date (Date): The date the student plans to vacate the hostel.

3.11 Room Preferences (Long Text Area): Any specific requests (e.g., cot near a window, first floor).

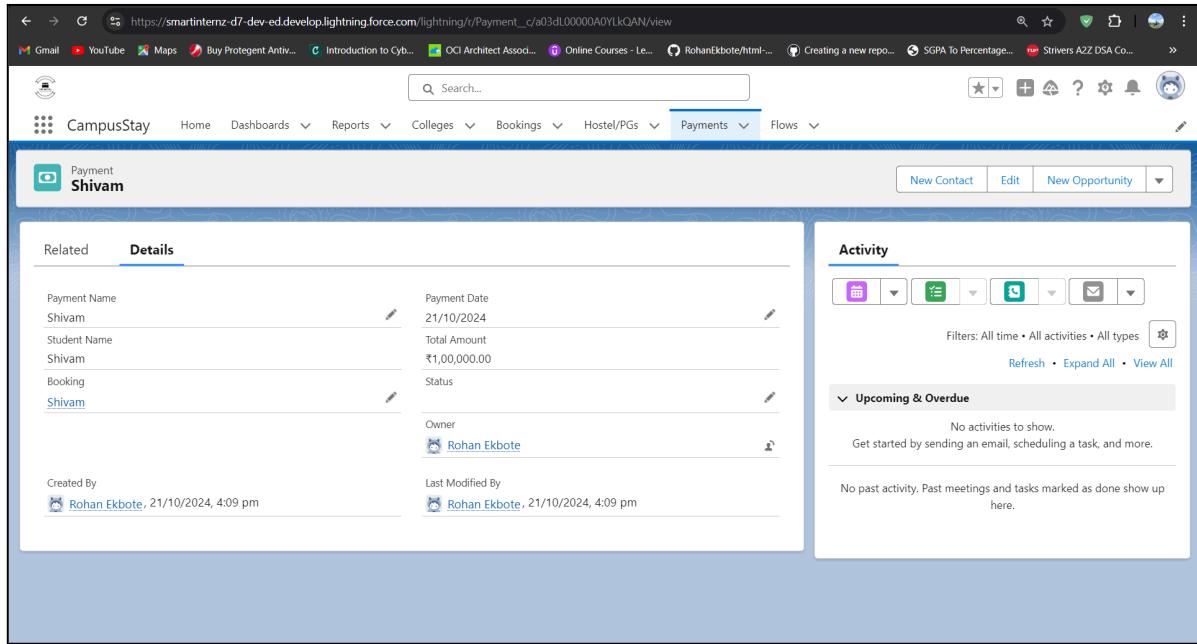
Relationships:

- Many-to-One: Each booking links to a hostel and student.
- Roll-Up Summary in Hostel object for managing cot availability.

## 4. Payment (Custom Object)

The screenshot shows the Salesforce Lightning interface for the 'Payments' custom object. The top navigation bar includes links for Gmail, YouTube, Maps, Buy Protectant Anti..., Introduction to Cyb..., OCI Architect Assoc..., Online Courses - Le..., RohanEkbote/html..., Creating a new repo..., SGPA To Percentage..., Strivers A2Z DSA Co..., and a GitHub icon. The main menu bar has items: CampusStay, Home, Dashboards, Reports, Colleges, Bookings, Hostel/PGs, Payments (selected), and Flows. The Payments page displays a list of 3 items, sorted by Payment Name (Cinna, Rohan, Shivam). The interface includes standard Salesforce buttons like New, Import, Change Owner, Printable View, and Assign Label, along with various filter and search tools.

Payment Name	Actions
Cinna	[Edit]
Rohan	[Edit]
Shivam	[Edit]



#### 4.1 Payment Name

4.2 Booking (Lookup to Booking): Links the payment to the booking.

4.3 Student Name (Formula, Text from Booking): Auto-filled from the booking's Student Name.

4.4 Total Amount (Formula, Currency): Same as Total Cost from the Booking object.

4.5 Payment Date (Date): The date when the payment is made.

4.6 Status (Picklist): Status options like "Pending", "Paid", etc.

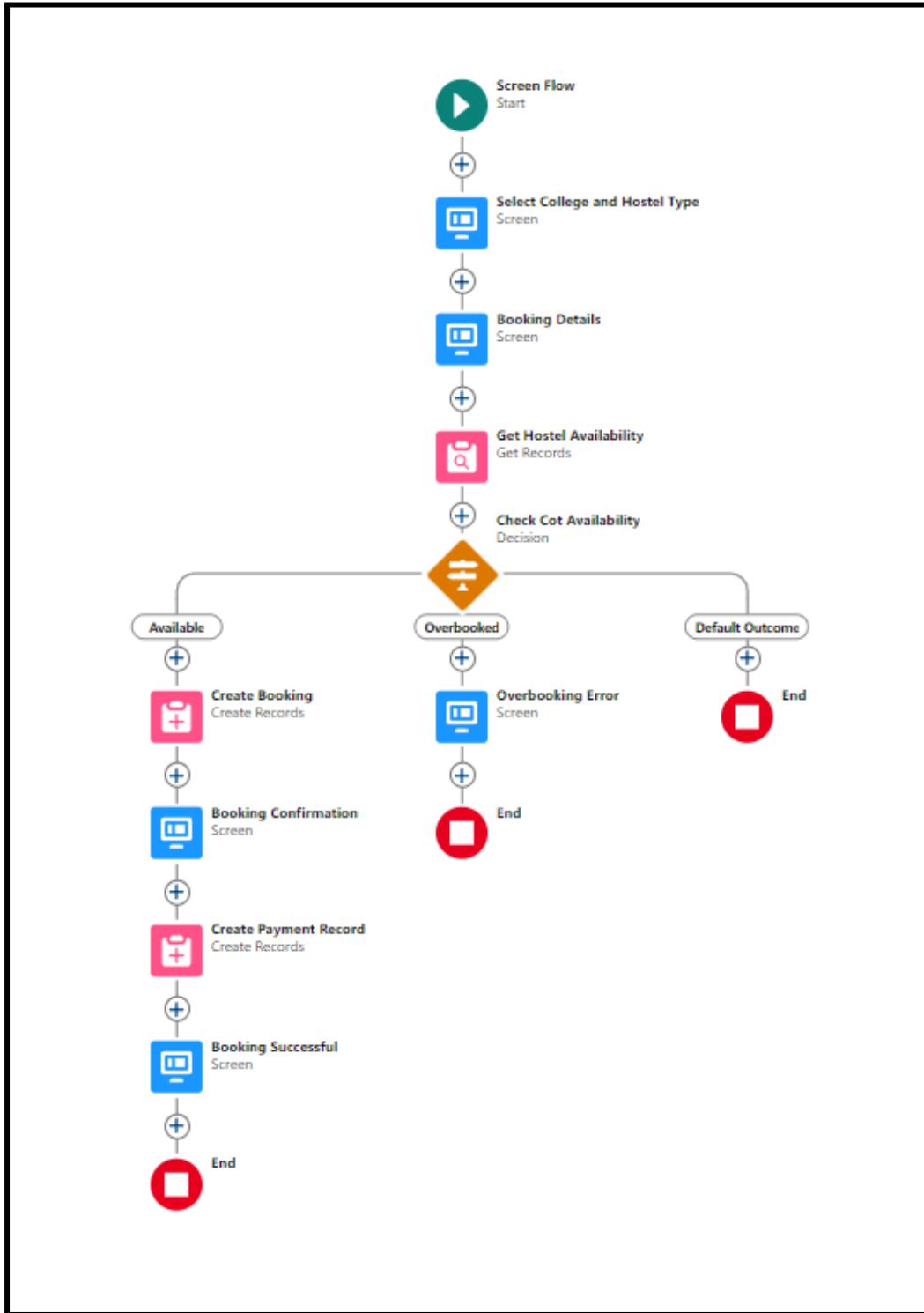
Relationships:

- One-to-One: Each payment is linked to one booking.
- The total amount is calculated based on the booking.

### Overall Relationships Overview of Objects:

- College ↔ Hostel/PG (One-to-Many):** A college has multiple hostels (Boys and Girls).
- Hostel/PG ↔ Booking (One-to-Many):** A hostel has multiple bookings, but the availability of cots is managed through the roll-up summary on Hostel/PG.
- Booking ↔ Payment (One-to-One):** Each booking has one associated payment.

## 5. Flow Design Overview



The CampusStay application uses a Salesforce **Screen Flow** to manage the booking process:

1. **Select College and Hostel Type Screen:** The user selects a college and either Boys' Hostel or Girls' Hostel. This selection drives further filtering.

- **Get Hostel Availability Element:** This element uses the **AND filter** to retrieve available cots based on the selected college and hostel type (filtered from the first screen). It fetches all hostels from the selected college and filters them by the hostel type (Boys or Girls).
2. **Booking Details Screen:** Captures specific booking details, including the number of cots, arrival and departure dates, and room preferences.
  3. **Get Hostel Availability:** It filters and selects the required Hostel.
  4. **Check Cot Availability (Decision Element):**
    - **If Available:** Proceeds to create a booking and shows a confirmation screen with total cost details.
    - **If Overbooked:** Displays an error message, informing the user that the requested number of cots is unavailable.
  5. **Create Booking and Payment Record:**
    - If cots are available, the booking is created, followed by the creation of the payment record.
  6. **Booking Confirmation Screen:** The user sees a screen with booking and payment details, including cost per cot and total amount, before final confirmation.
  7. **Booking Successful Screen:** Displays a success message confirming that the booking has been made, and payment details are auto-generated.

## 1. Select College and Hostel Type Screen

Campus2 V1

\*Colleges  
DYP

\*Hostel Type  
Boys Hostel

**Next**

Last saved on 21/10/2024, 11:47 pm **Active** Run Debug Save As New Version Save

**Edit Screen**

**Components** Fields

Search components...

**Campus2 V1**

\*Colleges  
--None--

\*Hostel Type  
--None--

Pause Previous Next

**Screen Properties**

\* Label

\* API Name

Description

> Configure Header

> Configure Footer

**Available** **Overbooked** **Default Outcome**

**Cancel** **Done**

## 2. Booking Details Screen

Campus2 V1

\* Student Name  
Rohan Ekbote

\* Hostel Name  
DYP Boys

\* Cots Booked  
3

\* Arrival Date  
23-Oct-2024

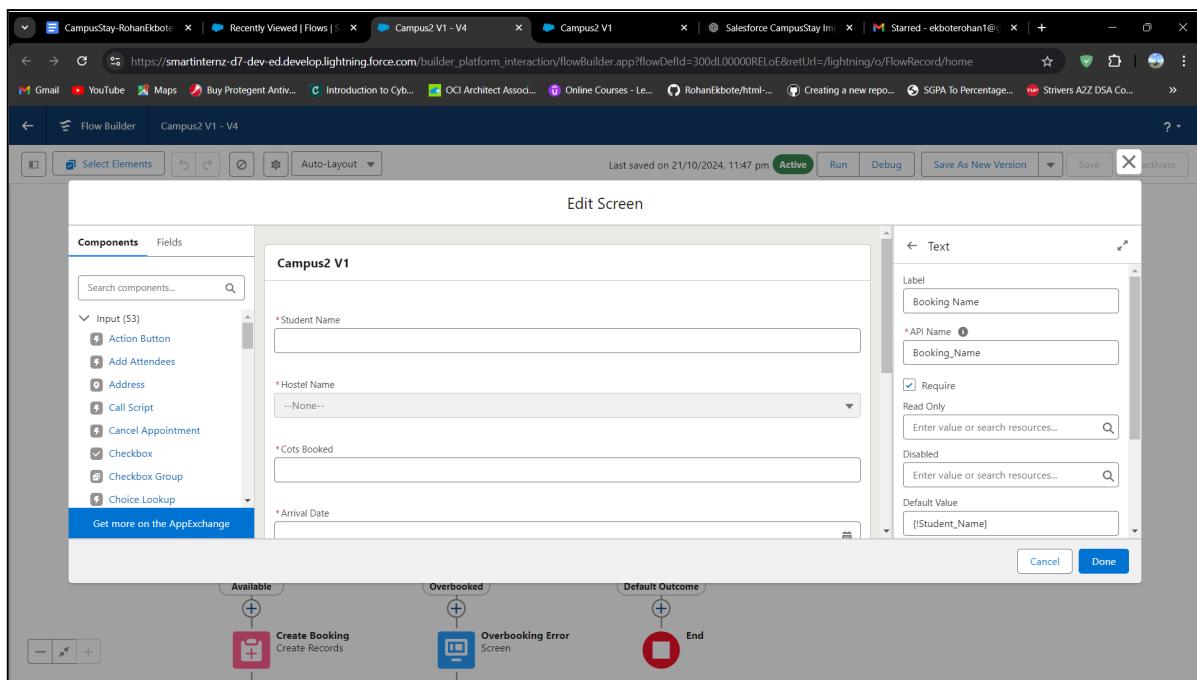
\* Departure Date  
26-Dec-2024

Room Preferences  
1st floor

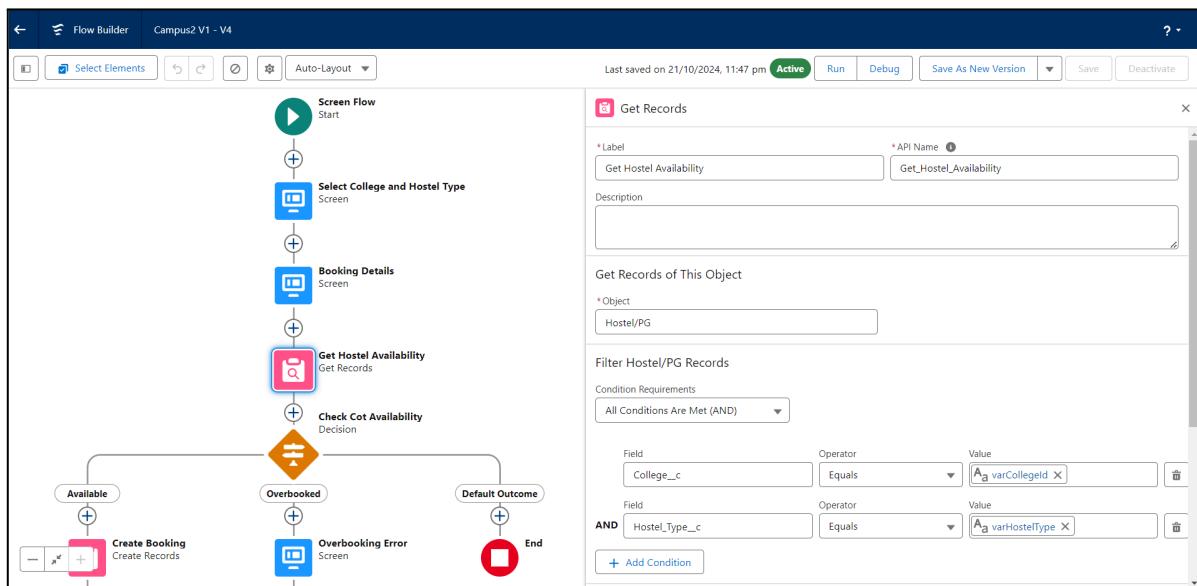
\* Booking Name  
Rohan Ekbote

\* Booking Date  
22-Oct-2024

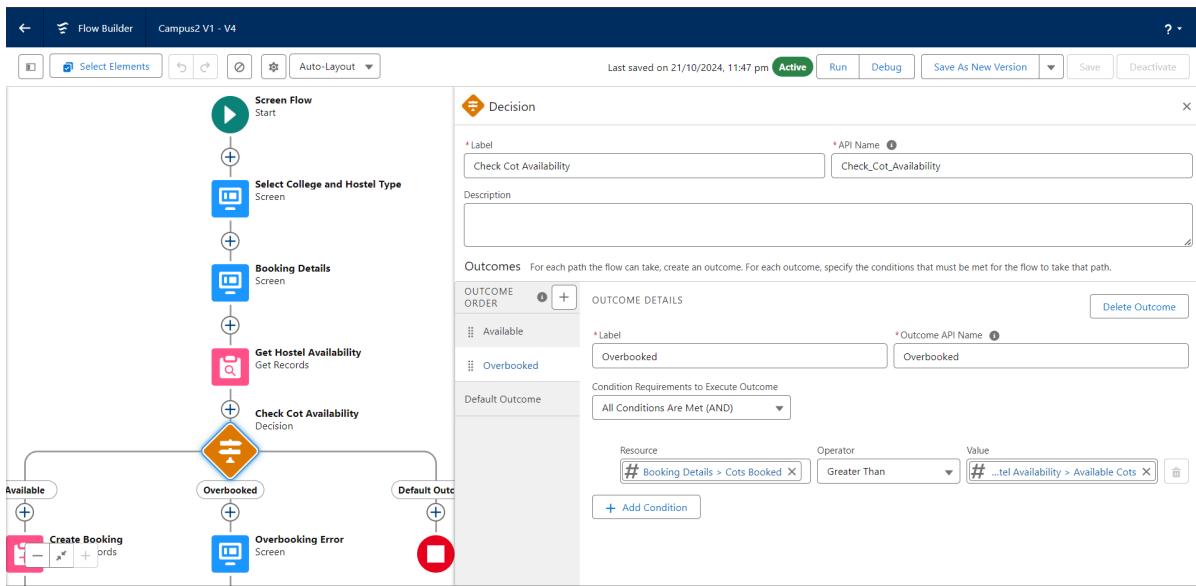
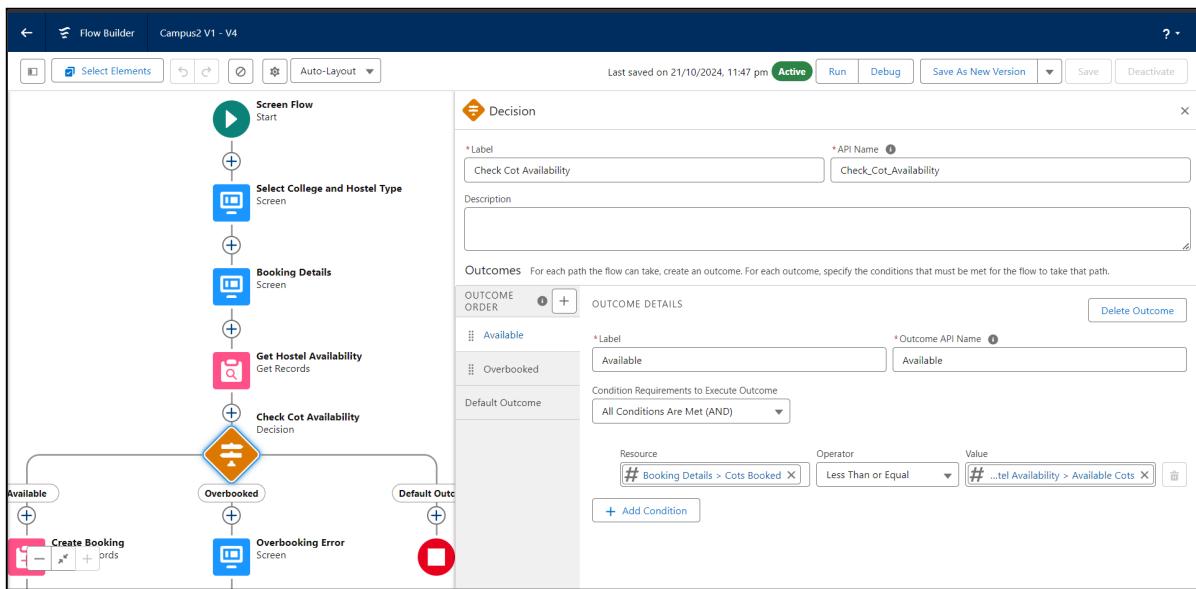
**Previous** **Next**



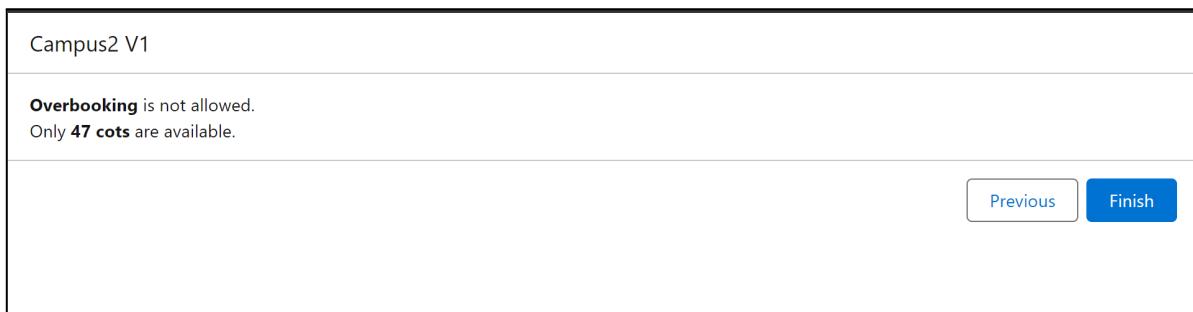
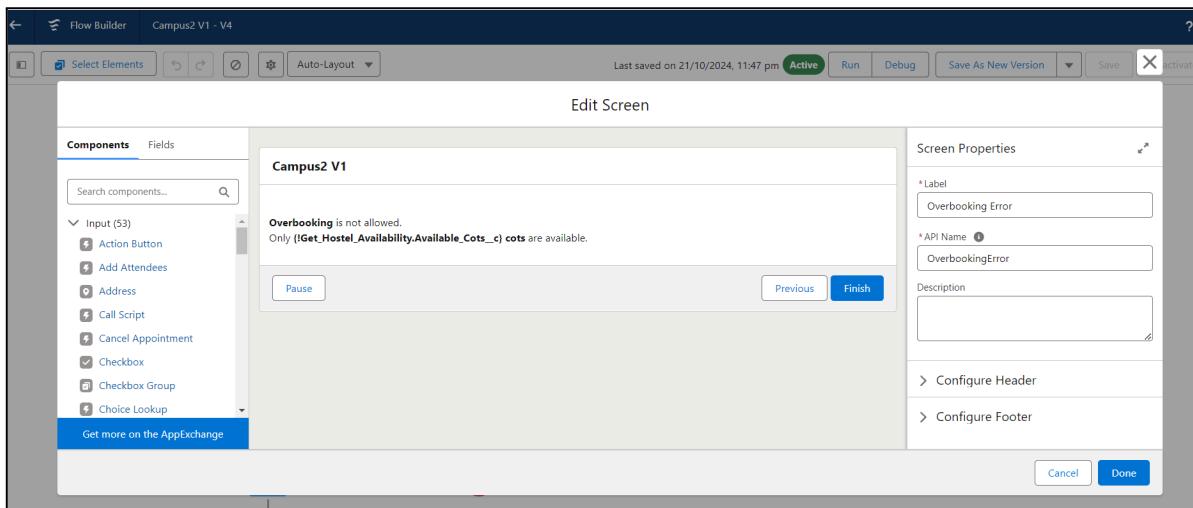
### 3. Get Hostel Availability



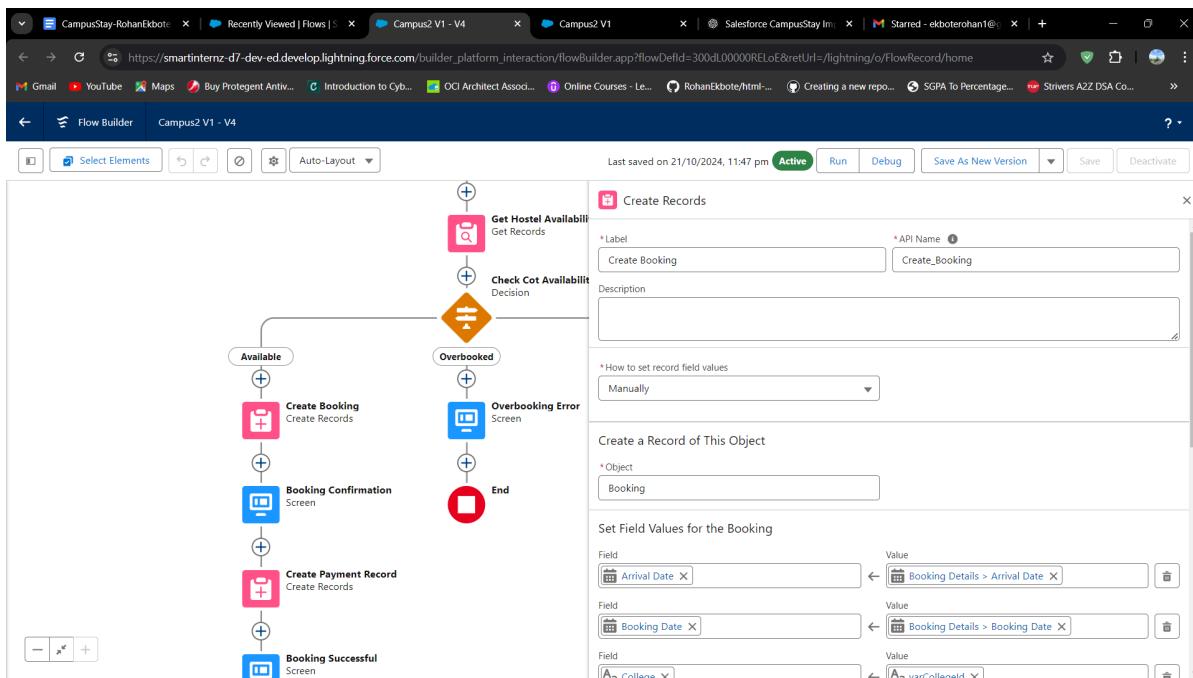
### 4. Check Cot Availability (Decision Element)



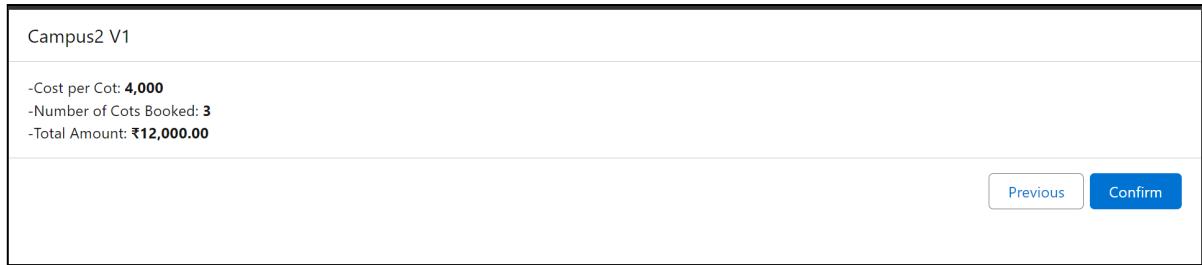
## 5. Overbooking Error



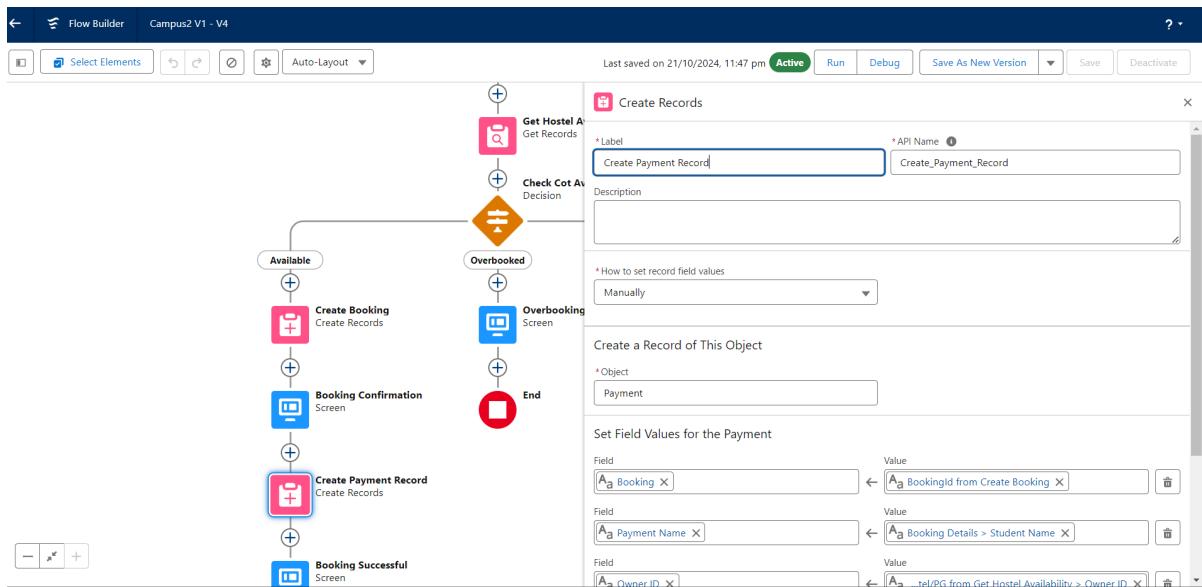
## 6. Create Booking



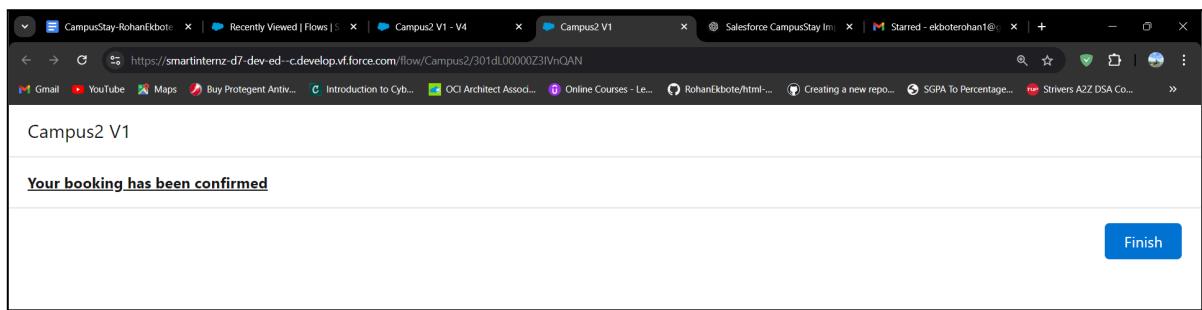
## 7. Booking Confirmation Screen

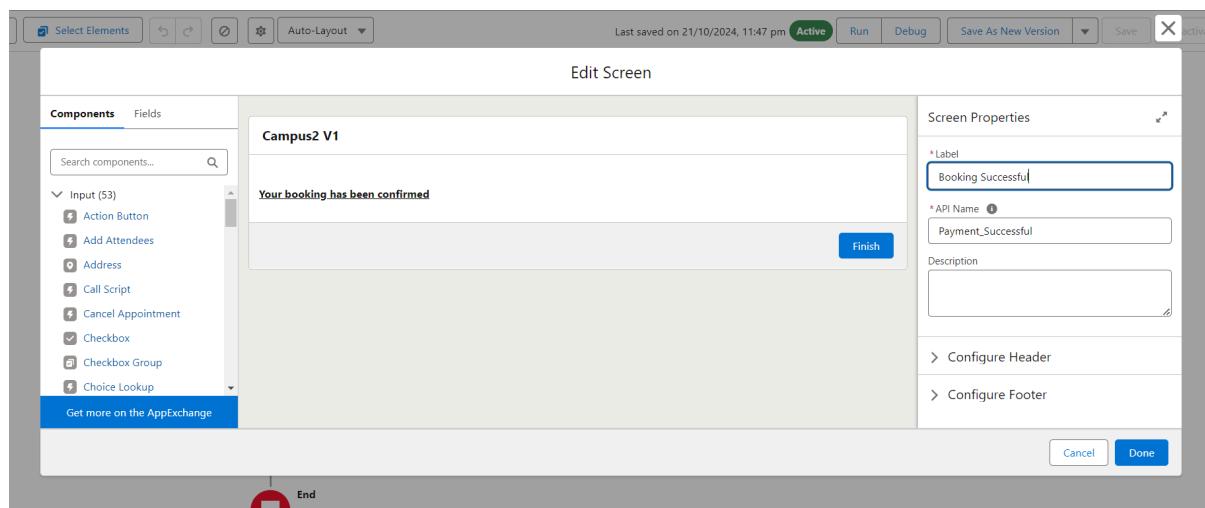


## 8. Payment Record



## 9. Booking Successful Screen





## 10. Booking Record Created (Rohan Ekbote)

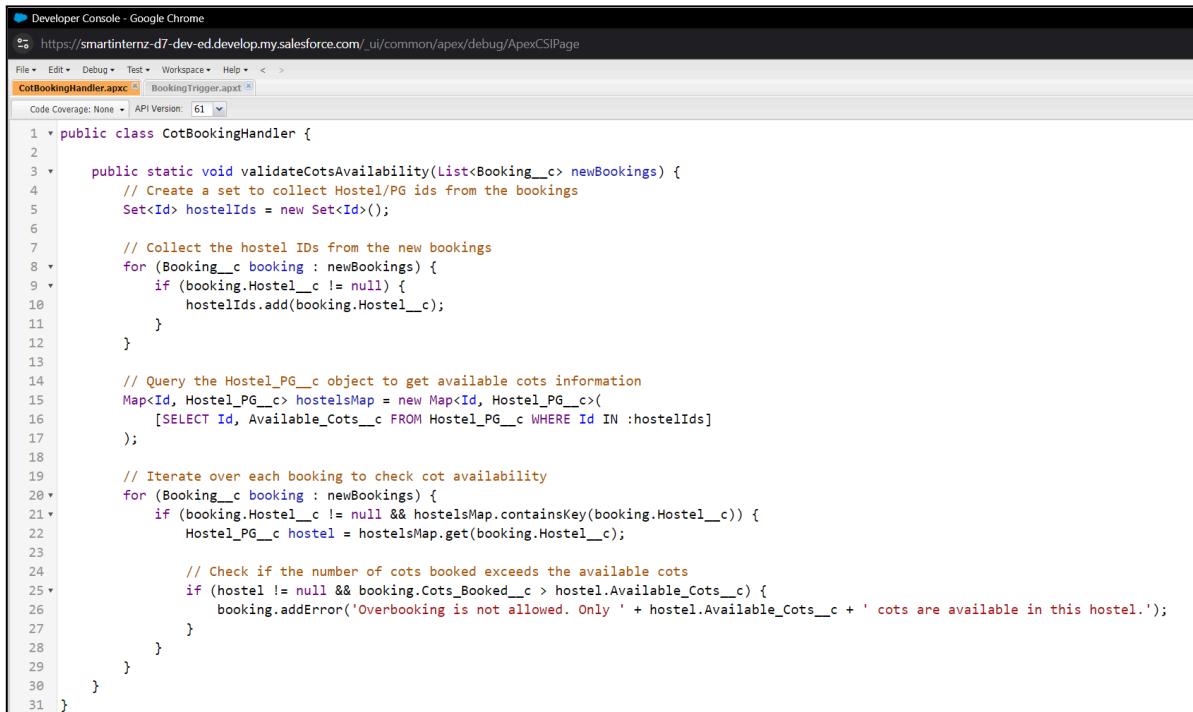
	Booking Name	Action
1	Anil	<input type="checkbox"/>
2	Cinna	<input type="checkbox"/>
3	Rohan	<input type="checkbox"/>
4	Rohan Ekbote	<input type="checkbox"/>
5	Shivam	<input type="checkbox"/>

## 11. Payment Record Created (Rohan Ekbote)

	Payment Name	Action
1	Cinna	<input type="checkbox"/>
2	Rohan	<input type="checkbox"/>
3	Rohan Ekbote	<input type="checkbox"/>
4	Shivam	<input type="checkbox"/>

## 6. Class Handler & Trigger Review

### 6.1 Class: CotBookingHandler



The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is https://smartinternz-d7-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSIPage. The tab is CotBookingHandler.apxc. The API Version is set to 61. The code is as follows:

```
1 public class CotBookingHandler {
2
3     public static void validateCotsAvailability(List<Booking__c> newBookings) {
4         // Create a set to collect Hostel/PG ids from the bookings
5         Set<Id> hostelIds = new Set<Id>();
6
7         // Collect the hostel IDs from the new bookings
8         for (Booking__c booking : newBookings) {
9             if (booking.Hostel__c != null) {
10                 hostelIds.add(booking.Hostel__c);
11             }
12         }
13
14         // Query the Hostel_PG__c object to get available cots information
15         Map<Id, Hostel_PG__c> hostelsMap = new Map<Id, Hostel_PG__c>(
16             [SELECT Id, Available_Cots__c FROM Hostel_PG__c WHERE Id IN :hostelIds]
17         );
18
19         // Iterate over each booking to check cot availability
20         for (Booking__c booking : newBookings) {
21             if (booking.Hostel__c != null && hostelsMap.containsKey(booking.Hostel__c)) {
22                 Hostel_PG__c hostel = hostelsMap.get(booking.Hostel__c);
23
24                 // Check if the number of cots booked exceeds the available cots
25                 if (hostel != null && booking.Cots_Booked__c > hostel.Available_Cots__c) {
26                     booking.addError('Overbooking is not allowed. Only ' + hostel.Available_Cots__c + ' cots are available in this hostel.');
27                 }
28             }
29         }
30     }
31 }
```

- **Purpose:** To check if there are enough available cots in the selected hostel before a booking is confirmed.
- **Key Points:**
  - Collects **Hostel IDs** from new bookings.
  - Queries the **Hostel/PG** object to retrieve the number of available cots for the selected hostels.
  - For each booking, check if the number of cots being booked exceeds the available cots.
  - If there is an overbooking attempt, it throws an error and prevents the booking.

## 6.2 Trigger: BookingTrigger

The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is https://smartinternz-d7-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSIPage. The tab title is CotBookingHandler.apxc [ BookingTrigger.apxt ]. The code editor displays the following Apex trigger:

```
1 trigger BookingTrigger on Booking__c (before insert, before update) {
2     // Call the handler to validate cot availability before insert or update
3     if (Trigger.isBefore && (Trigger.isInsert || Trigger.isUpdate)) {
4         CotBookingHandler.validateCotsAvailability(Trigger.new);
5     }
6 }
```

- **Purpose:** To trigger the cot availability validation before a booking is inserted or updated.
- **Key Points:**
  - Runs **before insert** and **before update** on the **Booking** object.
  - Calls the validateCotsAvailability method from the **CotBookingHandler** class.
  - Ensures no overbooking by validating cot availability before saving the record.

## Summary:

- This trigger-handler setup ensures that students cannot book more cots than are available in the selected hostel, effectively preventing overbooking errors. It runs every time a new booking is created or updated, validating the cot availability in real time.

---

## 7. Testing and Validation

### 7.1 Unit Testing:

- **Apex Classes and Triggers:** Tests ensure that triggers are correctly filling in the **Hostel Type** and that all logic around cot availability and payments works as expected.

New Booking

\* = Required Information

Information

* Booking Name Annapurna	* Hostel DYP Girls
* Student Name Annapurna	* Hostel Type Girls Hostel
* College DYP	* Cots Booked 250
	* Booking Date 21/10/2024
* Arrival Date 21/10/2024	
Departure Date 19/12/2024	

Room Preferences

Cancel Save & New Save

This screenshot shows a successful booking submission. All required fields are filled out correctly, and the form has been submitted successfully.

New Booking

\* = Required Information

Information

* Booking Name Annapurna	* Hostel DYP Girls
* Student Name Annapurna	* Hostel Type Girls Hostel
* College DYP	* Cots Booked 250
	* Booking Date 21/10/2024
* Arrival Date 21/10/2024	
Departure Date 19/12/2024	

We hit a snag.

Review the errors on this page.

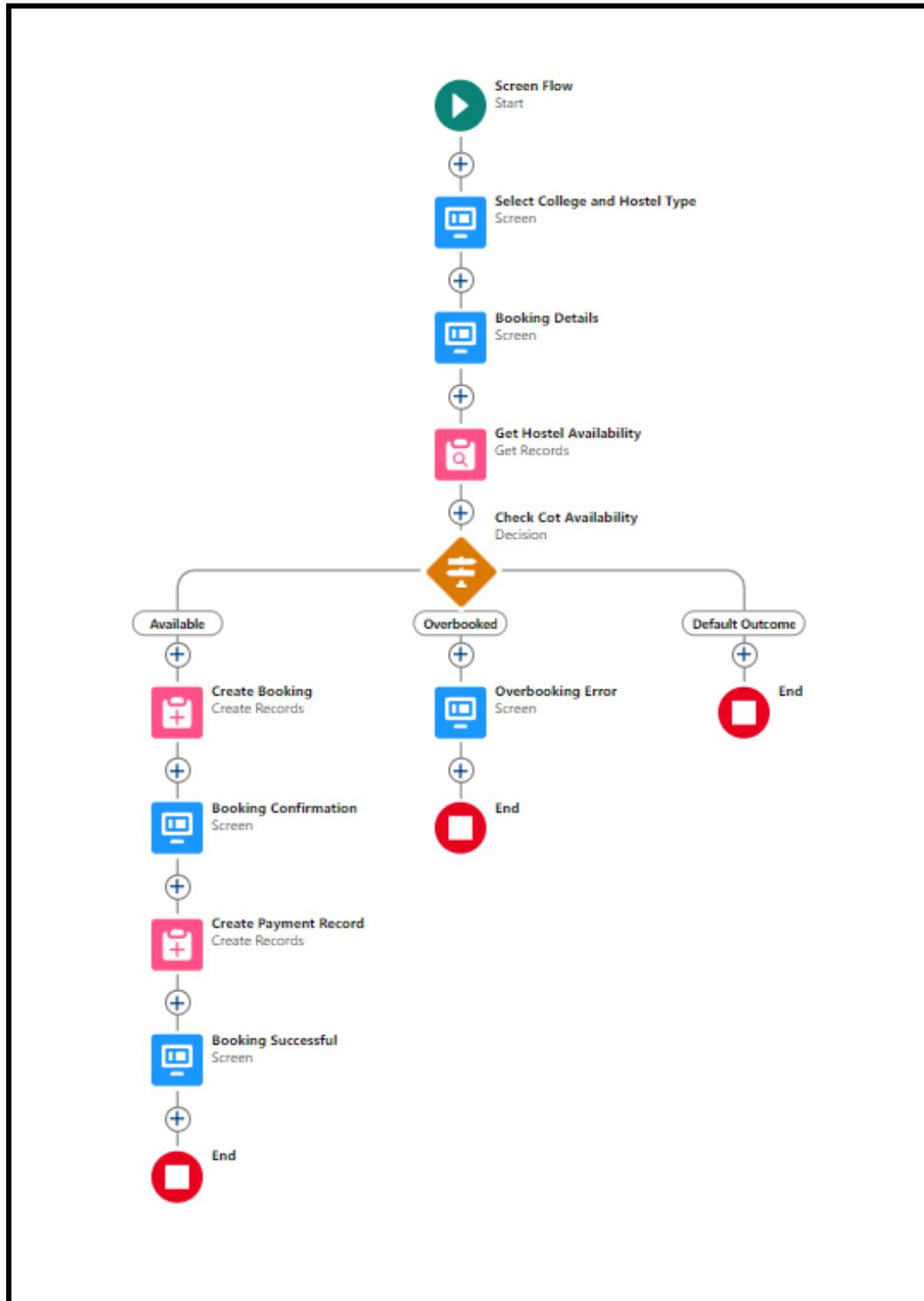
- Overbooking is not allowed. Only 44.00 cots are available in this hostel.

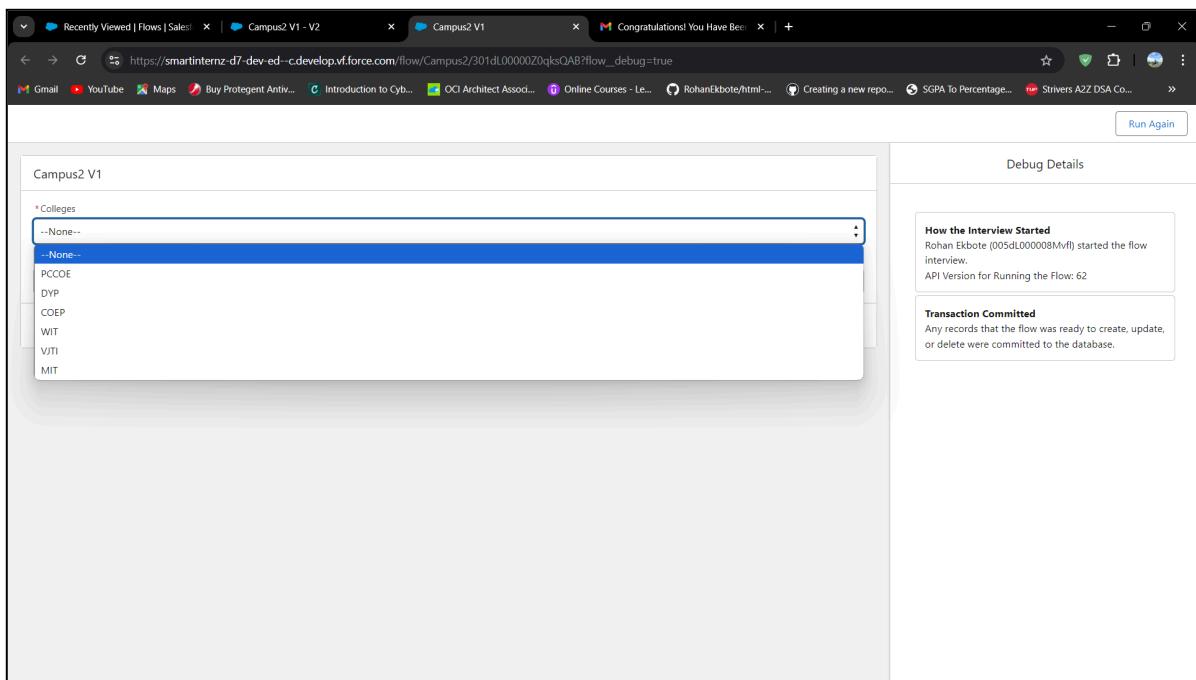
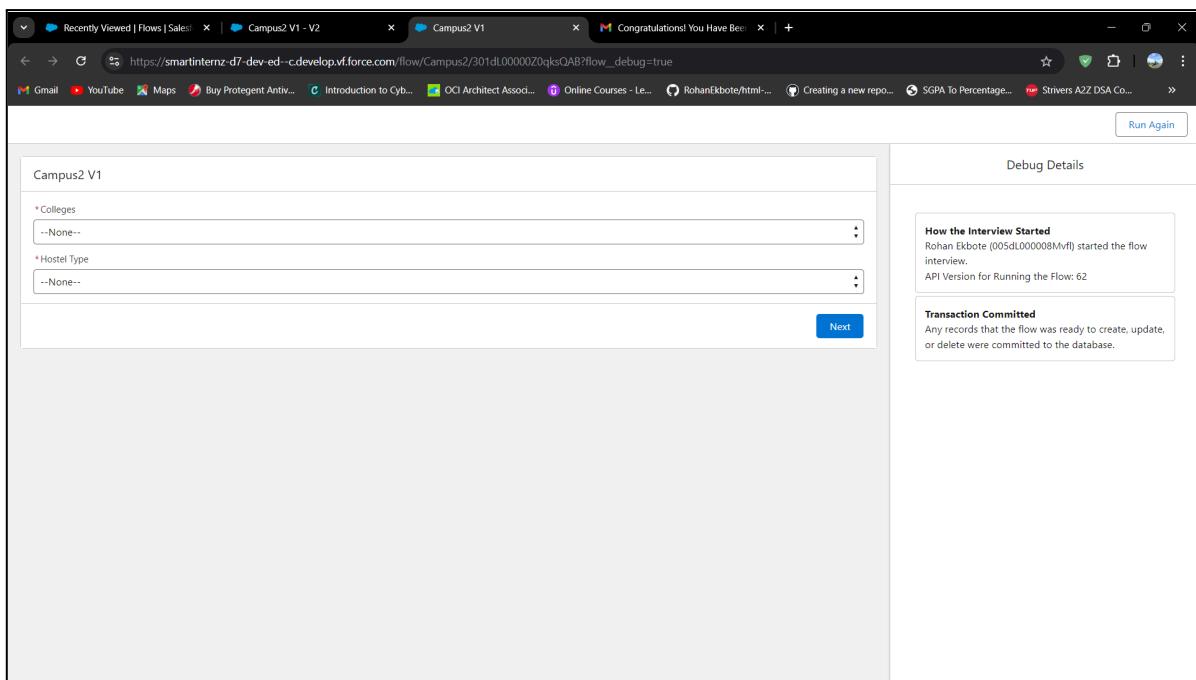
Cancel Save & New Save

This screenshot shows an error message indicating that overbooking is not allowed because there are only 44.00 cots available in the selected hostel, despite the user trying to book 250 cots. The arrival date is set to the current date, which is likely causing the overbooking issue.

## 7.2 User Interface Testing:

- **Flow Testing:** The flow was thoroughly tested to ensure proper cot availability checks, booking creation, and payment record auto-generation.
- Screens like the **Booking Confirmation Screen** and **Booking Successful Screen** were validated to ensure they display accurate information.





Recently Viewed | Flows | Sales| Campus2 V1 - V2 | Campus2 V1 | Congratulations! You Have Been...

https://smartinternz-d7-dev-ed-c.develop.vf.force.com/flow/Campus2/301dL00000Z0qksQAB?flow\_debug=true

Gmail YouTube Maps Buy Protectant Anti... Introduction to Cyb... OCI Architect Assoc... Online Courses - Le... RohanEkbote/html... Creating a new repo... SGPA To Percentage... Strivers A2Z DSA Co...

Run Again

Campus2 V1

\* Colleges  
VITI

+ Hostel Type  
--None--  
--None--  
Boys Hostel  
Girls Hostel

Debug Details

**How the Interview Started**  
Rohan Ekbote (005dL000008Mvfj) started the flow interview.  
API Version for Running the Flow: 62

**Transaction Committed**  
Any records that the flow was ready to create, update, or delete were committed to the database.

Recently Viewed | Flows | Sales| Campus2 V1 - V2 | Campus2 V1 | Congratulations! You Have Been...

https://smartinternz-d7-dev-ed-c.develop.vf.force.com/flow/Campus2/301dL00000Z0qksQAB?flow\_debug=true

Gmail YouTube Maps Buy Protectant Anti... Introduction to Cyb... OCI Architect Assoc... Online Courses - Le... RohanEkbote/html... Creating a new repo... SGPA To Percentage... Strivers A2Z DSA Co...

Run Again

Campus2 V1

\* Colleges  
VITI

+ Hostel Type  
Boys Hostel

Next

Debug Details

**How the Interview Started**  
Rohan Ekbote (005dL000008Mvfj) started the flow interview.  
API Version for Running the Flow: 62

**Transaction Committed**  
Any records that the flow was ready to create, update, or delete were committed to the database.

Campus2 V1

\* Student Name: Shivam

\* Hostel Name: **VITI Boys**

\* Booking Name: Shivam

\* Booking Date: 21-Oct-2024

**Debug Details**

Any records that the flow was ready to create, update, or delete were committed to the database.

**SCREEN: Select College and Hostel Type**  
**Dropdown List: Colleges**  
 Label: Colleges  
 Data Type: Text  
**Choices selected at runtime:**  
 Choice selected at runtime: VITI (CollegesSet)  
 Choice value: a00dL00000Q4GIZQAV  
 Stored field values:  
 {varCollegeId} = Id (a00dL00000Q4GIZQAV)

**Dropdown List: Hostel\_Type**  
 Label: Hostel\_Type  
 Data Type: Text  
**Choices selected at runtime:**  
 Choice selected at runtime: Boys Hostel (HostelTypeee)  
 Choice value: Boys Hostel  
 Stored field values:  
 {varHostelType} = Hostel\_Type\_\_c (Boys Hostel)

**Selected Navigation Button: NEXT**

**Transaction Committed**

Any records that the flow was ready to create, update, or delete were committed to the database.

Campus2 V1

\* Student Name: Shivam

\* Hostel Name: **VITI Boys**

\* Cots Booked: 10

\* Arrival Date: 22-Oct-2024

\* Departure Date: 31-Jan-2025

Format: 31-Dec-2024

Room Preferences

\* Booking Name: Shivam

\* Booking Date: 21-Oct-2024

**Debug Details**

Any records that the flow was ready to create, update, or delete were committed to the database.

**SCREEN: Select College and Hostel Type**  
**Dropdown List: Colleges**  
 Label: Colleges  
 Data Type: Text  
**Choices selected at runtime:**  
 Choice selected at runtime: VITI (CollegesSet)  
 Choice value: a00dL00000Q4GIZQAV  
 Stored field values:  
 {varCollegeId} = Id (a00dL00000Q4GIZQAV)

**Dropdown List: Hostel\_Type**  
 Label: Hostel\_Type  
 Data Type: Text  
**Choices selected at runtime:**  
 Choice selected at runtime: Boys Hostel (HostelTypeee)  
 Choice value: Boys Hostel  
 Stored field values:  
 {varHostelType} = Hostel\_Type\_\_c (Boys Hostel)

**Selected Navigation Button: NEXT**

**Transaction Committed**

Any records that the flow was ready to create, update, or delete were committed to the database.

Campus2 V1

\* Student Name: Shivam

+ Hostel Name: VJTI Boys

\* Cots Booked: 10

\* Arrival Date: 22-Oct-2024

\* Departure Date: 31-Jan-2025

Room Preferences: 1st Floor and Near Window

\* Booking Name: Shivam

\* Booking Date: 21-Oct-2024

[Previous](#) [Next](#)

[Run Again](#)

**Debug Details**

Any records that the flow was ready to create, update, or delete were committed to the database.

**SCREEN: Select College and Hostel Type**

**Dropdown List: Colleges**  
Label: Colleges  
Data Type: Text  
**Choices selected at runtime:**  
Choice selected at runtime: VITI (CollegesSet)  
Choice value: a00dL00000Q4GIZQAV  
Stored field values:  
{varCollegeId} = Id (a00dL00000Q4GIZQAV)

**Dropdown List: Hostel\_Type**  
Label: Hostel Type  
Data Type: Text  
**Choices selected at runtime:**  
Choice selected at runtime: Boys Hostel (HostelTypeee)  
Choice value: Boys Hostel  
Stored field values:  
{varHostelType} = Hostel\_Type\_\_c (Boys Hostel)

**Selected Navigation Button: NEXT**

**Transaction Committed**

Any records that the flow was ready to create, update, or delete were committed to the database.

Campus2 V1

-Cost per Cot: 10,000  
-Number of Cots Booked: 10  
-Total Amount: ₹1,00,000.00

[Previous](#) [Confirm](#)

[Run Again](#)

**Debug Details**

Outcome conditions:  
(!Cots\_Booked) (10) Less than or equal  
(!Get\_Hostel\_Availability.Available\_Cots\_\_c) (100)  
All conditions must be true (AND)

**CREATE RECORDS: Create Booking**

Create one Booking\_\_c record where:  
Arrival\_Date\_\_c = {Arrival\_Date} (22 October 2024)  
Booking\_Date\_\_c = {Booking\_Date} (21 October 2024)  
College\_\_c = {varCollegeId} (a00dL00000Q4GIZQAV)  
Cots\_Booked\_\_c = {Cots\_Booked} (10)  
Departure\_Date\_\_c = {Departure\_Date} (31 January 2025)  
Hostel\_Type\_\_c = {varHostelType} (Boys Hostel)  
Hostel\_\_c = {HostName} (a01dL00000Z0D1GA)  
Name = {Booking\_Name} (Shivam)  
Room\_Preferences\_\_c = {Room\_Preferences} (1st Floor and Near Window)  
Student\_Name\_\_c = {Student\_Name} (Shivam)

**Result**

A record is ready to be created when the next screen, pause, or local action is executed or when the interview finishes.  
a02dL000006LkR0QAK

**Transaction Committed**

Any records that the flow was ready to create, update, or delete were committed to the database.

The screenshot shows a Salesforce Flow debug session. The main window displays a screen titled "Campus2 V1" with the message "Your booking has been confirmed" and a "Finish" button. To the right is a "Debug Details" panel.

**SCREEN:** Booking Confirmation  
**Display Text:** BookingSummary  
Value at run time:  
-Cost per Cot: 10,000  
-Number of Cots Booked: 10  
-Total Amount: ₹1,00,000.00

**Selected Navigation Button:** NEXT

**CREATE RECORDS:** Create Payment Record  
Create one Payment\_\_c record where:  
Booking\_\_c = {!Create\_Booking}  
(a02dL000006LkR0QAK)  
Name = {Student\_\_Name} (Shivam)  
OwnerId = {Get\_Hostel\_Availability.OwnerId}  
(005dL000008MvfIQC)  
Payment\_Date\_\_c = {!Booking\_Date} (21 October 2024)  
**Result**  
A record is ready to be created when the next screen, pause, or local action is executed or when the interview finishes.  
a03dL00000A0YlkQAN

**Transaction Committed**  
Any records that the flow was ready to create, update, or delete were committed to the database.

The screenshot shows a Salesforce Flow debug session. The main window displays a screen titled "Campus2 V1" with the message "All done" and the note "The flow interview finished at 21-Oct-2024, 4:09:59 pm". It includes "Change Inputs" and "Run Again" buttons. To the right is a "Debug Details" panel.

Create one Payment\_\_c record where:  
Booking\_\_c = {!Create\_Booking}  
(a02dL000006LkR0QAK)  
Name = {Student\_\_Name} (Shivam)  
OwnerId = {Get\_Hostel\_Availability.OwnerId}  
(005dL000008MvfIQC)  
Payment\_Date\_\_c = {!Booking\_Date} (21 October 2024)  
**Result**  
A record is ready to be created when the next screen, pause, or local action is executed or when the interview finishes.  
a03dL00000A0YlkQAN

**Transaction Committed**  
Any records that the flow was ready to create, update, or delete were committed to the database.

**SCREEN:** Payment Successful  
**Display Text:** Payment  
Value at run time:  
Your booking has been confirmed

**Selected Navigation Button:** NEXT

**Transaction Committed**  
Any records that the flow was ready to create, update, or delete were committed to the database.

The screenshot shows the CampusStay application interface. The top navigation bar includes links for Home, Dashboards, Reports, Colleges, Bookings, Hostel/PGs, Payments, and Flows. The Bookings section is active, showing a list of bookings. The search bar at the top right contains the placeholder "Search this list...".

3 items • Sorted by Booking Name • Filtered by All bookings • Updated a few seconds ago

	Booking Name
1	Cinna
2	Rohan
3	Shivam

The screenshots illustrate the CampusStay software's user interface for managing payments. The top screenshot shows a list of payments, with one entry for 'Shivam'. The bottom screenshot provides a detailed view of this specific payment record, showing booking information, payment date, total amount, owner, and activity history.

## 8. Key Scenarios Addressed by Salesforce

- Booking a Hostel Room:** A student selects a college, chooses between Boys' and Girls' Hostels, and books a cot. The system automatically checks cot availability, ensures no overbooking occurs, and creates both the booking and payment records.
- Handling Overbooking:** The system prevents overbooking by checking cot availability in real-time. If no cots are available, the user receives an error message and cannot proceed.
- Payment Automation:** Once a booking is made, a payment record is automatically generated with all relevant details, including the total amount and payment status.

## 9. Roles and Profiles with Hierarchy

- **Student Profile:**

- Can view available hostels, make bookings, and view their payment history.
- **Hostel Manager Profile:**
  - Can manage bookings, view cot availability, and update hostel details (e.g., total cots and cot rates).
- **System Administrator:**
  - Has full access to all records, objects, and settings. Manages the flow logic, booking records, and triggers.

### Hierarchy:

1. **System Administrator:** Highest authority with full access.
2. **Hostel Manager:** Limited to managing hostels and bookings.
3. **Student:** Can only view and book cots, with access limited to their own records.

**SETUP**

**Roles**

## Creating the Role Hierarchy

You can build on the existing role hierarchy shown on this page. To insert a new role, click **Add Role**.

### Your Organization's Role Hierarchy

[Collapse All](#) [Expand All](#)

- **SmartInternz**
  - └ [Add Role](#)
  - **CEO** [Edit](#) | [Del](#) | [Assign](#)
    - └ [Add Role](#)
  - **Admin** [Edit](#) | [Del](#) | [Assign](#)
    - └ [Add Role](#)
  - **Hostel Manager** [Edit](#) | [Del](#) | [Assign](#)
    - └ [Add Role](#)
  - └ **Student** [Edit](#) | [Del](#) | [Assign](#)
    - └ [Add Role](#)
- **CFO** [Edit](#) | [Del](#) | [Assign](#)

## **10. Conclusion**

In the **CampusStay** project, we successfully developed a Salesforce-based hostel booking system that automates the process of room allocation and payment. By leveraging Salesforce's powerful Flow Builder, custom objects, triggers, and automation features, we've created a seamless experience for students and hostel managers. The application ensures no overbooking occurs and provides an easy way to manage payments.

### **Achievements:**

- Automated hostel booking and payment generation.
  - Prevented overbooking through real-time availability checks.
  - Simplified management for hostel managers through automated processes.
-