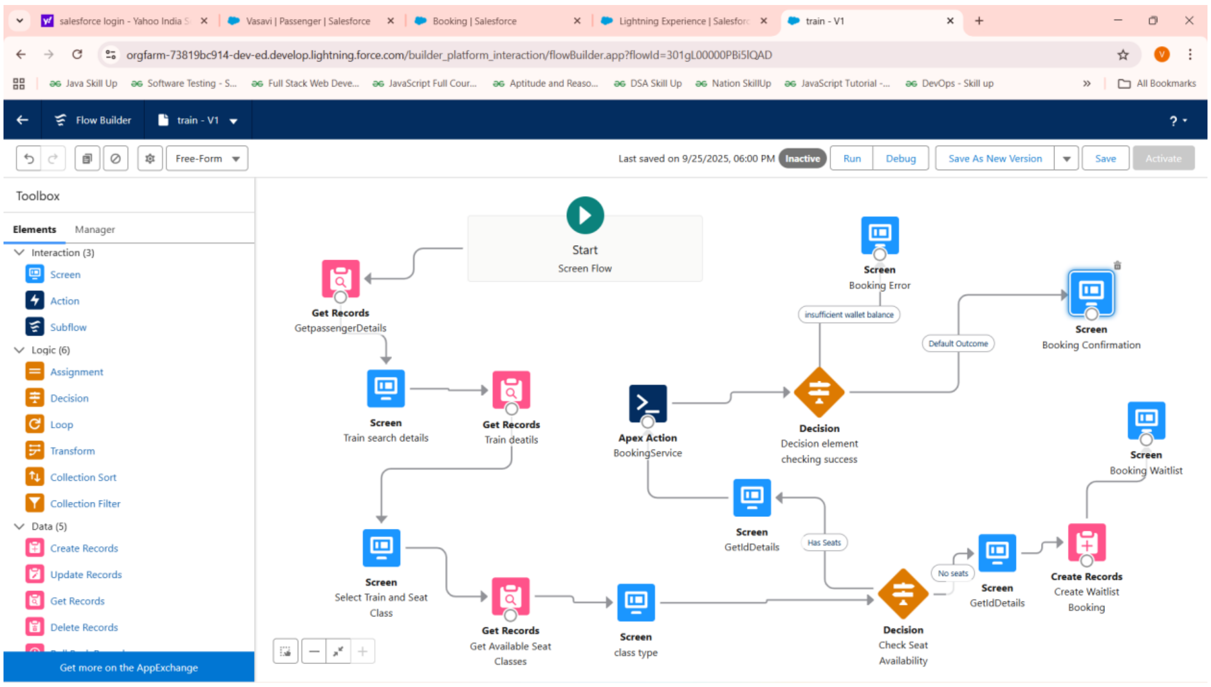
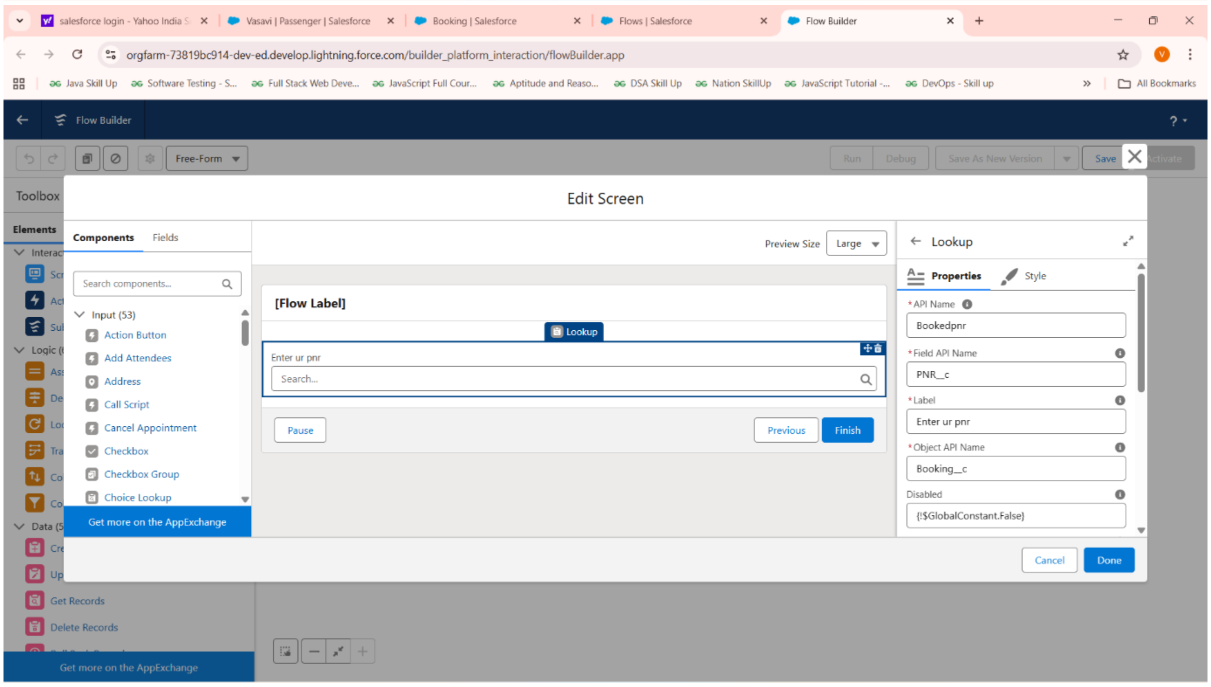
Train Ticket Booking System

Phase 4: Process Automation (Admin)

- Validation Rules
- Workflow Rules
- Process Builder
- Approval Process
- Flow Builder (Screen, Record-Triggered, Scheduled, Auto-launched)
- Email Alerts
- Field Updates
- Tasks
- Custom Notifications





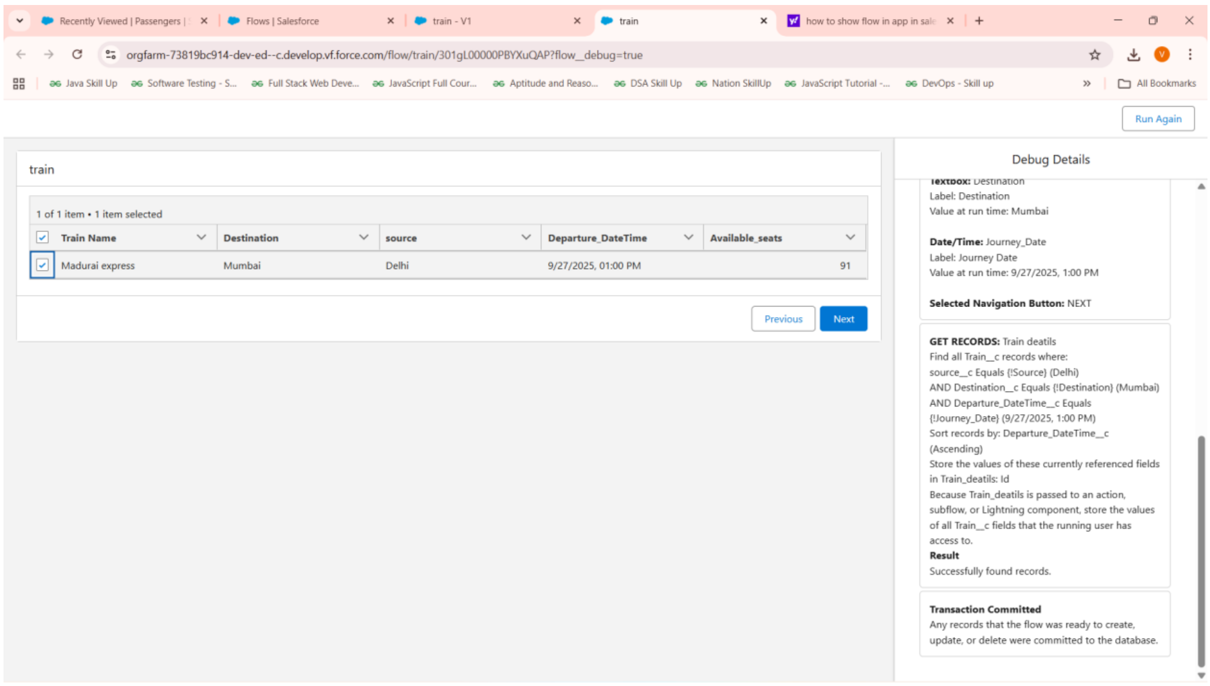
Edit Screen

[Flow Label]

Enter ur pnr
Search...

Lookup

* API Name: Bookedpnr
* Field API Name: PNR__c
* Label: Enter ur pnr
* Object API Name: Booking__c
Disabled: {!$GlobalConstant.False}



orgfarm-73819bc914-dev-ed--c.develop.vf.force.com/flow/train/301gL00000PBYXuQAP?flow__debug=true

train

1 of 1 item • 1 item selected

✓	Train Name	Destination	source	Departure_DateTime	Available_seats
✓	Madurai express	Mumbai	Delhi	9/27/2025, 01:00 PM	91

Debug Details

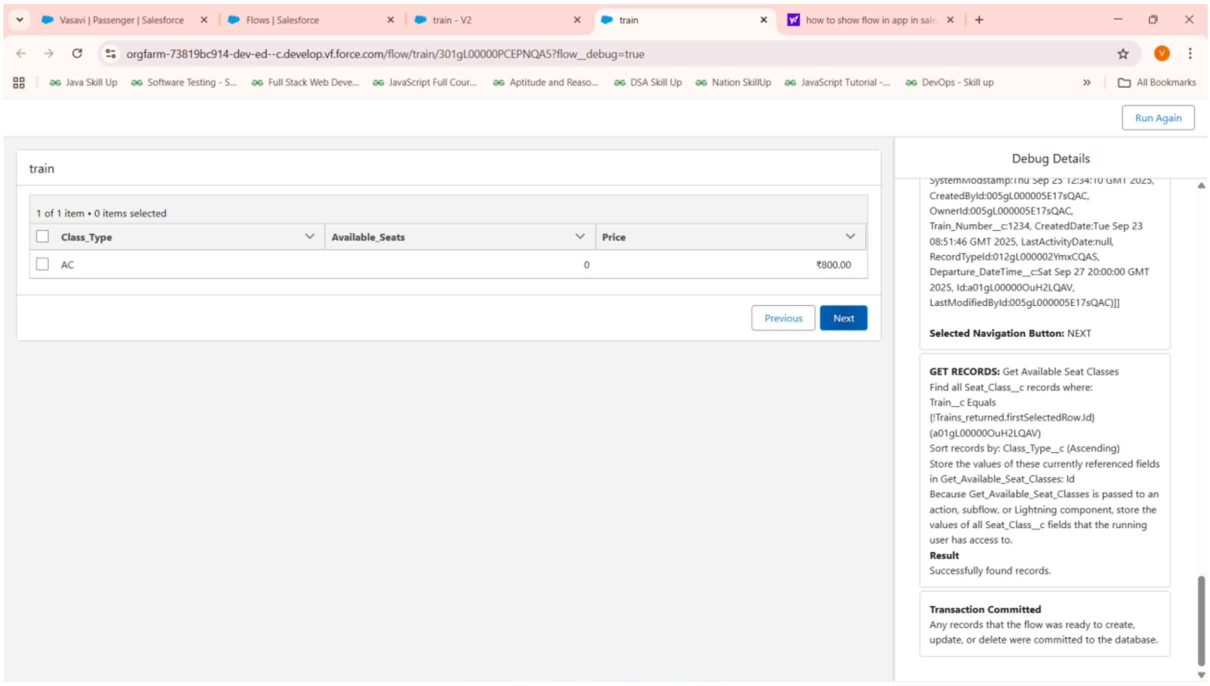
Textbox: Destination
Label: Destination
Value at run time: Mumbai

Date/Time: Journey_Date
Label: Journey Date
Value at run time: 9/27/2025, 1:00 PM

Selected Navigation Button: NEXT

GET RECORDS: Train deatils
Find all Train__c records where:
source__c Equals {!Source} (Delhi)
AND Destination__c Equals {!Destination} (Mumbai)
AND Departure_DateTime__c Equals
{!Journey_Date} (9/27/2025, 1:00 PM)
Sort records by: Departure_DateTime__c
(Ascending)
Store the values of these currently referenced fields in Train_deatils: Id
Because Train_deatils is passed to an action, subflow, or Lightning component, store the values of all Train__c fields that the running user has access to.
Result
Successfully found records.

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



orgfarm-73819bc914-dev-ed--c.develop.vf.force.com/flow/train/301gL00000PCEPNQA5?flow__debug=true

train

1 of 1 item • 0 items selected

	Class_Type	Available_Seats	Price
	AC	0	₹800.00

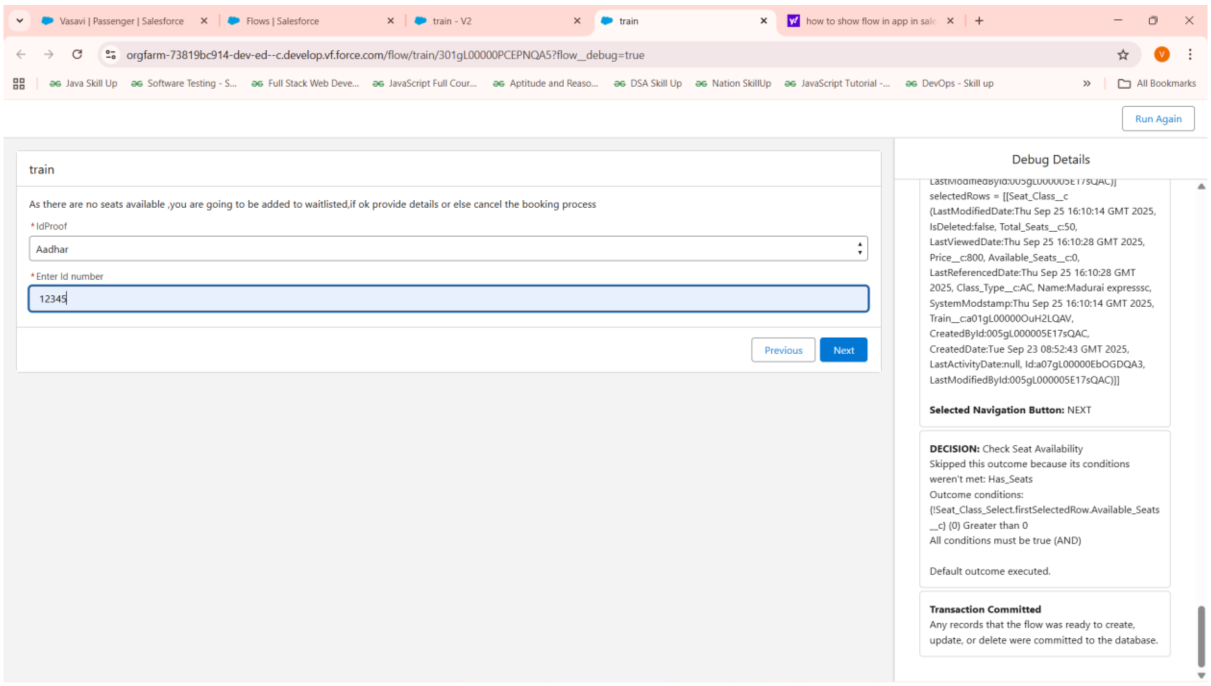
Debug Details

SystemModstamp:Thu Sep 25 12:34:10 GMT 2025,
CreatedById:005gL000005E17sQAC,
OwnerId:005gL000005E17sQAC,
Train_Number__c:1234, CreatedDate:Tue Sep 23 08:51:46 GMT 2025, LastActivityDate:null,
RecordTypeId:012gL000002YmxCQAS,
Departure_DateTime__c:Sat Sep 27 20:00:00 GMT 2025, Id:a01gL00000OuH2LQAV,
LastModifiedById:005gL000005E17sQAC)]]

Selected Navigation Button: NEXT

GET RECORDS: Get Available Seat Classes
Find all Seat_Class__c records where:
Train__c Equals
{!Trains_returned.firstSelectedRow.Id}
(a01gL00000OuH2LQAV)
Sort records by: Class_Type__c (Ascending)
Store the values of these currently referenced fields in Get_Available_Seat_Classes: Id
Because Get_Available_Seat_Classes is passed to an action, subflow, or Lightning component, store the values of all Seat_Class__c fields that the running user has access to.
Result
Successfully found records.

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



train

As there are no seats available ,you are going to be added to waitlisted,if ok provide details or else cancel the booking process

* IdProof

Aadhar

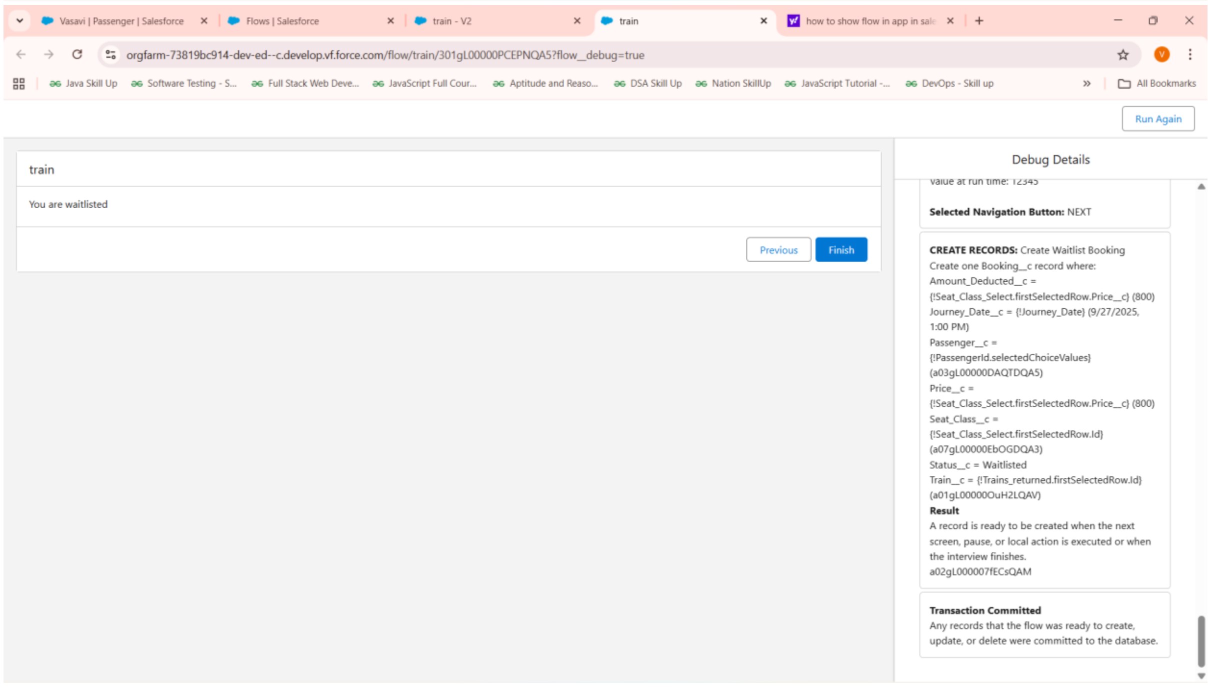
* Enter Id number

12345

Previous Next

Debug Details

```
LastModifiedById:005gL000005E17sQAC)]
selectedRows = [[Seat_Class__c
(LastModifiedDate:Thu Sep 25 16:10:14 GMT 2025,
IsDeleted:false, Total_Seats__c:50,
LastViewedDate:Thu Sep 25 16:10:28 GMT 2025,
Price__c:800, Available_Seats__c:0,
LastReferencedDate:Thu Sep 25 16:10:28 GMT
2025, Class_Type__c:AC, Name:Madurai expresssc,
SystemModstamp:Thu Sep 25 16:10:14 GMT 2025,
Train__c:a01gL00000OuH2LQAV,
CreatedById:005gL000005E17sQAC,
CreatedDate:Tue Sep 23 08:52:43 GMT 2025,
LastActivityDate:null, Id:a07gL00000EbOGDQA3,
LastModifiedById:005gL000005E17sQAC)]]

Selected Navigation Button: NEXT

DECISION: Check Seat Availability
Skipped this outcome because its conditions
weren't met: Has_Seats
Outcome conditions:
{!Seat_Class_Select.firstSelectedRow.Available_Seats
__c} (0) Greater than 0
All conditions must be true (AND)

Default outcome executed.

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



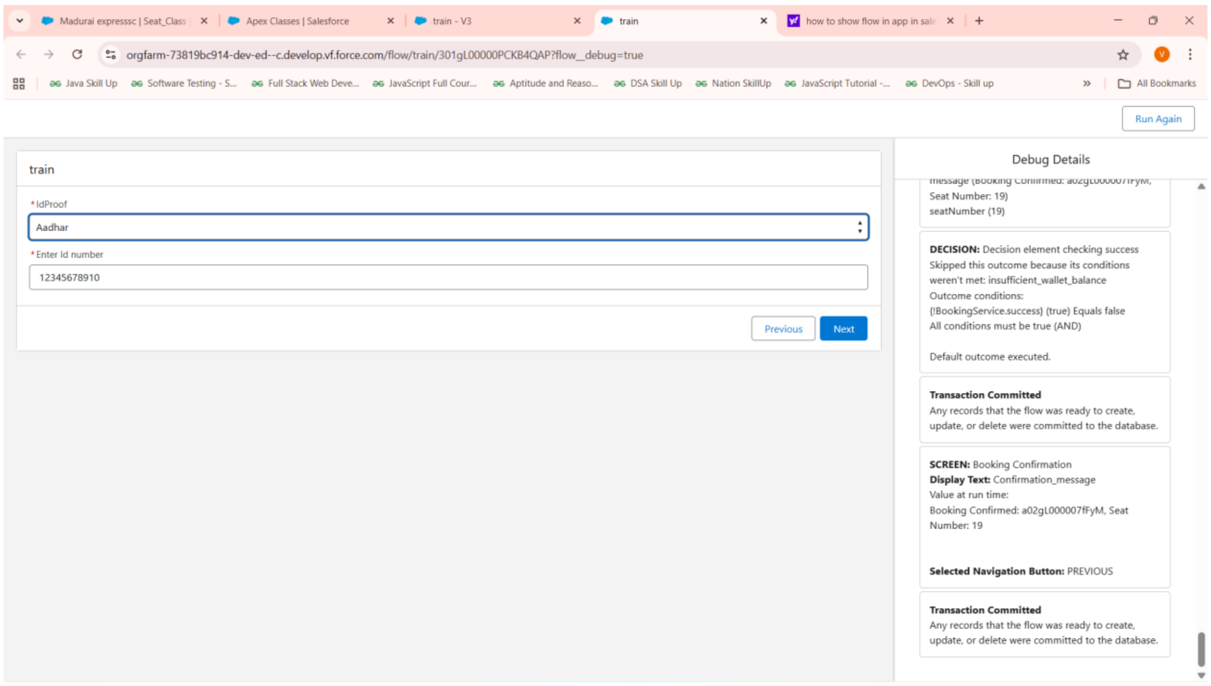
train

You are waitlisted

Previous Finish

Debug Details

```
Value at run time: 12345

Selected Navigation Button: NEXT

CREATE RECORDS: Create Waitlist Booking
Create one Booking__c record where:
Amount_Deducted__c =
{!Seat_Class_Select.firstSelectedRow.Price__c} (800)
Journey_Date__c = {!Journey_Date} (9/27/2025,
1:00 PM)
Passenger__c =
{!PassengerId.selectedChoiceValues}
(a03gL00000DAQTDQA5)
Price__c =
{!Seat_Class_Select.firstSelectedRow.Price__c} (800)
Seat_Class__c =
{!Seat_Class_Select.firstSelectedRow.Id}
(a07gL00000EbOGDQA3)
Status__c = Waitlisted
Train__c = {!Trains_returned.firstSelectedRow.Id}
(a01gL00000OuH2LQAV)
Result
A record is ready to be created when the next
screen, pause, or local action is executed or when
the interview finishes.
a02gL000007fECsQAM

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



train

*IdProof

Aadhar

*Enter Id number

12345678910

Previous Next

Debug Details

message (Booking Confirmed: a02gL000007fFyM,
Seat Number: 19)
seatNumber (19)

DECISION: Decision element checking success
Skipped this outcome because its conditions weren't met: insufficient_wallet_balance
Outcome conditions:
{!BookingService.success} (true) Equals false
All conditions must be true (AND)

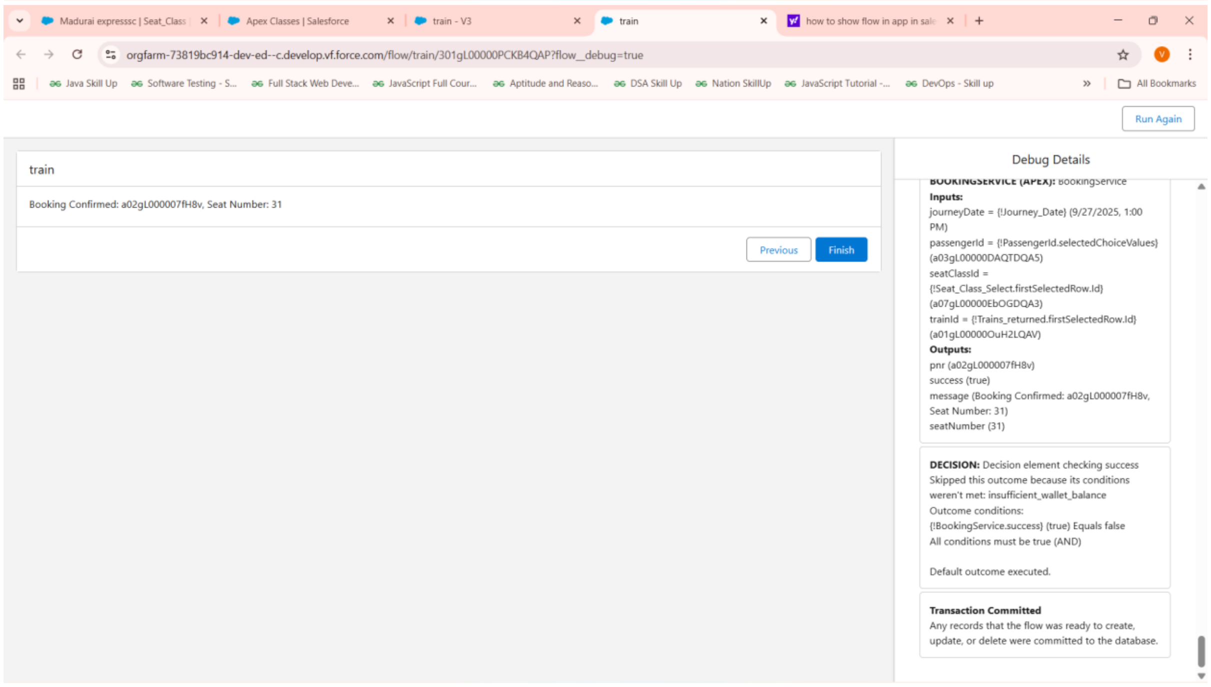
Default outcome executed.

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

SCREEN: Booking Confirmation
Display Text: Confirmation_message
Value at run time:
Booking Confirmed: a02gL000007fFyM, Seat Number: 19

Selected Navigation Button: PREVIOUS

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


Madurai expresssc | Seat_Class
Apex Classes | Salesforce
train - V3
train
how to show flow in app in sale
orgfarm-73819bc914-dev-ed--c.develop.vf.force.com/flow/train/301gL00000PCKB4QAP?flow__debug=true
Run Again

train

Booking Confirmed: a02gL000007fH8v, Seat Number: 31

Previous Finish

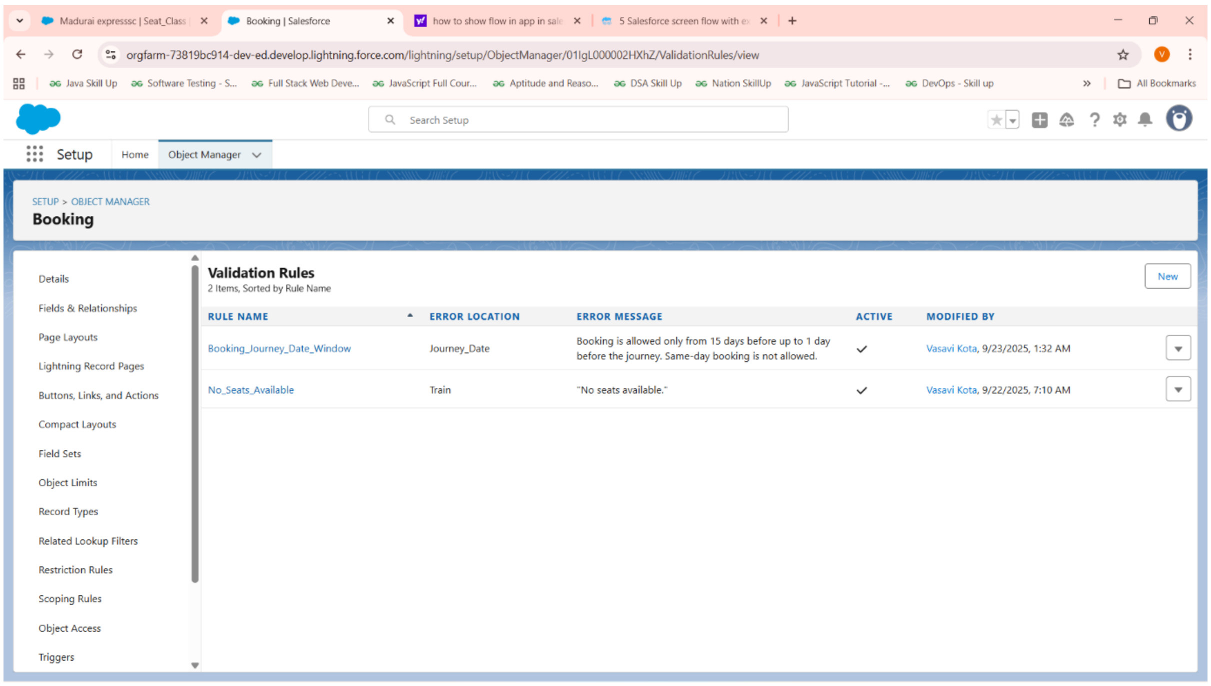
Debug Details

BOOKINGSERVICE (APEX): BookingService
Inputs:
journeyDate = {!Journey_Date} (9/27/2025, 1:00 PM)
passengerId = {!PassengerId.selectedChoiceValues} (a03gL00000DAQTDQA5)
seatClassId = {!Seat_Class_Select.firstSelectedRow.Id} (a07gL00000EbOGDQA3)
trainId = {!Trains_returned.firstSelectedRow.Id} (a01gL00000OuH2LQAV)
Outputs:
pnr (a02gL000007fH8v)
success (true)
message (Booking Confirmed: a02gL000007fH8v, Seat Number: 31)
seatNumber (31)

DECISION: Decision element checking success
Skipped this outcome because its conditions weren't met: insufficient_wallet_balance
Outcome conditions:
{!BookingService.success} (true) Equals false
All conditions must be true (AND)

Default outcome executed.

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

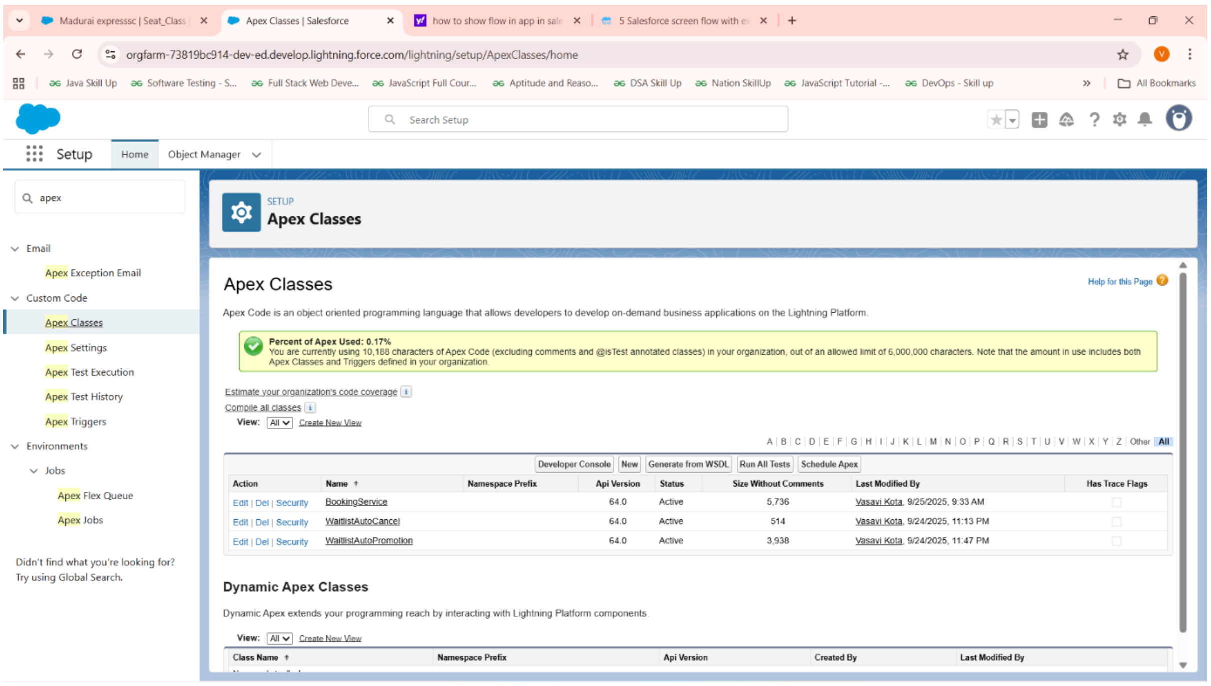

SETUP > OBJECT MANAGER
Booking

Validation Rules
2 Items, Sorted by Rule Name

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Booking_Journey_Date_Window	Journey_Date	Booking is allowed only from 15 days before up to 1 day before the journey. Same-day booking is not allowed.	✓	Vasavi Kota, 9/23/2025, 1:32 AM
No_Seats_Available	Train	"No seats available."	✓	Vasavi Kota, 9/22/2025, 7:10 AM

Phase 5: Apex Programming (Developer)

- Classes & Objects
- Apex Triggers (before/after insert/update/delete)
- Trigger Design Pattern
- SOQL & SOSL
- Collections: List, Set, Map
- Control Statements
- Batch Apex
- Queueable Apex
- Scheduled Apex
- Future Methods
- Exception Handling
- Test Classes
- Asynchronous Processing

- 

SETUP
Apex Classes

Apex Classes

Apex Code is an object oriented programming language that allows developers to develop on-demand business applications on the Lightning Platform.

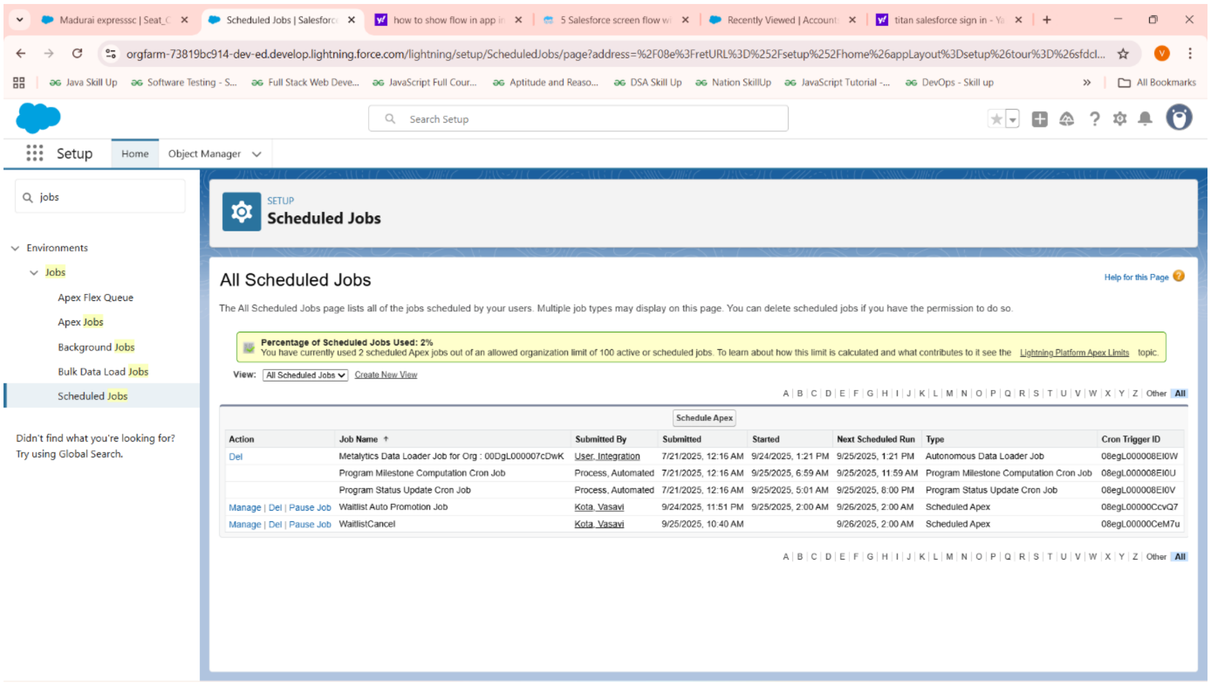
Percent of Apex Used: 0.17%
You are currently using 10,188 characters of Apex Code (excluding comments and @isTest annotated classes) in your organization, out of an allowed limit of 6,000,000 characters. Note that the amount in use includes both Apex Classes and Triggers defined in your organization.

Action	Name	Namespace Prefix	Api Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
Edit | Del | Security	BookingService		64.0	Active	5,736	Vasavi Kota, 9/25/2025, 9:33 AM	
Edit | Del | Security	WaitlistAutoCancel		64.0	Active	514	Vasavi Kota, 9/24/2025, 11:13 PM	
Edit | Del | Security	WaitlistAutoPromotion		64.0	Active	3,938	Vasavi Kota, 9/24/2025, 11:47 PM	

Dynamic Apex Classes

Dynamic Apex extends your programming reach by interacting with Lightning Platform components.

Class Name	Namespace Prefix	Api Version	Created By	Last Modified By



SETUP
Scheduled Jobs

All Scheduled Jobs

The All Scheduled Jobs page lists all of the jobs scheduled by your users. Multiple job types may display on this page. You can delete scheduled jobs if you have the permission to do so.

Percentage of Scheduled Jobs Used: 2%
You have currently used 2 scheduled Apex jobs out of an allowed organization limit of 100 active or scheduled jobs. To learn about how this limit is calculated and what contributes to it see the Lightning Platform Apex Limits topic.

Action	Job Name	Submitted By	Submitted	Started	Next Scheduled Run	Type	Cron Trigger ID
Del	Metalytics Data Loader Job for Org : 00DgL000007cDwK	User, Integration	7/21/2025, 12:16 AM	9/24/2025, 1:21 PM	9/25/2025, 1:21 PM	Autonomous Data Loader Job	08egL000008EI0W
	Program Milestone Computation Cron Job	Process, Automated	7/21/2025, 12:16 AM	9/25/2025, 6:59 AM	9/25/2025, 11:59 AM	Program Milestone Computation Cron Job	08egL000008EI0U
	Program Status Update Cron Job	Process, Automated	7/21/2025, 12:16 AM	9/25/2025, 5:01 AM	9/25/2025, 8:00 PM	Program Status Update Cron Job	08egL000008EI0V
Manage | Del | Pause Job	Waitlist Auto Promotion Job	Kota, Vasavi	9/24/2025, 11:51 PM	9/25/2025, 2:00 AM	9/26/2025, 2:00 AM	Scheduled Apex	08egL00000CcvQ7
Manage | Del | Pause Job	WaitlistCancel	Kota, Vasavi	9/25/2025, 10:40 AM		9/26/2025, 2:00 AM	Scheduled Apex	08egL00000CeM7u

```
public class BookingService {
   @AuraEnabled
   public static String cancelBooking(Id bookingId) {
      Booking__c b = [SELECT Id, Passenger__c, Amount_Deducted__c, Status__c, Journey_Date__c F
ROM Booking__c WHERE Id = :bookingId FOR UPDATE];
      if (b.Status__c == 'Cancelled') {
         return 'Booking already cancelled.';
      }
```

```
        Date today = Date.today();
        Integer daysToJourney = b.Journey_Date__c.daysBetween(today);

        if (daysToJourney < 2) {
            return 'Cannot cancel booking less than 2 days before journey.';
        }

        b.Status__c = 'Cancelled';

        Passenger__c p = [SELECT Id, Wallet_Balance__c FROM Passenger__c WHERE Id = :b.Passenger_
_c FOR UPDATE];

        if (b.Amount_Deducted__c != null && b.Amount_Deducted__c > 0) {
            p.Wallet_Balance__c += b.Amount_Deducted__c;

            Wallet_Transaction__c wt = new Wallet_Transaction__c(
                Passenger__c = p.Id,
                Amount__c = b.Amount_Deducted__c,
                Transaction_Type__c = 'Refund',
                Transaction_Date__c = Datetime.now(),
                Balance_After__c = p.Wallet_Balance__c,
                Reference__c = b.Name
            );
            update p;
            insert wt;
        }

        update b;
        return 'Booking cancelled successfully.';
    }

    public class BookingResult {
        @InvocableVariable public Boolean success;
        @InvocableVariable public String message;
        @InvocableVariable public String pnr;
        @InvocableVariable public String seatNumber;
    }

    @InvocableMethod(label='Create Booking')
    public static List<BookingResult> createBooking(List<Request> requests){
        List<BookingResult> results = new List<BookingResult>();
        for(Request req : requests){
            BookingResult br = new BookingResult();
            br.success = false;
            Passenger__c p = [SELECT Id, Wallet_Balance__c FROM Passenger__c WHERE Id = :req.passen
gerId FOR UPDATE];
            Seat_Class__c sc = [SELECT Id, Available_Seats__c, Price__c, Train__c, Total_Seats__c FROM S
eat_Class__c WHERE Id = :req.seatClassId FOR UPDATE];
```

```
        Train__c t = [SELECT Id, Available_Seats__c, Price__c FROM Train__c WHERE Id = :sc.Train__c
FOR UPDATE];
        Decimal fare = sc.Price__c != null ? sc.Price__c : t.Price__c;
        if(sc.Available_Seats__c > 0 && t.Available_Seats__c > 0){
          if(p.Wallet_Balance__c >= fare){
            // Assign seat number randomly
            Set<Integer> bookedSeats = new Set<Integer>();
            for(Booking__c prev : [SELECT Seat_Number__c FROM Booking__c WHERE Seat_Class__c
= :sc.Id AND Journey_Date__c = :req.journeyDate]) {
              if(prev.Seat_Number__c != null) bookedSeats.add(Integer.valueOf(prev.Seat_Number_
_c));
            }
            List<Integer> availableSeats = new List<Integer>();
            for(Integer i = 1; i <= sc.Total_Seats__c; i++) {
              if(!bookedSeats.contains(i)) availableSeats.add(i);
            }
            Integer seatNumber = availableSeats.size() > 0 ? availableSeats[Math.mod(Math.abs(Cryp
to.getRandomInteger()), availableSeats.size())] : null;

            p.Wallet_Balance__c -= fare;
            Wallet_Transaction__c wt = new Wallet_Transaction__c(
              Passenger__c = p.Id,
              Amount__c = -fare,
              Transaction_Type__c = 'Deduction',
              Transaction_Date__c = Datetime.now(),
              Balance_After__c = p.Wallet_Balance__c,
              Reference__c = 'Booking'
            );
            Booking__c b = new Booking__c(
              Passenger__c = p.Id,
              Train__c = t.Id,
              Seat_Class__c = sc.Id,
              Status__c = 'Confirmed',
              Journey_Date__c = req.journeyDate,
              Price__c = fare,
              Amount_Deducted__c = fare,
              Seat_Number__c = String.valueOf(seatNumber)
            );
            sc.Available_Seats__c -= 1;
            t.Available_Seats__c -= 1;
            update p;
            insert wt;
            insert b;
            // Query inserted booking to get generated PNR (Name)
            Booking__c booked = [SELECT Name FROM Booking__c WHERE Id = :b.Id];

            update sc;
            update t;
```

```
                br.pnr = booked.Name;
                br.seatNumber = String.valueOf(seatNumber);
                br.success = true;
                br.message = 'Booking Confirmed: ' + booked.Name + ', Seat Number: ' + seatNumber;
            } else {
                br.message = 'Insufficient wallet balance.';
            }
        } else {
            Booking__c wb = new Booking__c(
                Passenger__c = p.Id,
                Train__c = t.Id,
                Seat_Class__c = sc.Id,
                Status__c = 'Waitlist',
                Journey_Date__c = req.journeyDate,
                Price__c = fare,
                Amount_Deducted__c = 0
            );
            insert wb;
            br.success = true;
            br.message = 'Added to waitlist. PNR: ' + wb.Name;
            br.pnr = wb.Name;
        }
        results.add(br);
    }
    return results;
}

public class Request {
    @InvocableVariable public Id passengerId;
    @InvocableVariable public Id trainId;
    @InvocableVariable public Id seatClassId;
    @InvocableVariable public Date journeyDate;
}
}
```

```
global class WaitlistAutoCancel implements Schedulable {
 global void execute(SchedulableContext sc) {
  Date checkDate = Date.today().addDays(1);
  List<Booking__c> waitlist = [SELECT Id, Passenger__c, Price__c FROM Booking__c WHERE Status__
c = 'Waitlist' AND Journey_Date__c = :checkDate];
  List<Booking__c> toCancel = new List<Booking__c>();
  for(Booking__c b : waitlist){
   b.Status__c = 'Cancelled';
```

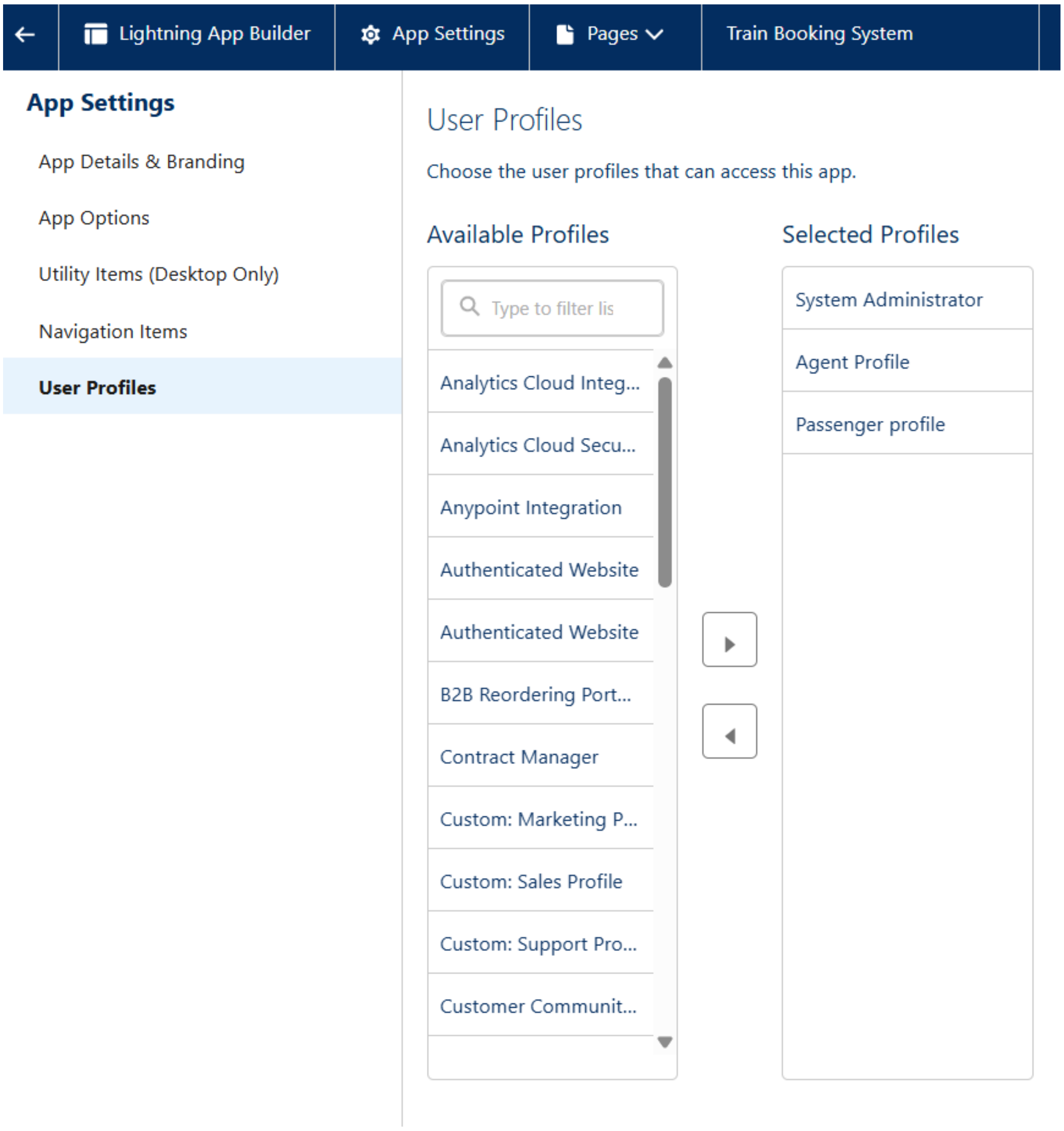
```
      toCancel.add(b);
      // Optionally, call an email utility here to notify the passenger and create refund Wallet_Transacti
on__c if needed.
    }
    if(!toCancel.isEmpty()) {
      update toCancel;
      // Optionally, send notification emails or process further logic.
    }
  }
}
```

```
global class WaitlistAutoPromotion implements Schedulable {

  global void execute(SchedulableContext sc) {
    Date checkDate = Date.today().addDays(1); // day before journey

    // Query all cancelled bookings freeing seats
    List<Booking__c> cancelledBookings = [
      SELECT Train__c, Seat_Class__c, Journey_Date__c
      FROM Booking__c
      WHERE Status__c = 'Cancelled' AND Journey_Date__c = :checkDate
    ];

    // Map to count freed seats by train + seat class
    Map<String, Integer> freedSeatsCount = new Map<String, Integer>();
    for (Booking__c b : cancelledBookings) {
      String key = b.Train__c + ':' + b.Seat_Class__c;
        Integer currentCount = freedSeatsCount.containsKey(key) ? freedSeatsCount.get(key) : 0;
freedSeatsCount.put(key, currentCount + 1);

    }

    // Process each train + seat class combination
    for (String key : freedSeatsCount.keySet()) {
      String[] parts = key.split(':');
      Id trainId = parts[0];
      Id seatClassId = parts[1];
      Integer seatsToPromote = freedSeatsCount.get(key);

      // Query waitlist bookings FIFO order
      List<Booking__c> waitlistBookings = [SELECT Id, Passenger__c, Journey_Date__c FROM Booking_
_c
                                WHERE Status__c = 'Waitlist' AND Train__c = :trainId AND Seat_Class__c = :se
atClassId
                                AND Journey_Date__c = :checkDate ORDER BY CreatedDate ASC LIMIT :seats
```

```
ToPromote];

    // Bulk query passengers and seat class/train details for efficiency
    Set<Id> passengerIds = new Set<Id>();
    for(Booking__c wb : waitlistBookings) passengerIds.add(wb.Passenger__c);
    Map<Id, Passenger__c> passengers = new Map<Id, Passenger__c>([
      SELECT Id, Wallet_Balance__c FROM Passenger__c WHERE Id IN :passengerIds FOR UPDATE
    ]);
    Seat_Class__c scs = [SELECT Id, Available_Seats__c, Train__c, Total_Seats__c, Price__c FROM Sea
t_Class__c WHERE Id = :seatClassId FOR UPDATE];
    Train__c train = [SELECT Id, Available_Seats__c, Price__c FROM Train__c WHERE Id = :trainId FOR
 UPDATE];

    Decimal fare = scs.Price__c != null ? scs.Price__c : train.Price__c;
    List<Booking__c> toUpdateBookings = new List<Booking__c>();
    List<Wallet_Transaction__c> walletTransactions = new List<Wallet_Transaction__c>();

    Integer seatsPromoted = 0;
    for (Booking__c waitlist : waitlistBookings) {
      Passenger__c p = passengers.get(waitlist.Passenger__c);
      if (p != null && p.Wallet_Balance__c >= fare && seatsPromoted < seatsToPromote) {

        // Find and assign next available seat number
        Set<Integer> bookedSeats = new Set<Integer>();
        for(Booking__c b : [SELECT Seat_Number__c FROM Booking__c WHERE Seat_Class__c = :scs.Id
AND Journey_Date__c = :waitlist.Journey_Date__c AND Status__c='Confirmed']) {
          if (b.Seat_Number__c != null) bookedSeats.add(Integer.valueOf(b.Seat_Number__c));
        }
        List<Integer> availableSeats = new List<Integer>();
        for (Integer i = 1; i <= scs.Total_Seats__c; i++) {
          if (!bookedSeats.contains(i)) availableSeats.add(i);
        }
        if (availableSeats.isEmpty()) break; // no seats to assign, break

        Integer seatNumber = availableSeats[Math.mod(Math.abs(Crypto.getRandomInteger()), availa
bleSeats.size())];

        // Deduct fare from wallet
        p.Wallet_Balance__c -= fare;

        // Update booking to confirmed
        waitlist.Status__c = 'Confirmed';
        waitlist.Amount_Deducted__c = fare;
        waitlist.Price__c = fare;
        waitlist.Seat_Number__c = String.valueOf(seatNumber);

        // Track records for update/insert
        toUpdateBookings.add(waitlist);
        walletTransactions.add(new Wallet_Transaction__c(
          Passenger__c = p.Id,
          Amount__c = -fare,
          Transaction_Type__c = 'Deduction',
          Transaction_Date__c = Datetime.now(),
          Balance_After__c = p.Wallet_Balance__c,
          Reference__c = 'Booking Promotion'
        ));

        seatsPromoted++;
        scs.Available_Seats__c -= 1;
        train.Available_Seats__c -= 1;
      } else {
        // Optionally notify passenger for insufficient balance or handle per policy
      }
    }

    if (!toUpdateBookings.isEmpty()) {
      update toUpdateBookings;
      update new List<Passenger__c>(passengers.values());
      insert walletTransactions;
      update scs;
      update train;
    }
  }
 }
}
```

Phase 6: User Interface Development

- Lightning App Builder
- Record Pages
- Tabs
- Home Page Layouts
- Utility Bar
- LWC (Lightning Web Components)
- Apex with LWC
- Events in LWC
- Wire Adapters
- Imperative Apex Calls
- Navigation Service



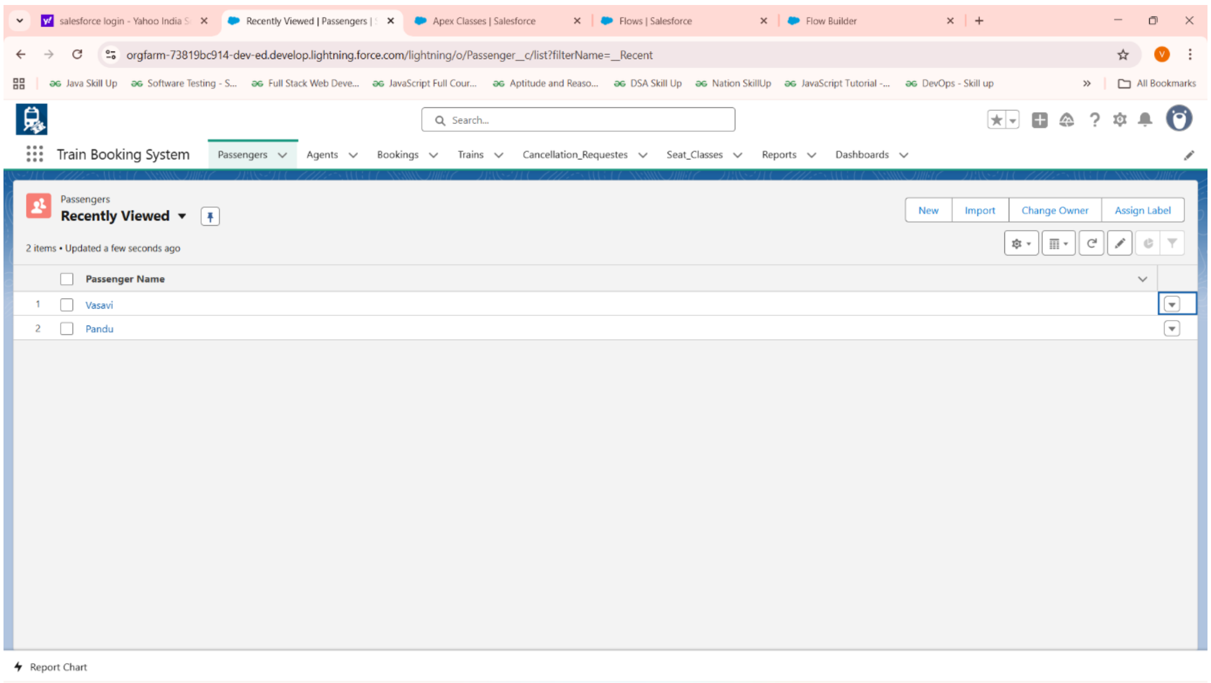
App Settings

- App Details & Branding
- App Options
- Utility Items (Desktop Only)
- Navigation Items
- **User Profiles**

User Profiles

Choose the user profiles that can access this app.

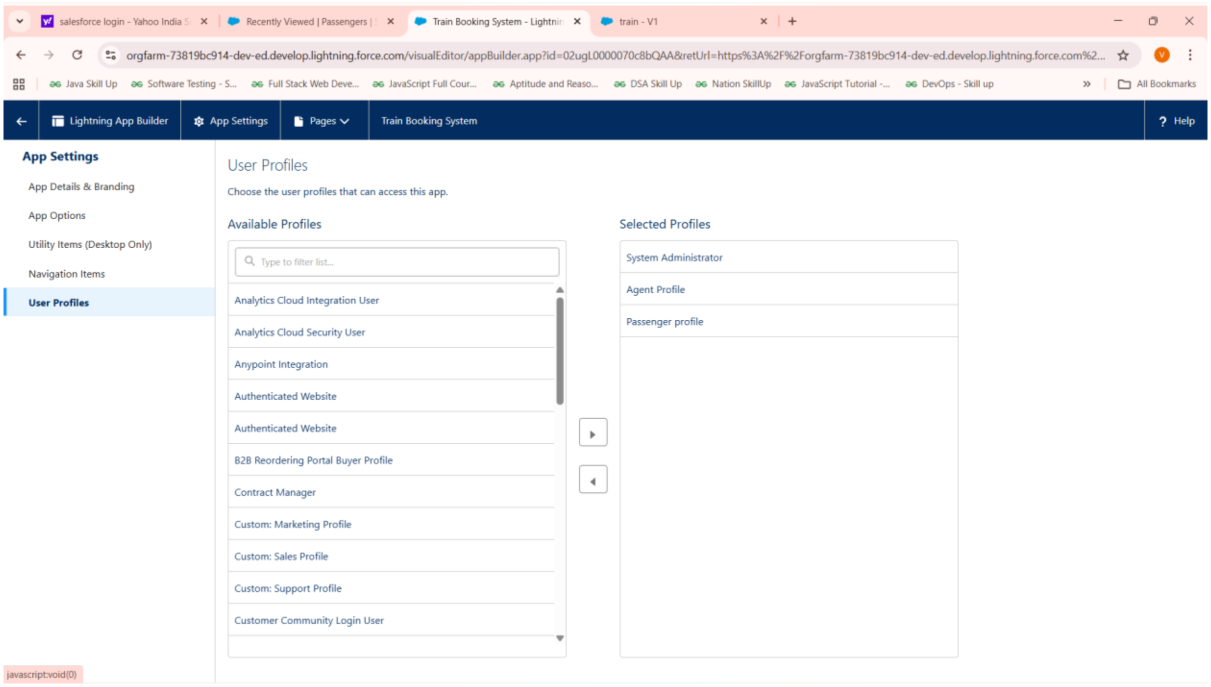
Available Profiles	Selected Profiles
Analytics Cloud Integ...	System Administrator
Analytics Cloud Secu...	Agent Profile
Anypoint Integration	Passenger profile
Authenticated Website	
Authenticated Website	
B2B Reordering Port...	
Contract Manager	
Custom: Marketing P...	
Custom: Sales Profile	
Custom: Support Pro...	
Customer Communit...	



Passengers
Recently Viewed

2 items • Updated a few seconds ago

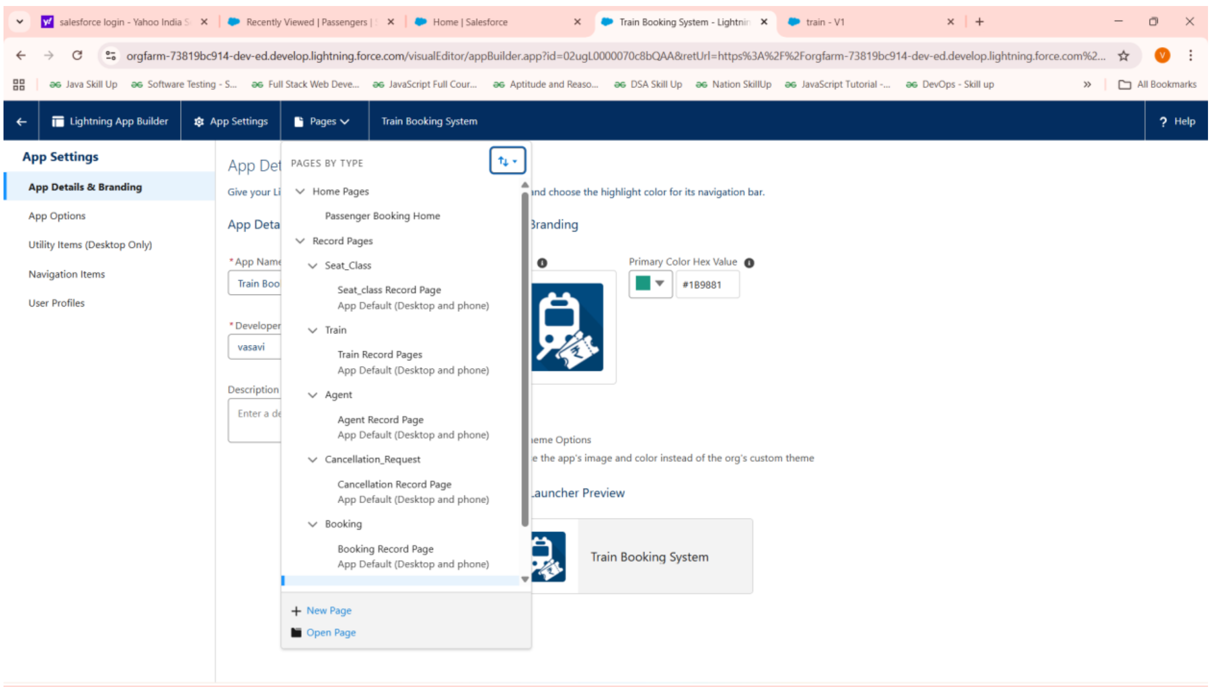
	Passenger Name
1	Vasavi
2	Pandu



User Profiles

Choose the user profiles that can access this app.

Available Profiles	Selected Profiles
Analytics Cloud Integration User	System Administrator
Analytics Cloud Security User	Agent Profile
Anypoint Integration	Passenger profile
Authenticated Website	
Authenticated Website	
B2B Reordering Portal Buyer Profile	
Contract Manager	
Custom: Marketing Profile	
Custom: Sales Profile	
Custom: Support Profile	
Customer Community Login User	


Lightning App Builder | App Settings | Pages | Train Booking System

App Settings

- App Details & Branding
- App Options
- Utility Items (Desktop Only)
- Navigation Items
- User Profiles

PAGES BY TYPE

- Home Pages
  - Passenger Booking Home
- Record Pages
  - Seat_Class
    - Seat_class Record Page
      App Default (Desktop and phone)
  - Train
    - Train Record Pages
      App Default (Desktop and phone)
  - Agent
    - Agent Record Page
      App Default (Desktop and phone)
  - Cancellation_Request
    - Cancellation Record Page
      App Default (Desktop and phone)
  - Booking
    - Booking Record Page
      App Default (Desktop and phone)

+ New Page
Open Page

App Det… Give your … and choose the highlight color for its navigation bar.

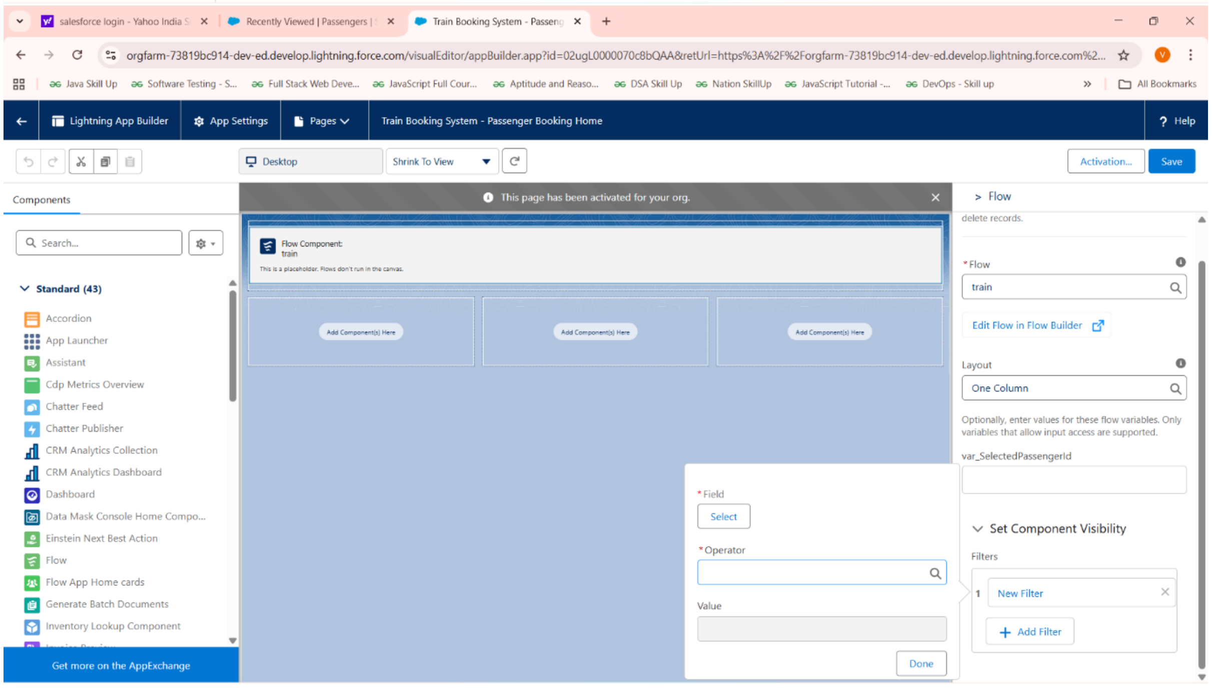
App Deta… Branding

*App Name: Train Boo…
*Developer: vasavi
Description: Enter a de…

Primary Color Hex Value: #1B9881

…eme Options
…e the app's image and color instead of the org's custom theme

…auncher Preview
Train Booking System


Lightning App Builder | App Settings | Pages | Train Booking System - Passenger Booking Home | Help

Desktop | Shrink To View | Activation... | Save

This page has been activated for your org.

Flow Component:
train
This is a placeholder. Flows don't run in the canvas.

Add Component(s) Here | Add Component(s) Here | Add Component(s) Here

Components

Standard (43)
- Accordion
- App Launcher
- Assistant
- Cdp Metrics Overview
- Chatter Feed
- Chatter Publisher
- CRM Analytics Collection
- CRM Analytics Dashboard
- Dashboard
- Data Mask Console Home Compo…
- Einstein Next Best Action
- Flow
- Flow App Home cards
- Generate Batch Documents
- Inventory Lookup Component

Get more on the AppExchange

> Flow
delete records.

*Flow: train
Edit Flow in Flow Builder

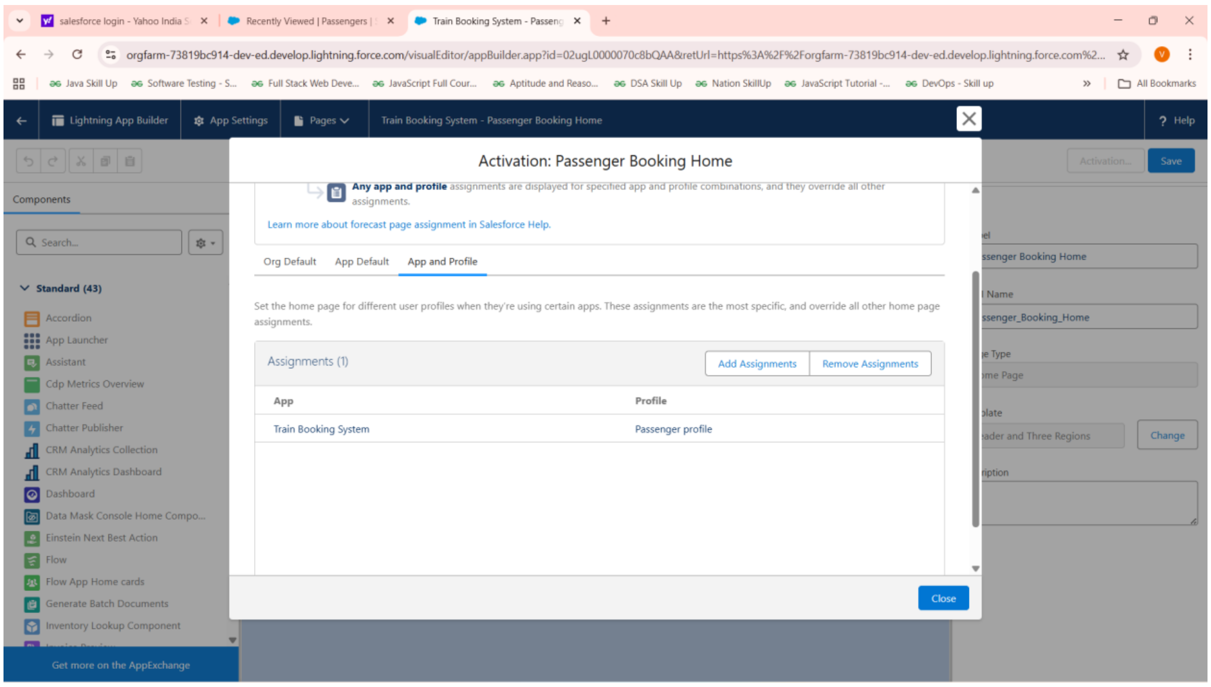
Layout: One Column

Optionally, enter values for these flow variables. Only variables that allow input access are supported.

var_SelectedPassengerId:

Set Component Visibility
Filters
1 New Filter
+ Add Filter

*Field: Select
*Operator:
Value:
Done



Activation: Passenger Booking Home

Any app and profile assignments are displayed for specified app and profile combinations, and they override all other assignments.

Learn more about forecast page assignment in Salesforce Help.

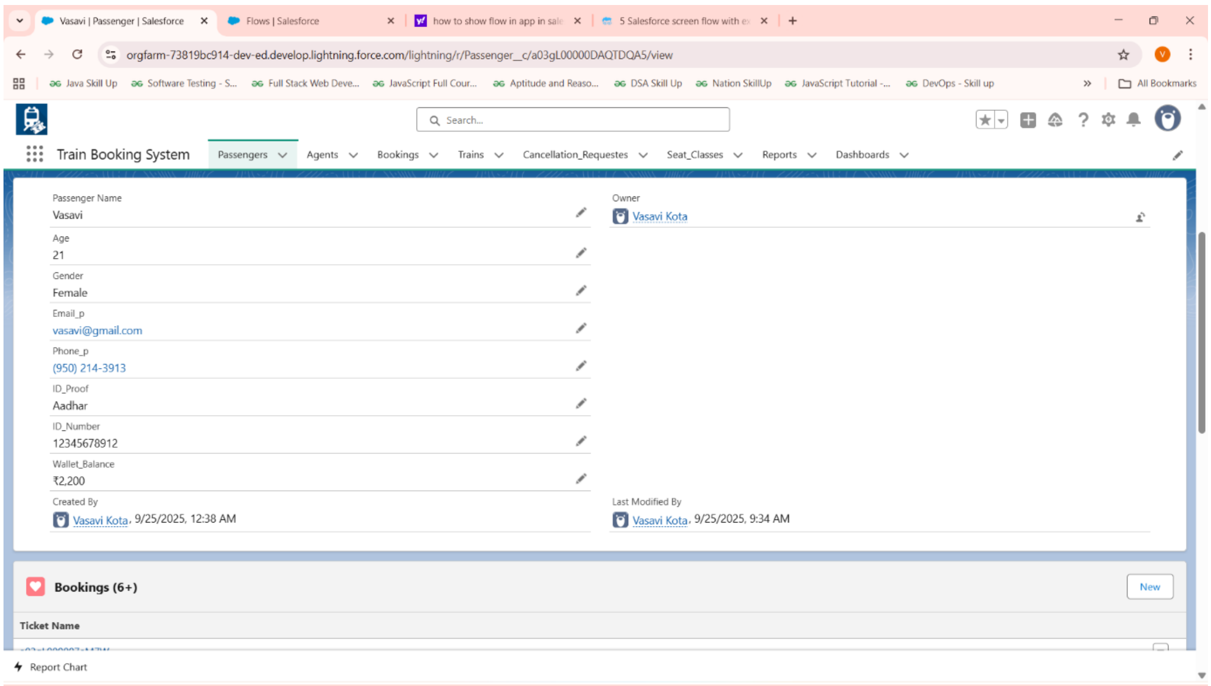
Org Default | App Default | App and Profile

Set the home page for different user profiles when they're using certain apps. These assignments are the most specific, and override all other home page assignments.

Assignments (1)

App	Profile
Train Booking System	Passenger profile

orgfarm-73819bc914-dev-ed.develop.lightning.force.com/visualEditor/appBuilder.app?id=02ugL0000070c8bQAA&retUrl=https%3A%2F%2Forgfarm-73819bc914-dev-ed.develop.lightning.force.com%2...


Train Booking System

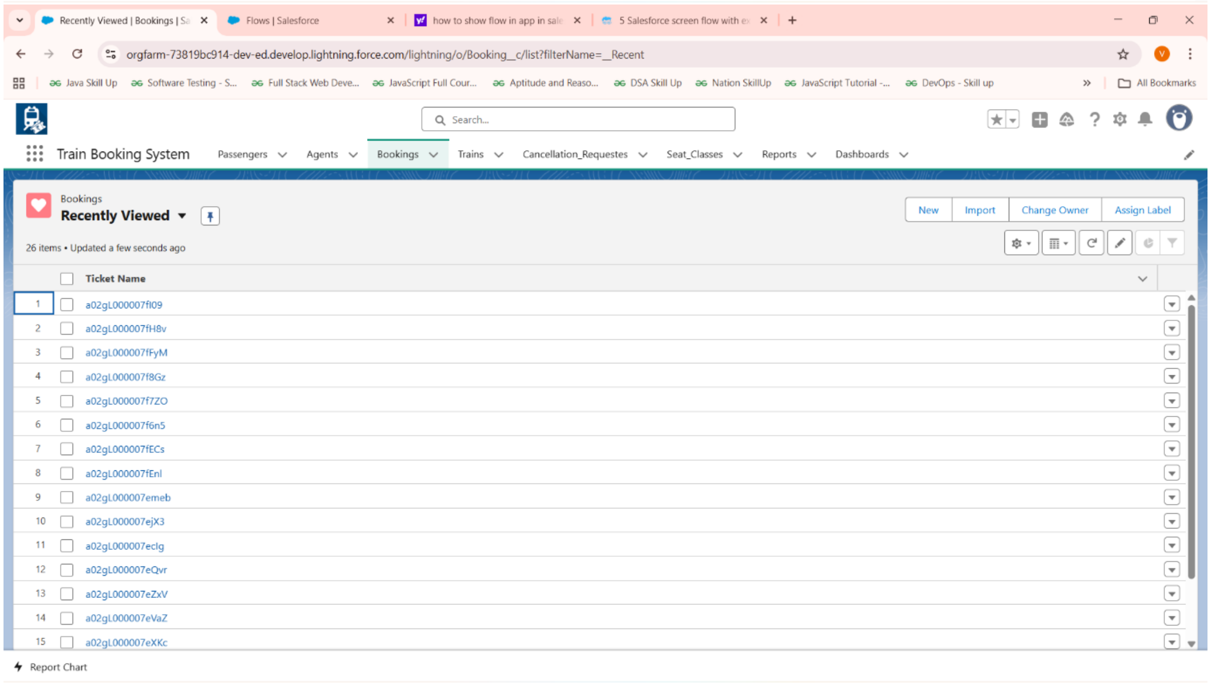
Passengers | Agents | Bookings | Trains | Cancellation_Requestes | Seat_Classes | Reports | Dashboards

Passenger Name	Vasavi
Age	21
Gender	Female
Email_p	vasavi@gmail.com
Phone_p	(950) 214-3913
ID_Proof	Aadhar
ID_Number	12345678912
Wallet_Balance	₹2,200
Created By	Vasavi Kota, 9/25/2025, 12:38 AM
Owner	Vasavi Kota
Last Modified By	Vasavi Kota, 9/25/2025, 9:34 AM

Bookings (6+)

Ticket Name

Report Chart


Train Booking System

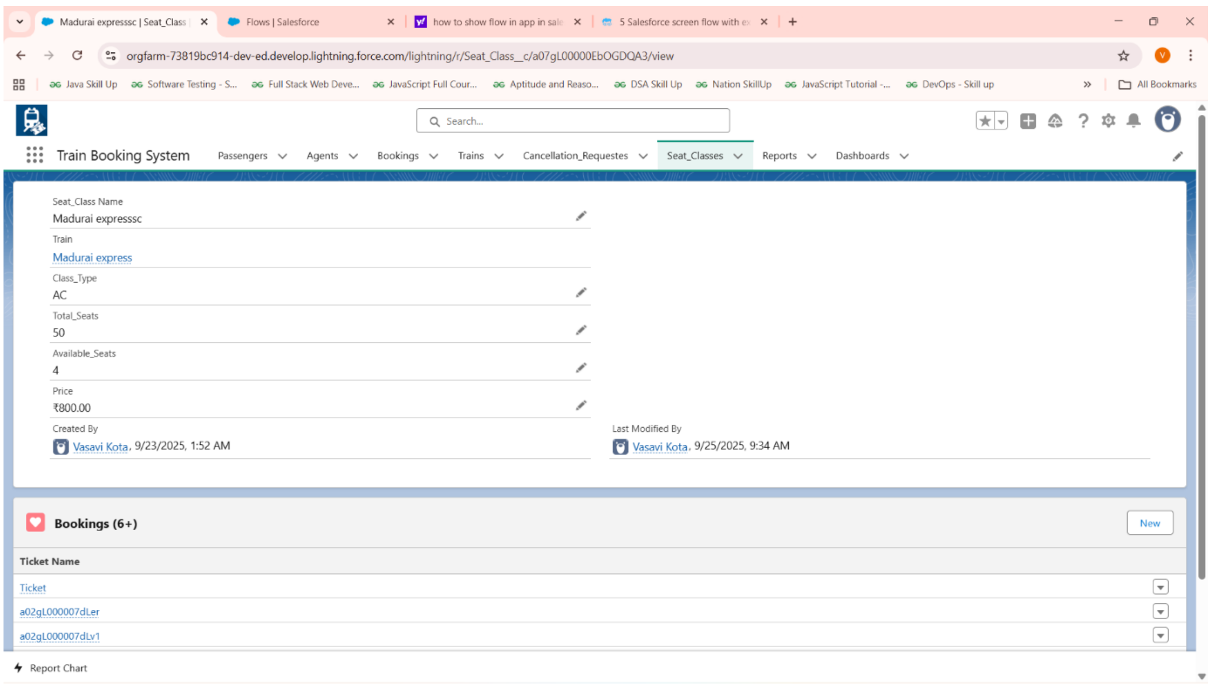
Passengers | Agents | Bookings | Trains | Cancellation_Requestes | Seat_Classes | Reports | Dashboards

Bookings
Recently Viewed

26 items • Updated a few seconds ago

	Ticket Name
1	a02gL000007fI09
2	a02gL000007fH8v
3	a02gL000007fFyM
4	a02gL000007f8Gz
5	a02gL000007f7ZO
6	a02gL000007f6n5
7	a02gL000007fECs
8	a02gL000007fEnl
9	a02gL000007emeb
10	a02gL000007ejX3
11	a02gL000007eclg
12	a02gL000007eQvr
13	a02gL000007eZxV
14	a02gL000007eVaZ
15	a02gL000007eXKc

Report Chart



orgfarm-73819bc914-dev-ed.develop.lightning.force.com/lightning/r/Seat_Class__c/a07gL00000EbOGDQA3/view

Seat_Class Name: Madurai expresssc
Train: Madurai express
Class_Type: AC
Total_Seats: 50
Available_Seats: 4
Price: ₹800.00
Created By: Vasavi Kota, 9/23/2025, 1:52 AM
Last Modified By: Vasavi Kota, 9/25/2025, 9:34 AM

Bookings (6+)

Ticket Name
Ticket
a02gL000007dLer
a02gL000007dLv1

Overall summary:

Phase 4: Process Automation (Admin)

- Validation Rules: Implemented to enforce business rules like cancellation timing policies.
- Workflow Rules & Process Builder: Limited use; moved towards Flow Builder for UI and automation.
- Approval Process: Not implemented in current scope; can be added if needed.
- Flow Builder:
    - Created Screen Flows for Booking and Cancellation processes.
    - Used Record-Triggered and Scheduled Flows via Apex scheduled jobs for waitlist management.
- Email Alerts: Setup for booking confirmation, cancellation notifications (integration with Apex or Flow).
- Field Updates & Tasks: Integrated as part of Apex logic and Flow outcomes.
- Custom Notifications: Configured or considered for alerting users on booking statuses.

Phase 5: Apex Programming (Developer)

- Classes & Objects: Built core BookingService and Waitlist management Apex classes.
- Apex Triggers: Not extensively implemented; Apex classes handle logic.
- Trigger Design Patterns: Not covered but can be adopted for scale.

- SOQL & SOSL: Used for querying bookings, passengers, trains, seat classes efficiently.
- Collections: Utilized Lists, Sets, and Maps for data processing.
- Control Statements: Implemented complex booking logic and validations.
- Scheduled Apex: Developed WaitlistAutoCancel to cancel waitlisted tickets before 1 day from journeydate and WaitlistAutoPromotion for freed seats from cancellation to confirmed booking schedulable classes
- Exception Handling & Test Classes: Recommendations to enhance for production readiness.
- Asynchronous Processing: Scheduled Apex for batch/scheduled jobs.

Phase 6: User Interface Development

- Lightning App Builder: Used for building flow containers and app pages.
- Record Pages & Tabs: Setup for quick access to Booking and related objects.
- Home Page Layouts & Utility Bar: Customized for better UX.
- Lightning Web Components (LWC): Not implemented; but working on it will complete soon
- Apex with LWC Integration: Considered for complex UI functionality.
- Events, Wire Adapters, Imperative Apex Calls: Planned for advanced interaction.
- Navigation Service: Used for smooth UI transitions in Lightning apps.