**TITLE:TRAIN TICKET BOOKING MANAGEMENT SYSTEM**

**Project Overview:**

The **Railway Ticketing System CRM** is a Salesforce-based application designed to simplify and automate the process of booking and managing train tickets. The system allows passengers to register, book tickets, view their booking history, and request cancellations or refunds through a centralized platform. Administrators can manage trains, routes, schedules, and passenger records efficiently.

This CRM solution enhances transparency, reduces manual errors, and ensures that all booking and passenger data are stored securely within the Salesforce environment. The system leverages Salesforce’s capabilities like custom objects, workflows, and sharing rules to provide a user-friendly and reliable ticket management experience.

**Key Features:**

* Passenger registration and profile management.
* Train, route, and schedule management.
* Ticket booking, cancellation, and refund functionality.
* Role-based access for passengers and administrators.
* Real-time reporting and dashboards for monitoring bookings and availability.

**Business Need:**  
Railway operations often involve managing large volumes of passenger and ticket data, which can lead to inefficiencies when handled manually. This CRM automates booking and record management processes, improving operational efficiency, accuracy, and customer satisfaction.

## **Objectives of the Project**

1. **Automate the Ticket Booking Process** — Enable passengers to book and manage their train tickets easily within Salesforce.
2. **Improve Data Management** — Maintain a centralized and accurate database of trains, passengers, routes, and bookings.
3. **Enhance Customer Experience** — Provide passengers with an intuitive and seamless ticketing system.
4. **Ensure Secure Access** — Implement controlled data access through role-based permissions for passengers and administrators.
5. **Provide Insights Through Reports** — Create dashboards and analytical reports for administrators to monitor ticket sales, cancellations, and trends.
6. **Support Future Scalability** — Build a flexible data model that can accommodate additional features such as payment integration or mobile access in future phases.

**Phase 2: Org Setup Complete**

* Set up company profile and business hours.
* Created Admin, Agent, Passenger users with correct licenses.
* Made profiles, roles, and permission sets for secure access.
* Configured sharing rules and organization-wide defaults to protect ticket and booking data.
* Enabled admin login access for realistic user testing.​

**Phase 3: Data Model Built**

* Created custom objects: Train, Passenger, Ticket, Booking, Seat\_Class,.
* Added fields to objects, set up relationships using lookups and master-detail as needed.
* Created record types like Booked, Cancelled, Waitlisted for Tickets.
* Designed multiple page and compact layouts for different user profiles.
* Visualized and confirmed schema in Schema Builder.​

**Phase 4: Automation Implemented**

* Built validation rules and custom process automation using Flow Builder.
* Setup email alerts and notifications for booking status changes.
* Created screen flows for core booking .
* Scheduled flows and jobs for waitlist management (like auto-promotion/cancellation).
* Approval process considered while sending cancellation request

**Phase 5: Apex Coding Done**

* Wrote extensive Apex classes for handling booking, cancellation, wallet refunds, and waitlist.
* Developed schedulable jobs for waitlist auto-cancel and auto-promotion.
* Used SOQL/SOSL queries, collections, and advanced logic.
* Implemented exception handling, control statements, and testing recommendations.​

**Phase 6: UI Developed**

* Used Lightning App Builder to design pages and tabs for booking management.
* Customized home page layouts and utility bar for better UX.
* Built and planned Lightning Web Components (LWC); LWC integration for cancellation UI.​

**Phase 7: Data Management & Deployment Done**

* Imported data for objects using Data Import Wizard and scheduled regular backups.
* Used change sets for deployment of Apex classes, LWCs, and other components.​

**Phase 8: Packaging and Source Control Done**

* Used unmanaged packages for sharing customizations.
* Used managed packages for secure, upgradeable apps.
* Made use of VS Code SFDX for source-driven development, deploying, and managing LWC and Apex code.​

**Phase 9: Reporting, Dashboards, Security Done**

* Created different types of reports (tabular, summary, matrix, joined) for bookings and trains.
* Built dashboards for visualizing train bookings and stats.
* Set up sharing rules, field-level security, and session/login policies for safety (15-min timeout, MFA, IP restrictions).​

### Phase 10: Quality Assurance Testing

### BOOKING\_TESTCASES:

### TestCase 1:

### The passenger should be able to search the trains on that specific date from the desired source to destination and select a train and seat class

### Input: Enter Source, Destination,Date details

### Fig: Entered Source, Destination,Date and passenger details

### Expected Output:Trains on that day from 12am to 11:59pm from that source to destination and should be able to select a train and seat class

### Actual Output: Trains available 24hours from input date and time and can able to select a train and seat class

### Selecting the Train output:

### 

### Selecting the Seat –Class output:

### 

### Next collect the id Proof details even if the seats or available for booking or for waitlist booking

### Fig: Collecting Idproof Details:

### 

### Output for Collecting Id details of waitlist category :

### 

### SubtestCase -1:

### Input: If seats are available in the selected seat-class and has sufficient balance in the wallet

### Expected Output: Confirm Booking the ticket and assign a seat and Provide PNR details and update available seats in that seat-class ,update the wallet Balance and available seats

### Actual Output: Confirm Booking the ticket and assign a seat and Provide PNR details and update available seats in that seat-class, update the wallet Balance and available seats

### FIG:Enter details

### 

### Fig:select the train

### 

### FIG:Select the seat-class

### 

### FIG:Booking confirmation :

### 

### Balance before booking:

### 

### Balance After Booking :

### 

### Subtestcase-2:

### Input: If seats are available in the selected seatclass and If does’nt has sufficient balance in the wallet

### 

### Expected Output: Should Show Insufficient wallet balance message

### Actual Output: Displays Insufficient wallet balance message.

### 

### Subtestcase-3:

### Input:If does’nt have seats in the selected seat\_class

### Expected Output: Ask to maintain sufficient wallet balance and Waitlist the Booking and Provide waitlist PNR details,update waitlist count for that seatclass of that train

### Actual Output: Ask to maintain sufficient wallet balance and Waitlist the Booking and Provide waitlist PNR details, update waitlist count for that seatclass of that train

### 

### TestCase-2: Able to book the ticket for trains in the range of 1 to 15 days

### Input:Date which is greater than 15 days from today’s Date

### Expected output : Show error message and ask passenger to enter a valid date in the range

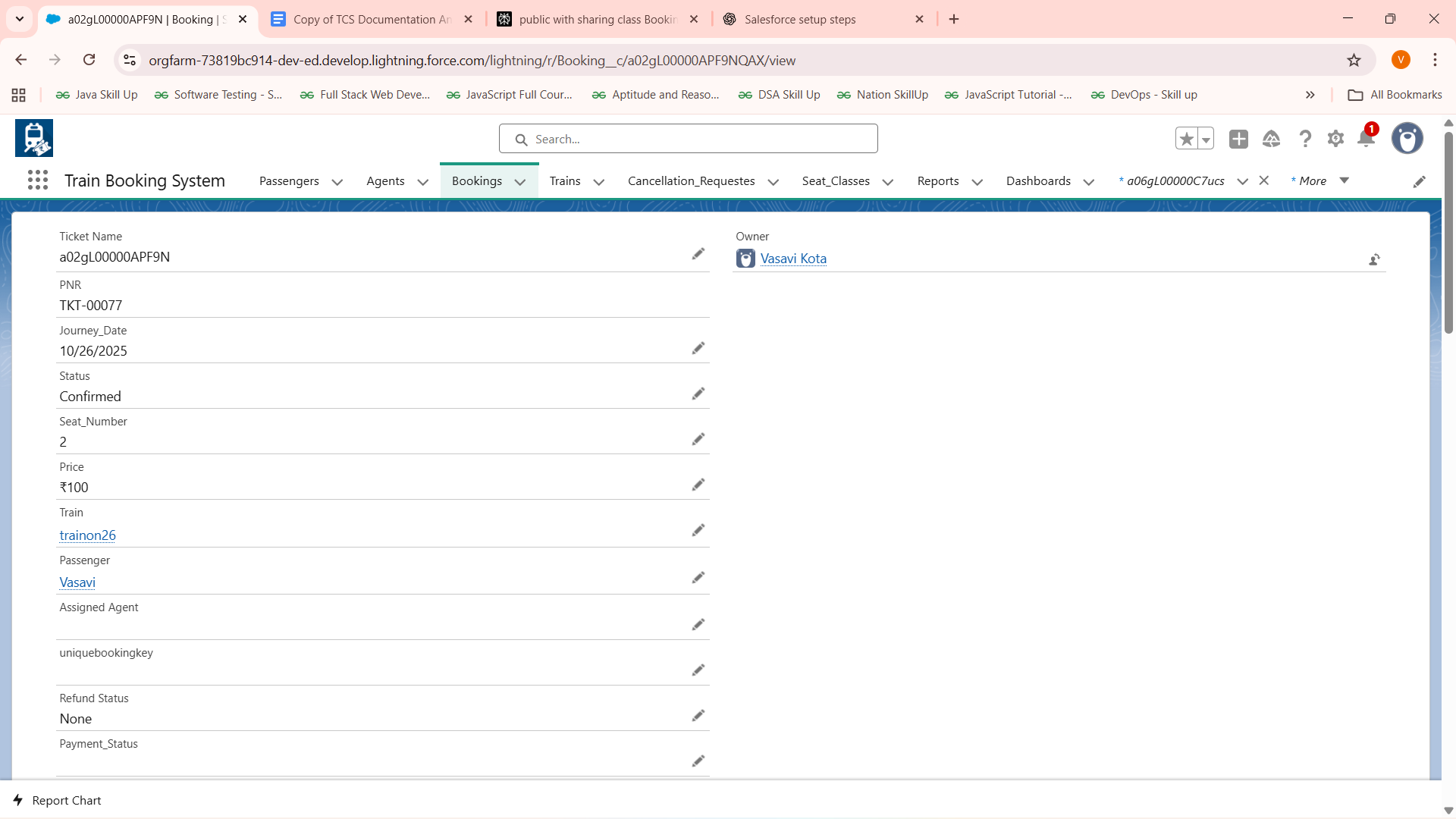
### Actual Output:

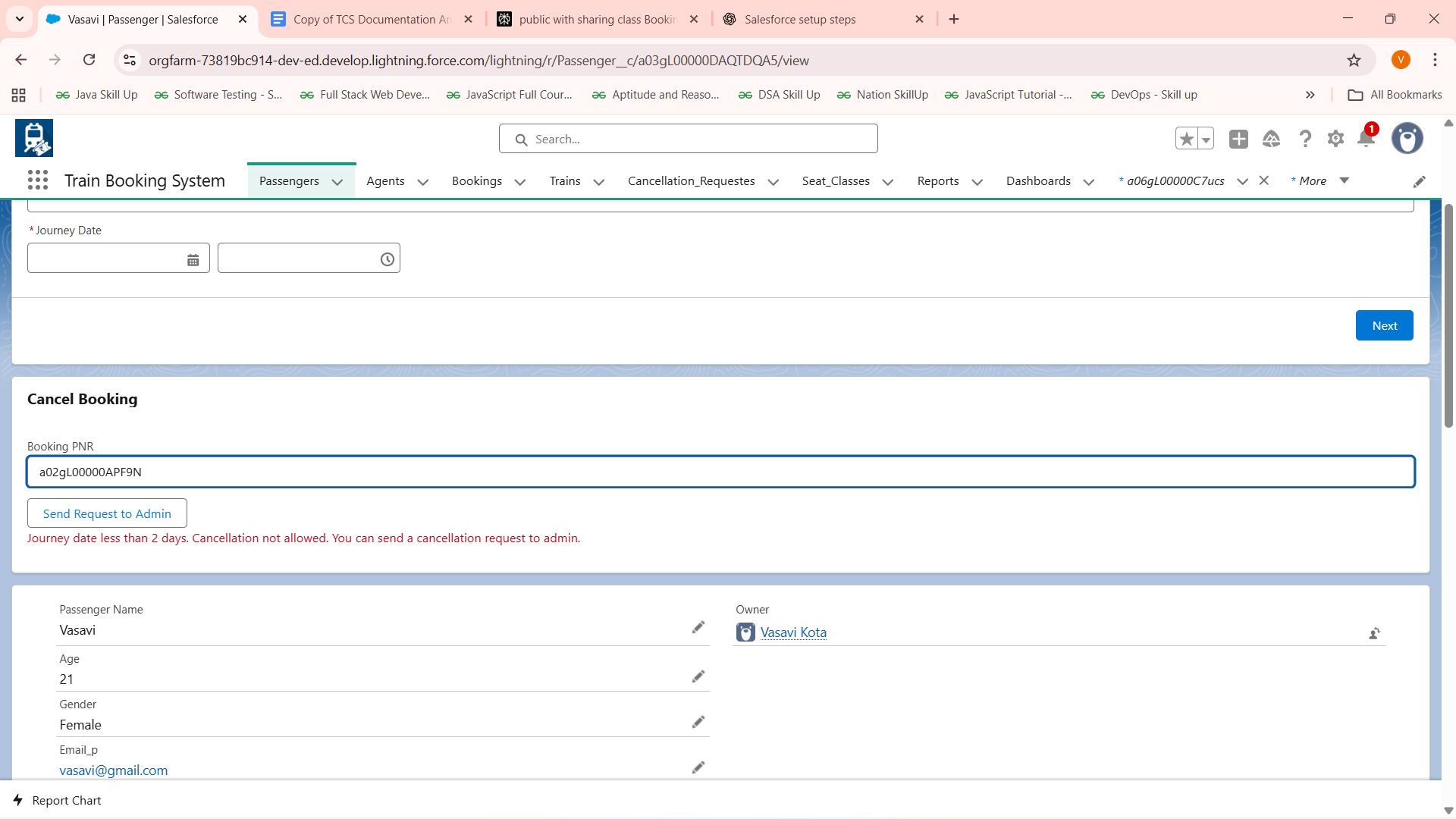
### 

**CANCELLATION\_PROCESS\_TESTCASES:**

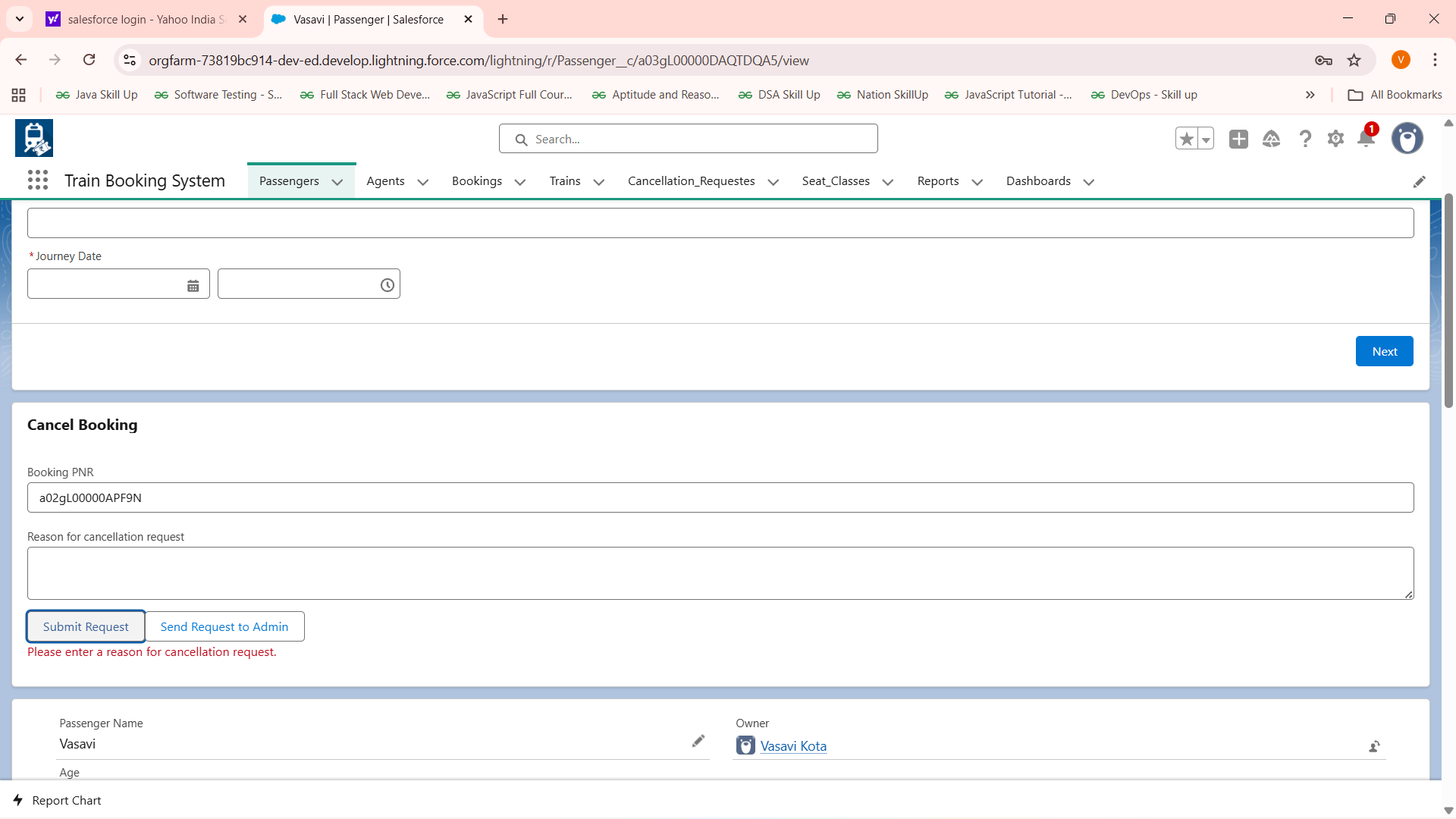
**TestCase-1:**Cancellation for confirmed ticket

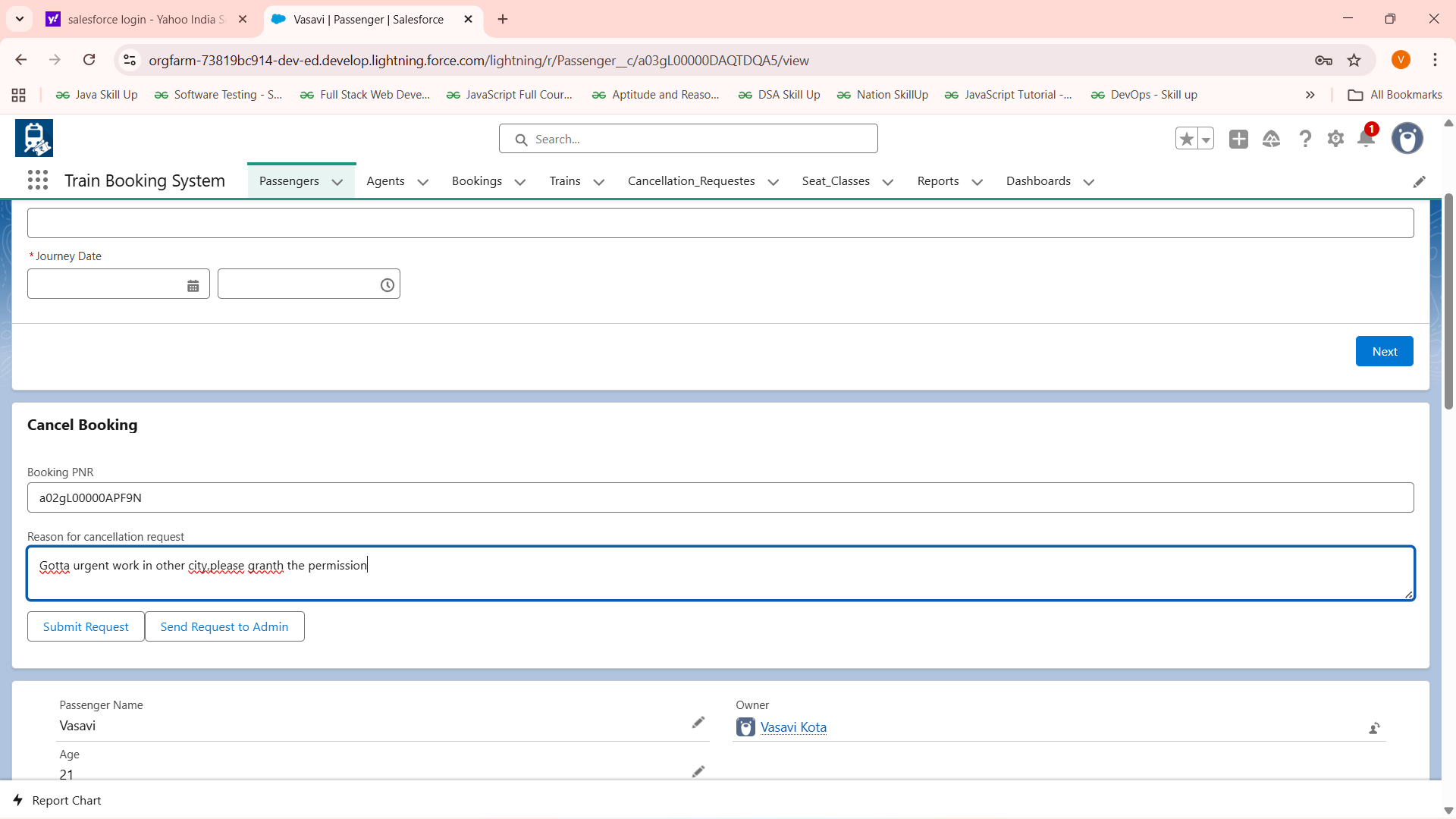
**Subtestcase-1**:Joumey Date less than 2 days

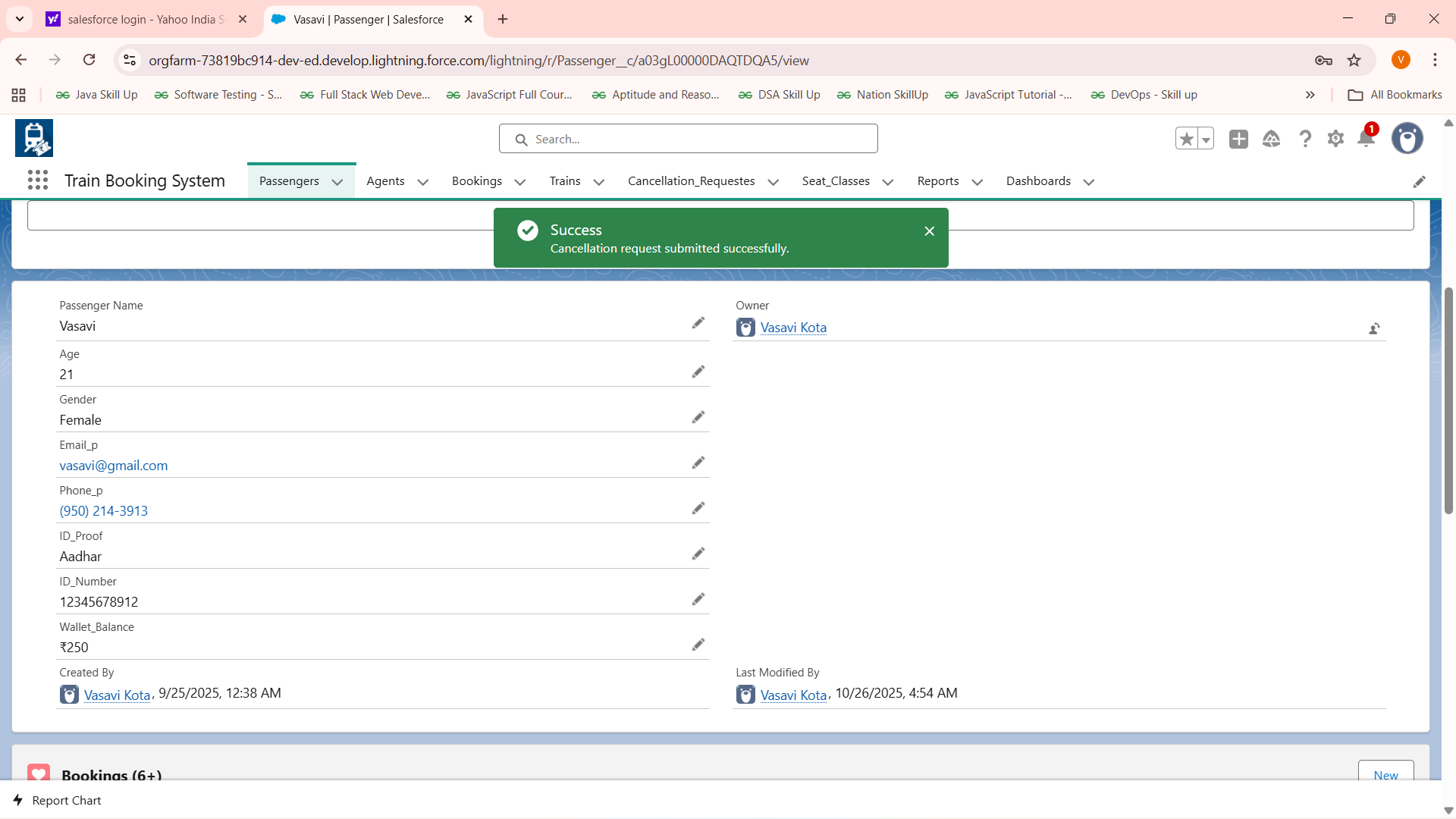


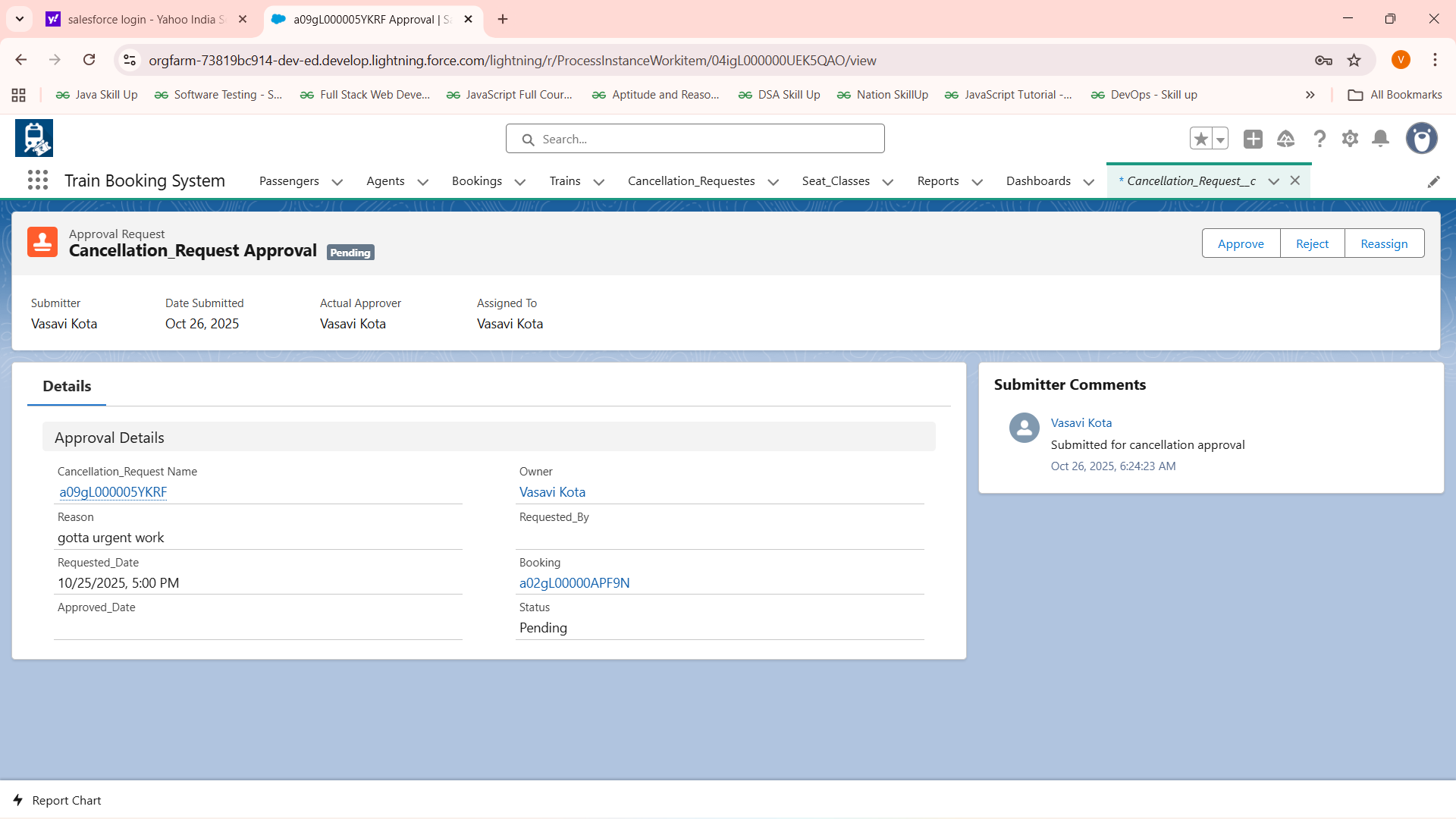


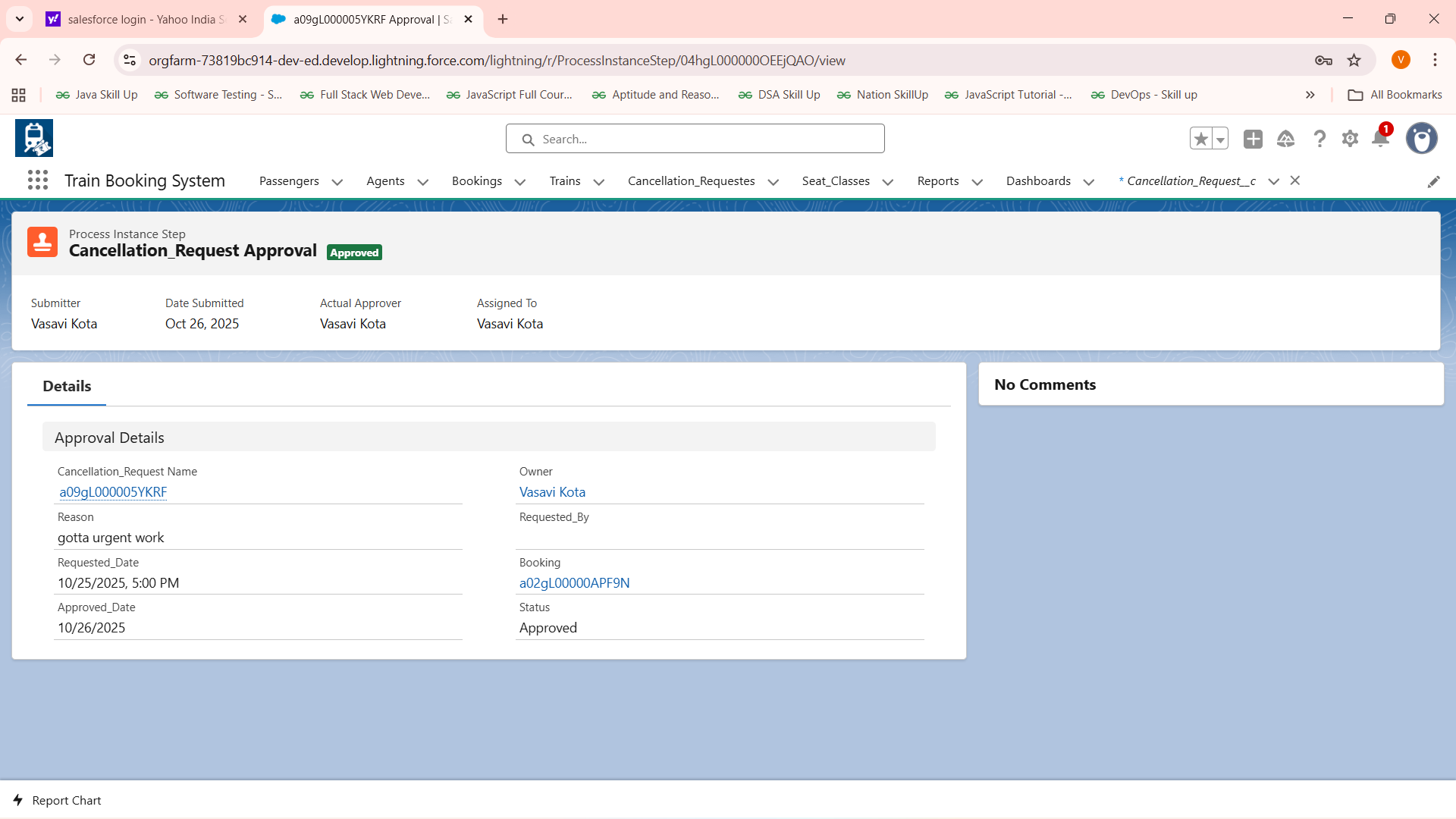
**So we can Send request to the Admin for cancellation request**





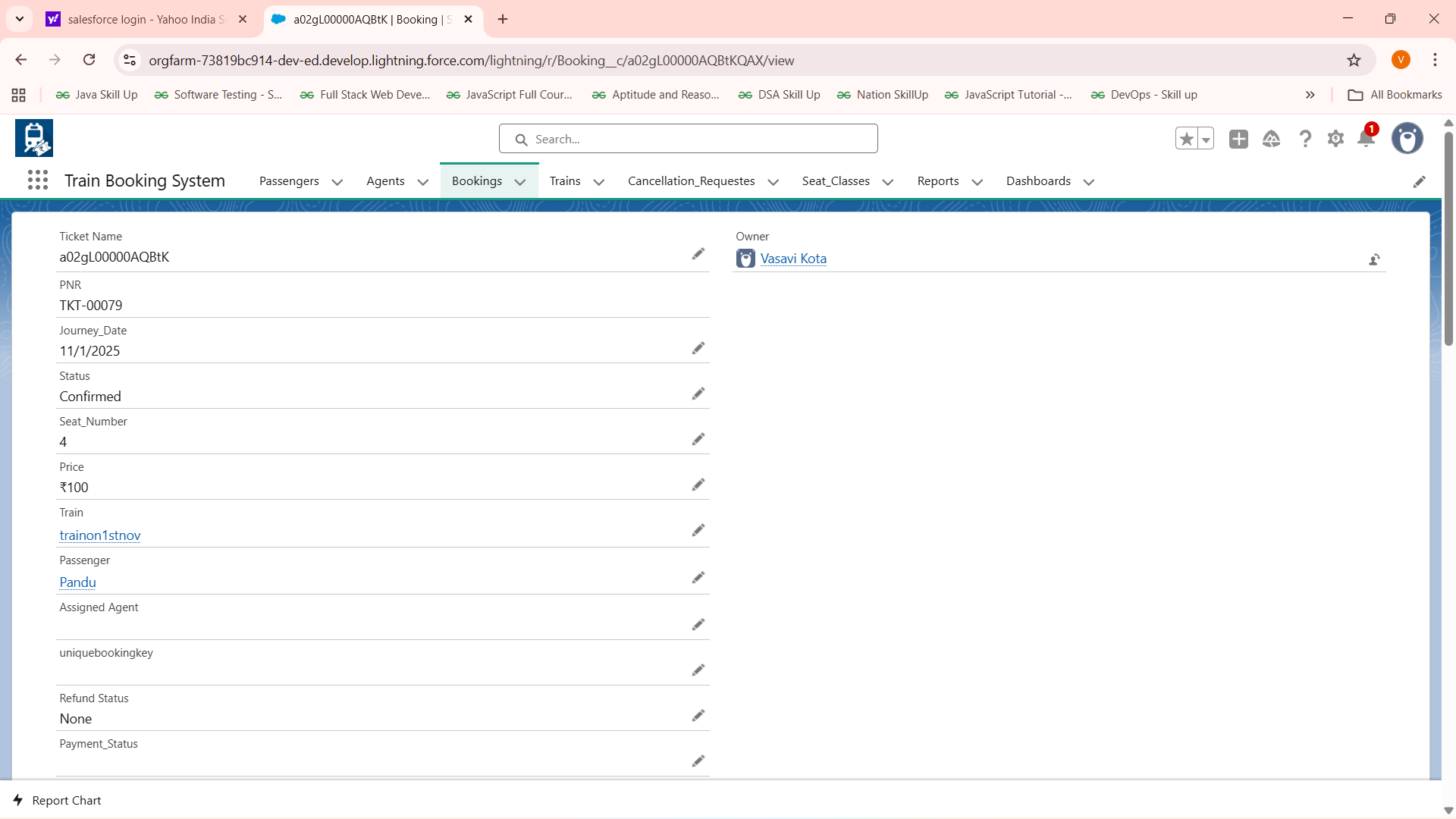






**Subtestcase-2:** IF Joumey Date greater >= 2 days and less than & 7 days so 75% refund and update available seats

Input: Confirmed booking PNR

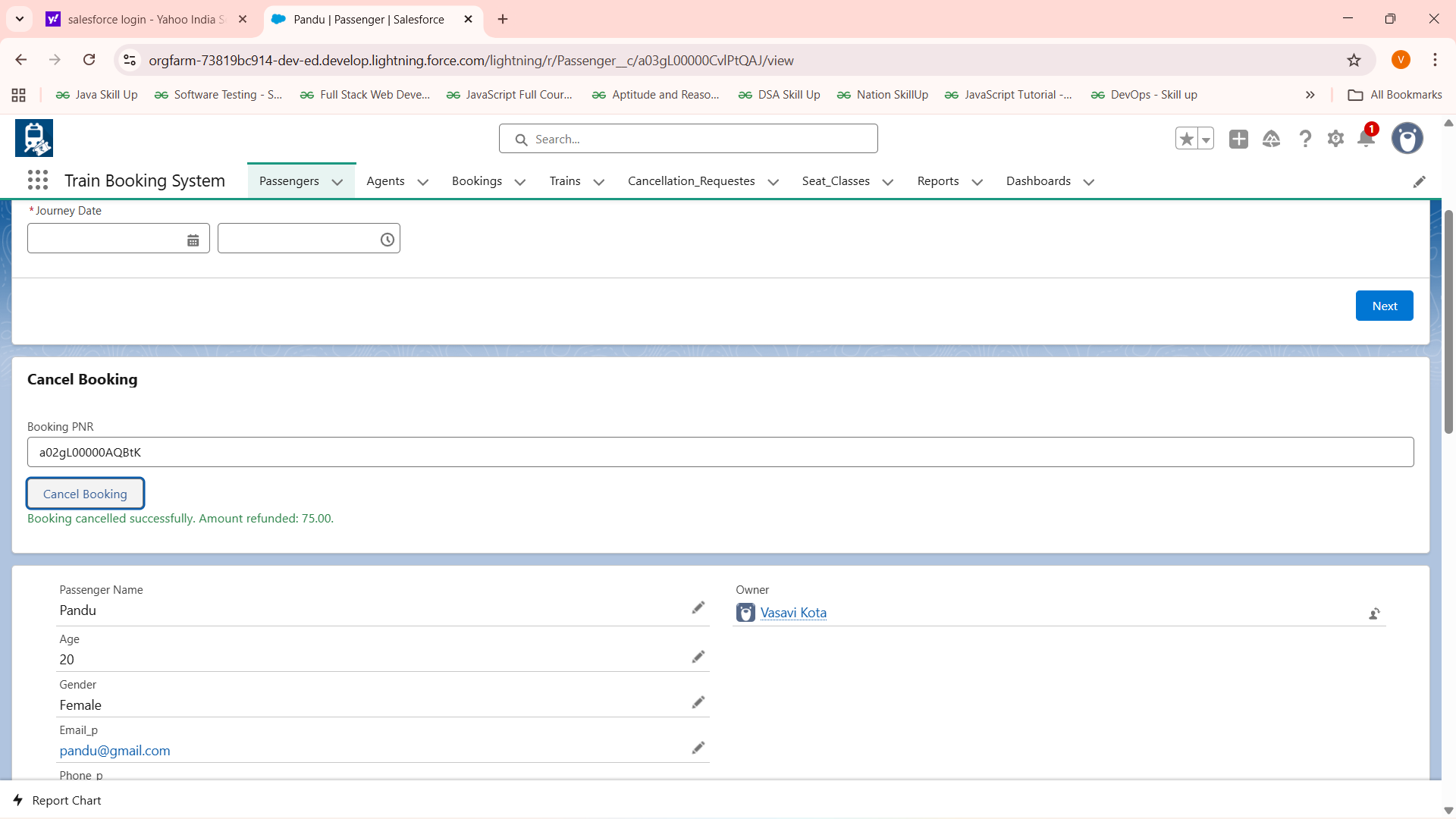


Expected Output: Joumey Date greater >= 2 days and less than & 7 days so 75% refund and update available seats

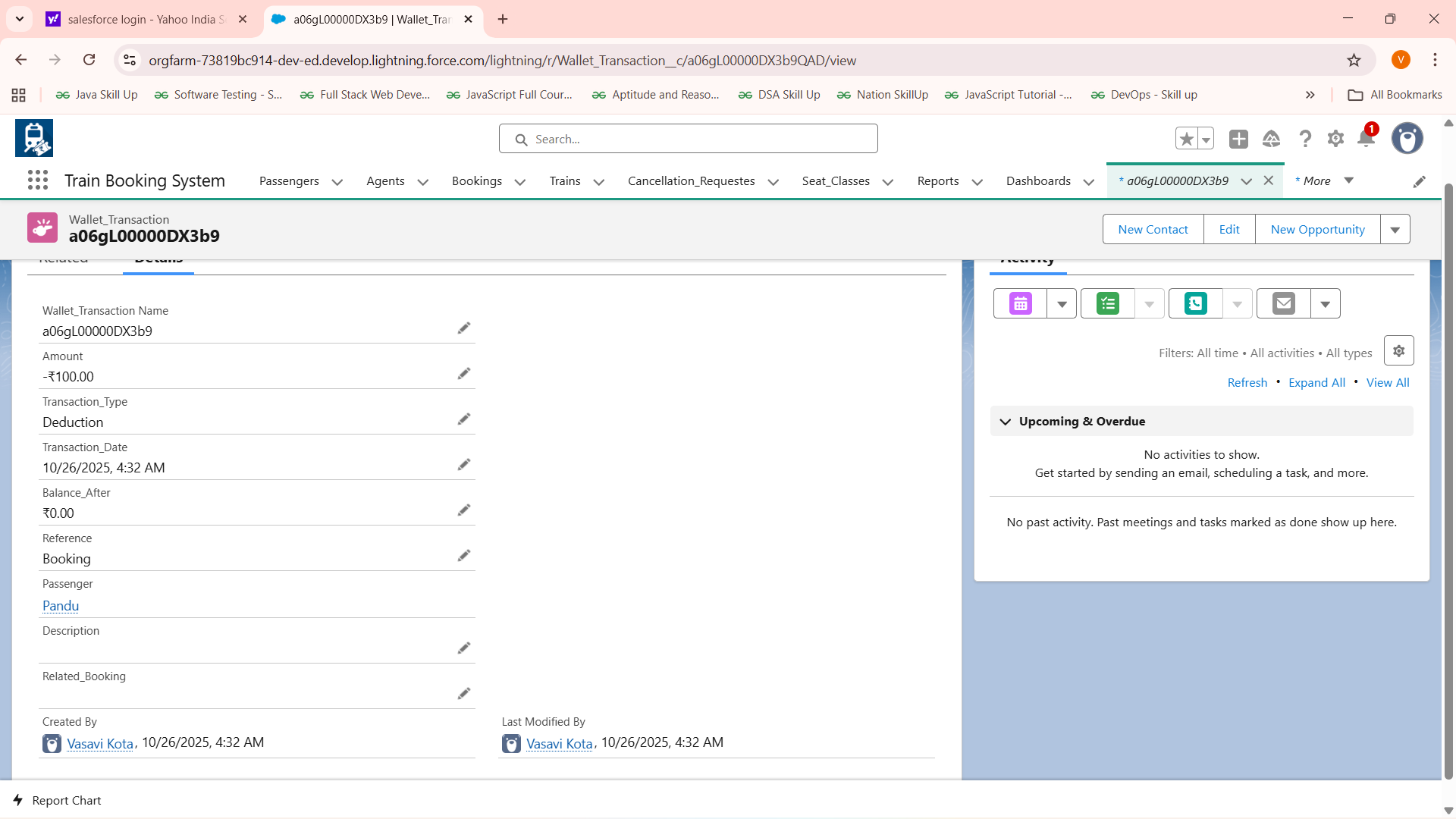
**Actual Output:**



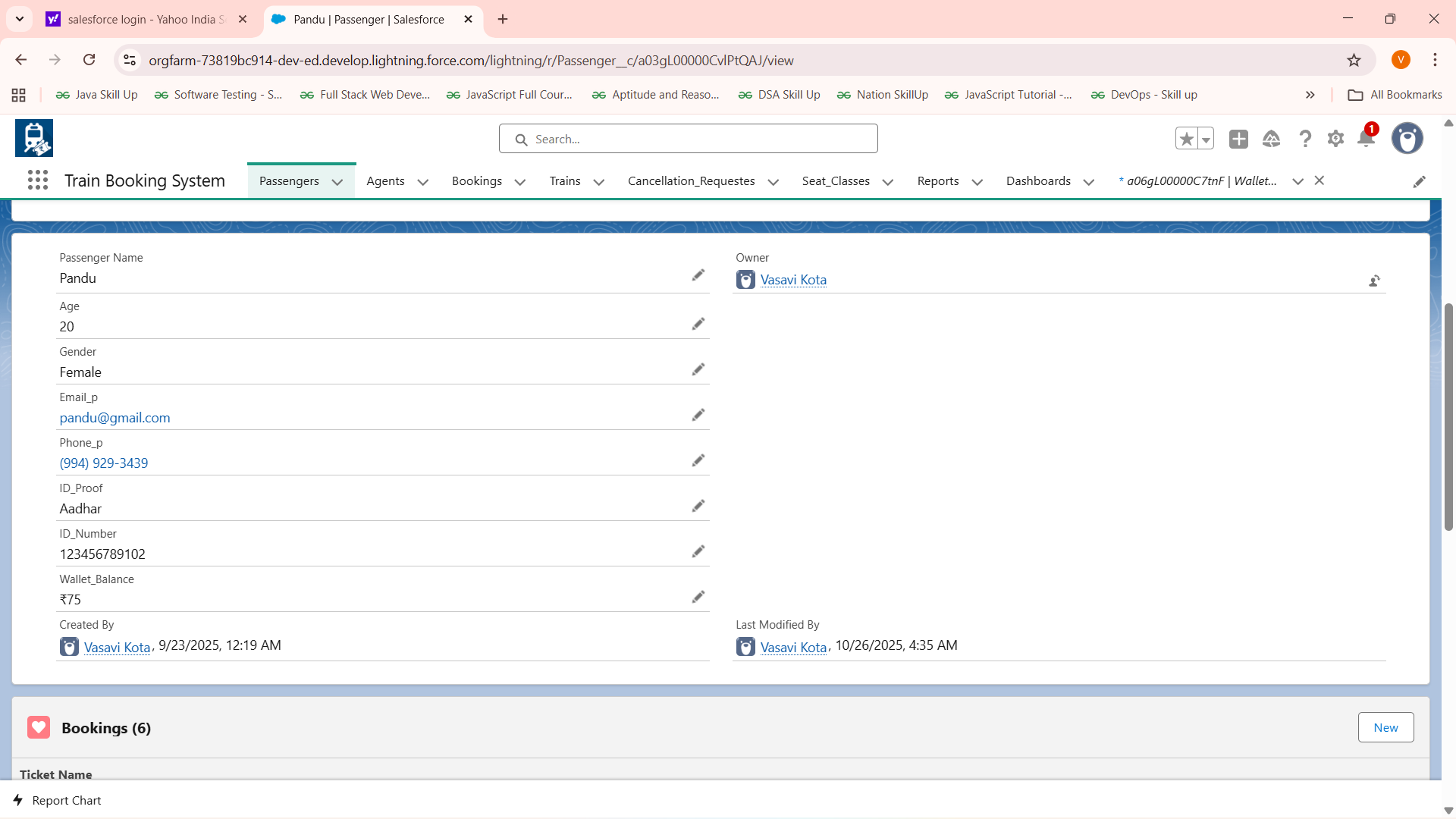
**After clicking Cancel Booking :**

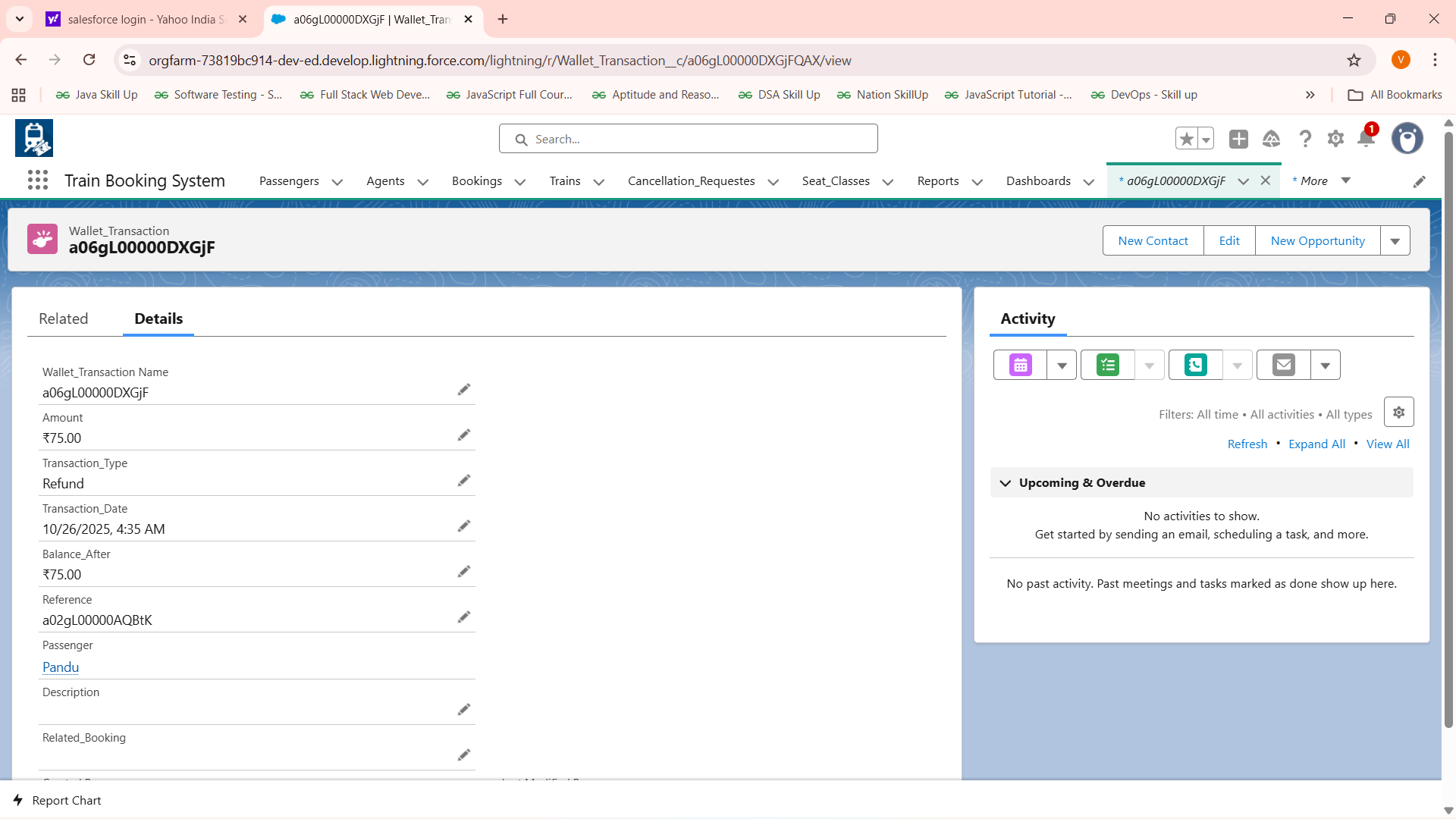


**Wallet Balance before Cancellation :**

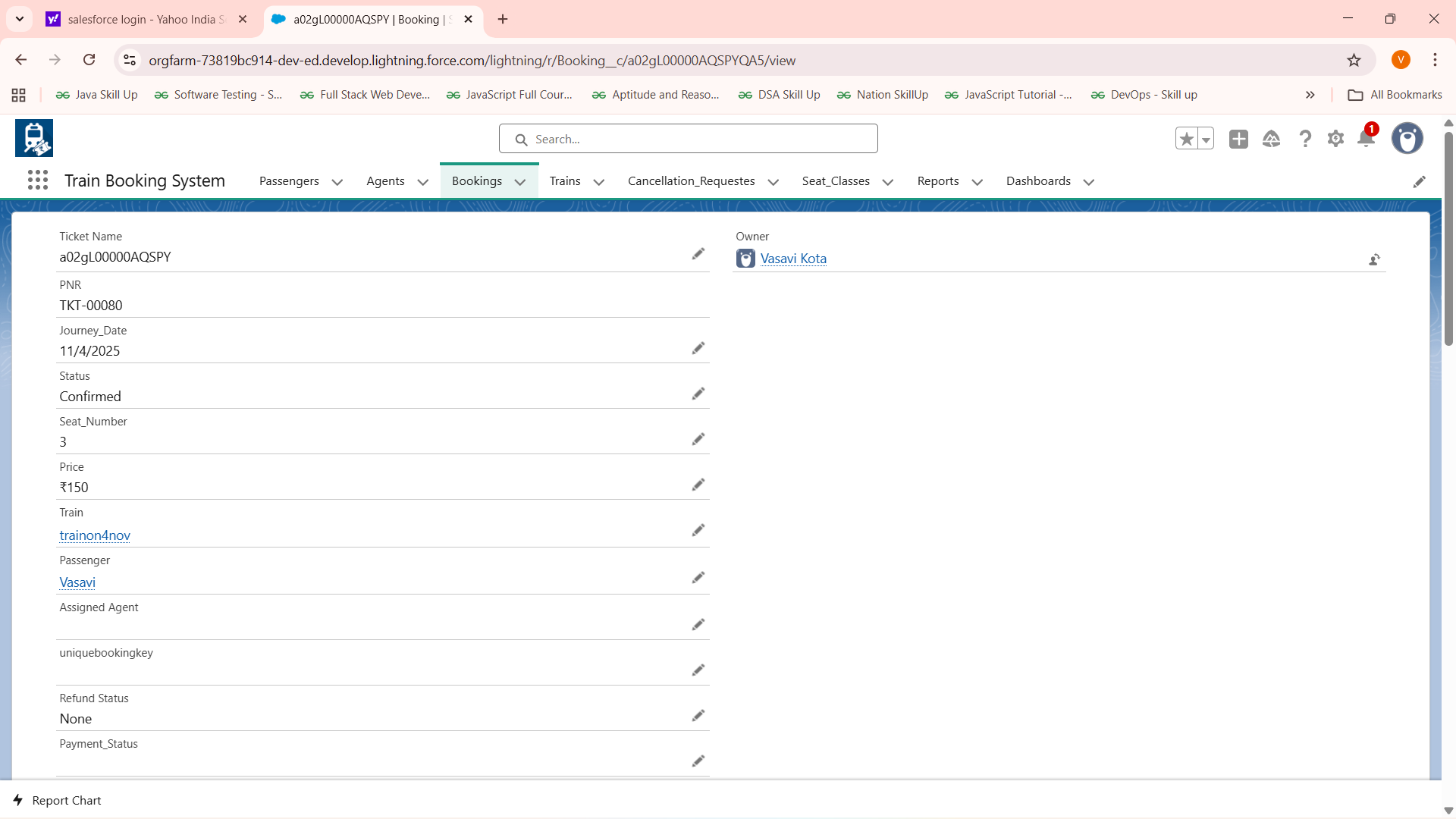


**Wallet Balance After Cancellation :**





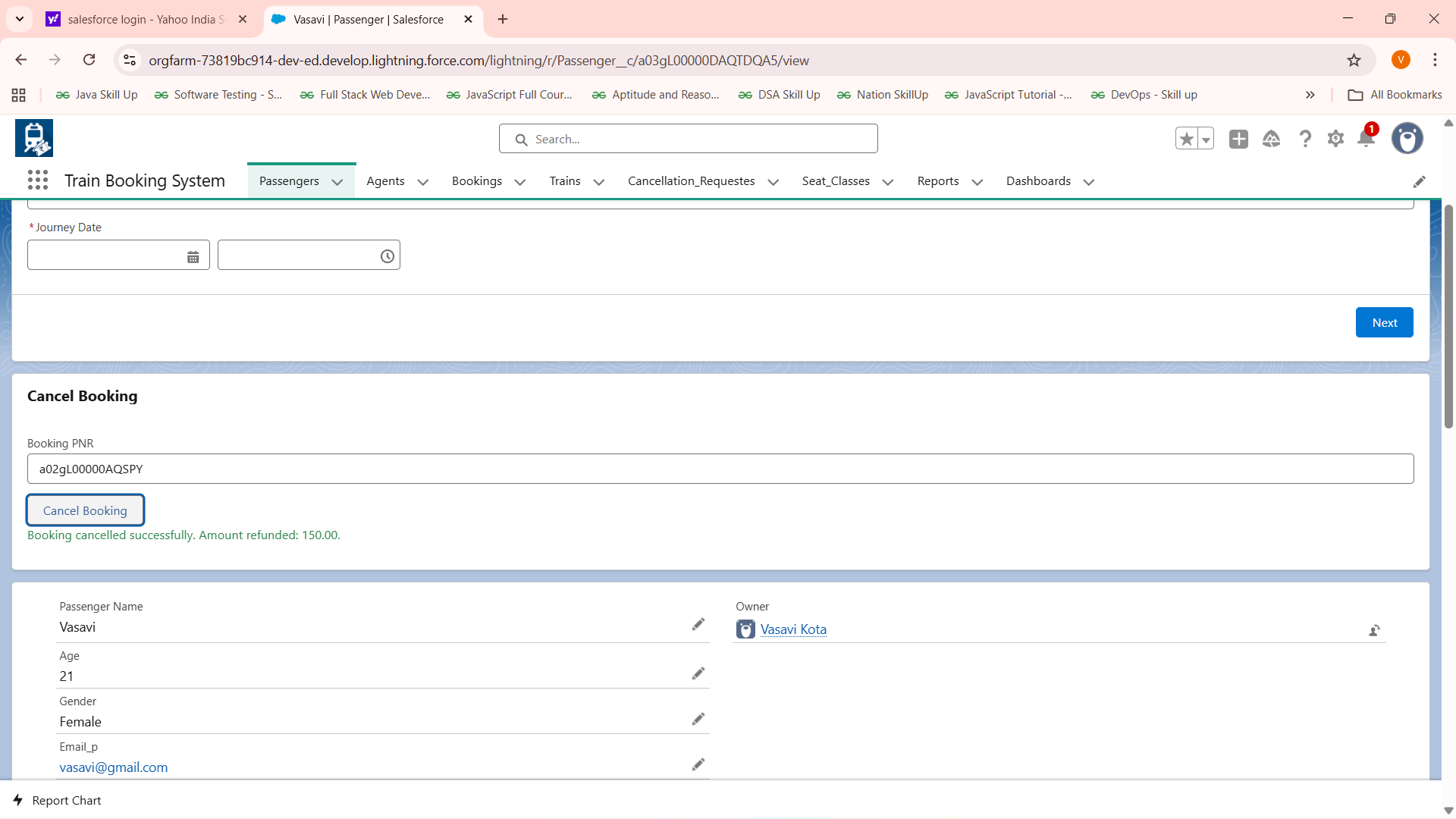
**SubtestCase-3:Cancellation Date >=7 Days**

**Input:PNR details** 

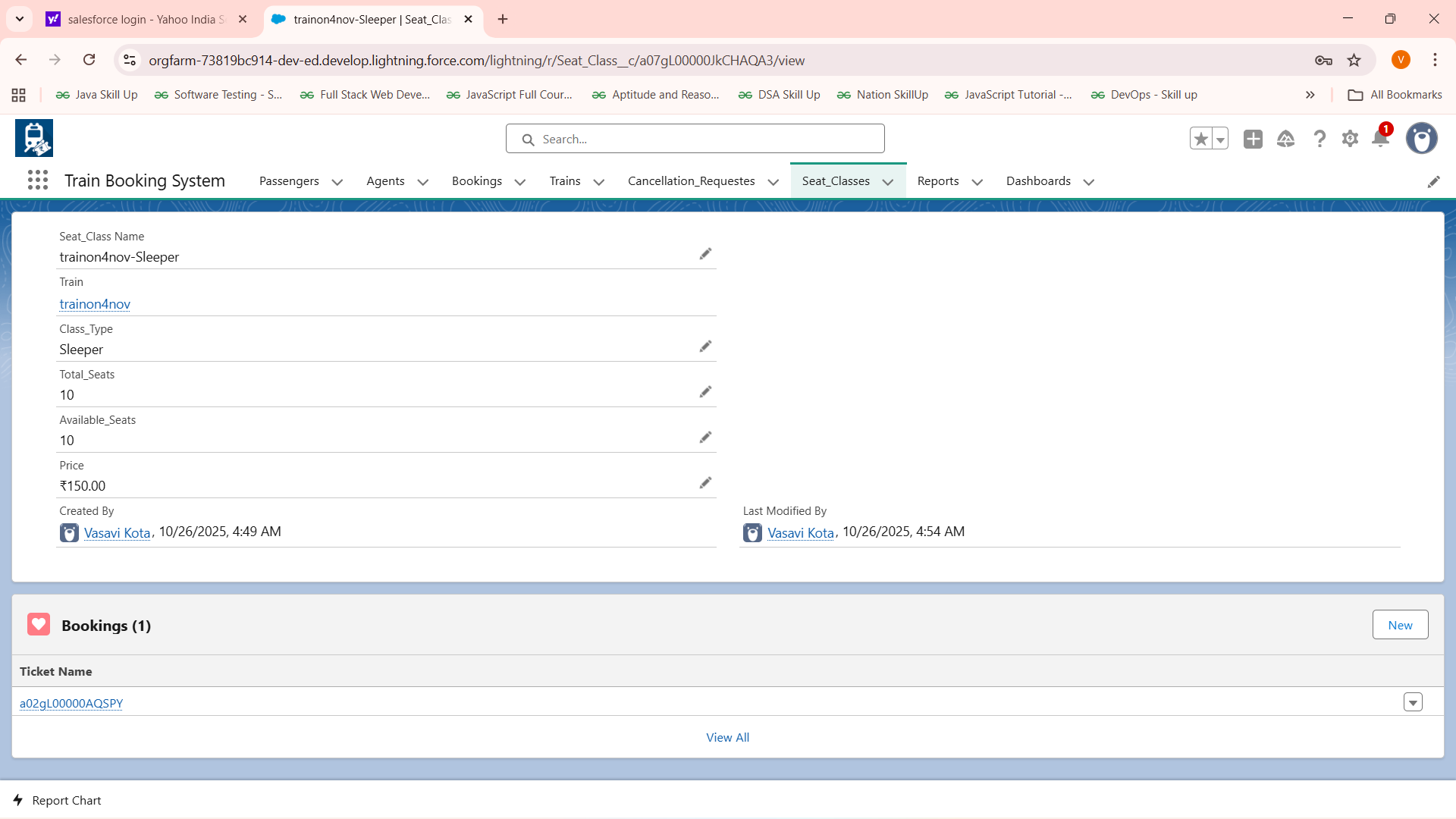
**Expected Output:refund full amount and update available seats**

**Actual Output:**

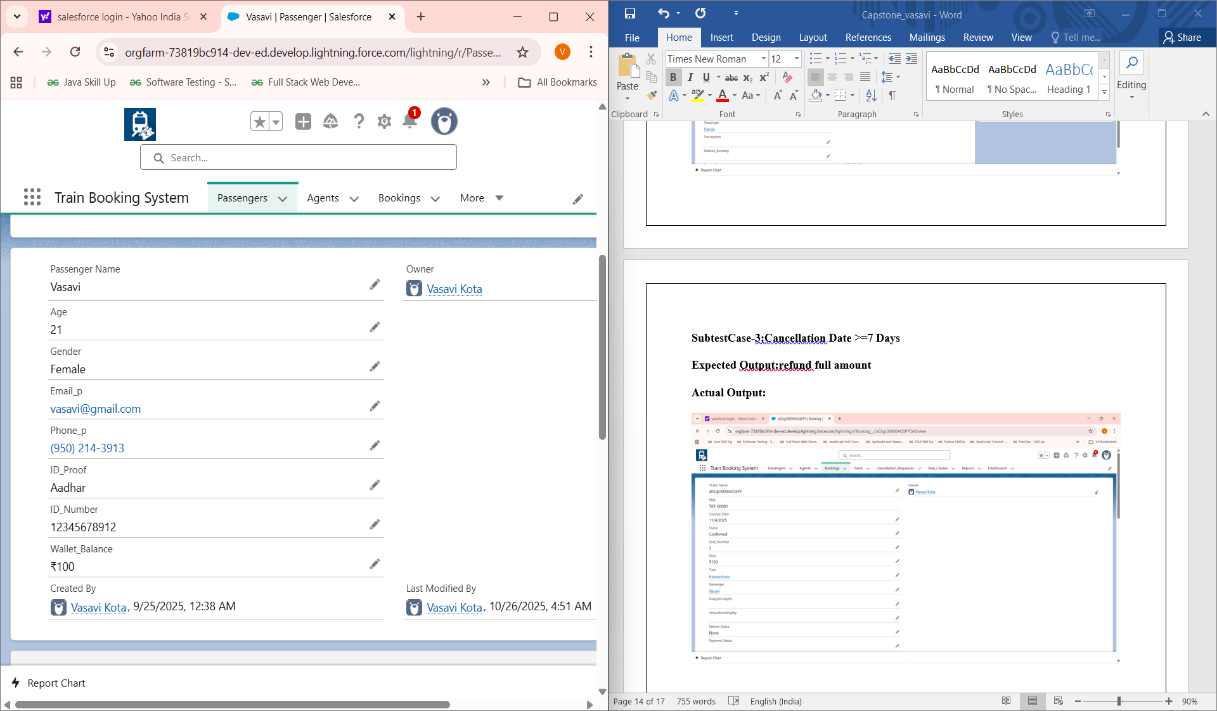
**FIG:Booking cancelled:**

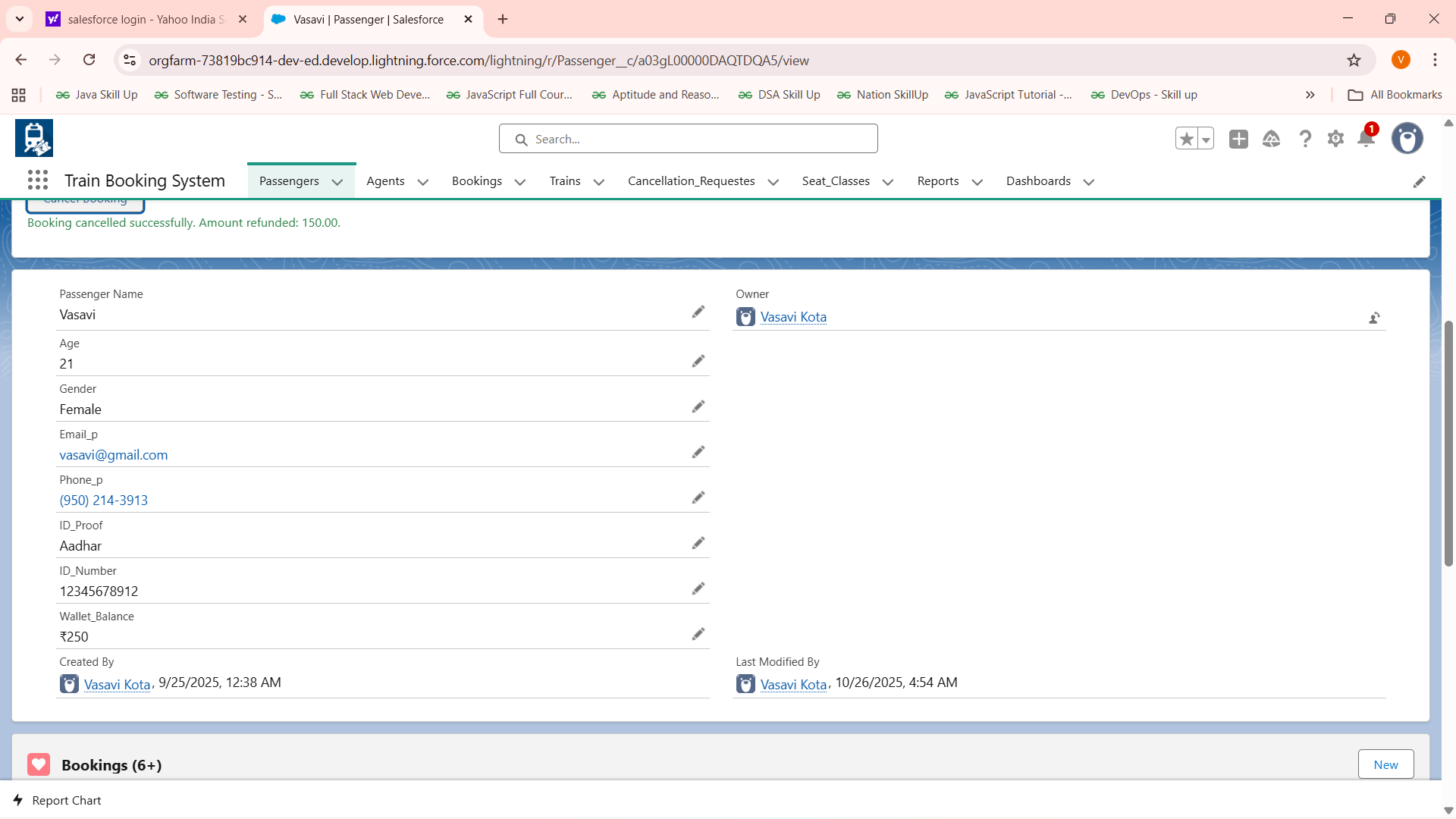


**FIG: Seat –Class Update :**



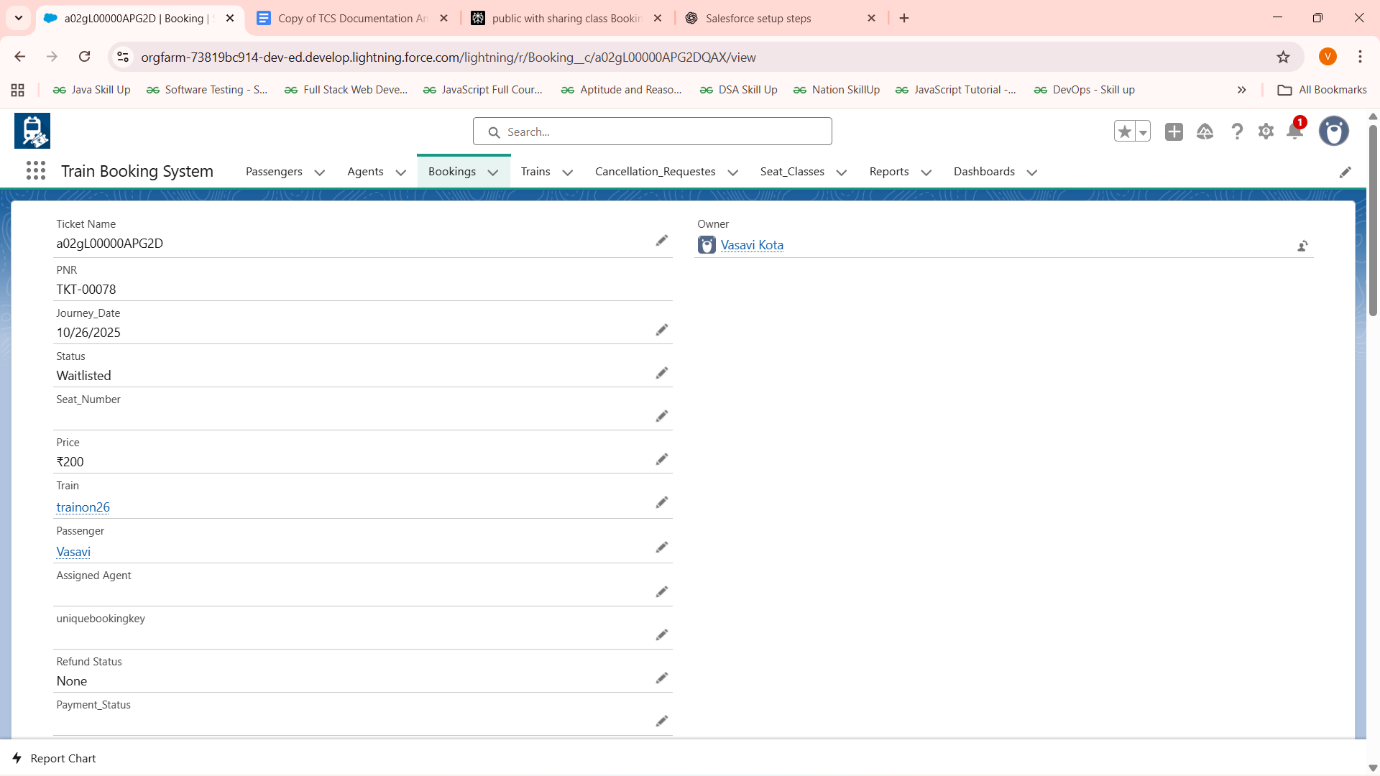
**Balance Before Cancellation:**

**Balance After Ca****ncellation:**



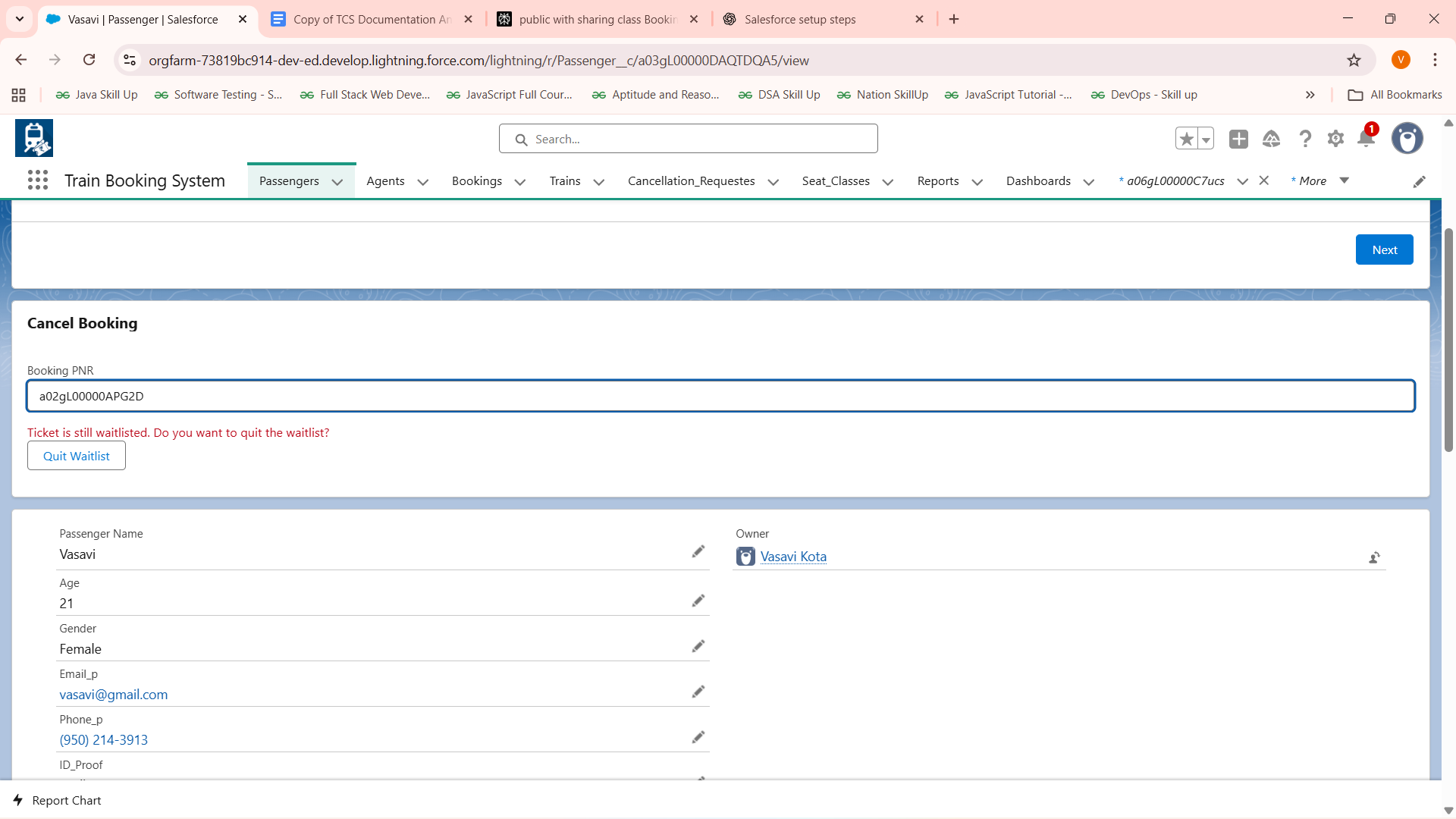
**TestCase-2:**Cancellation for Waitlist ticket(for waitlist we have not deducted money until confirmed so no need to update the wallet)

Input : Entering waitlist PNR

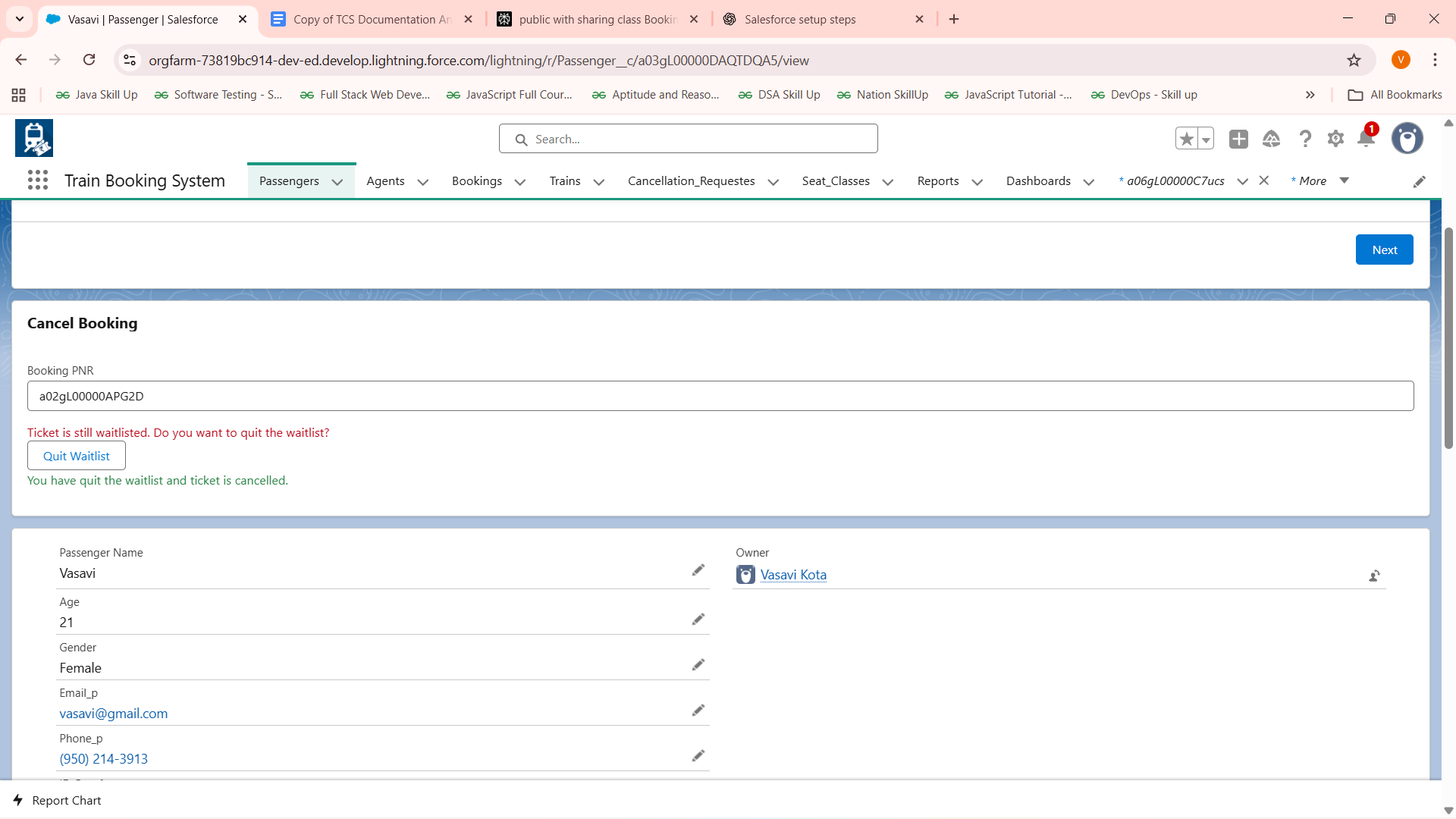


Expected Output: Ask for quitting waitlist and update Waitlist count if cancelled waitlist and update status to cancelled

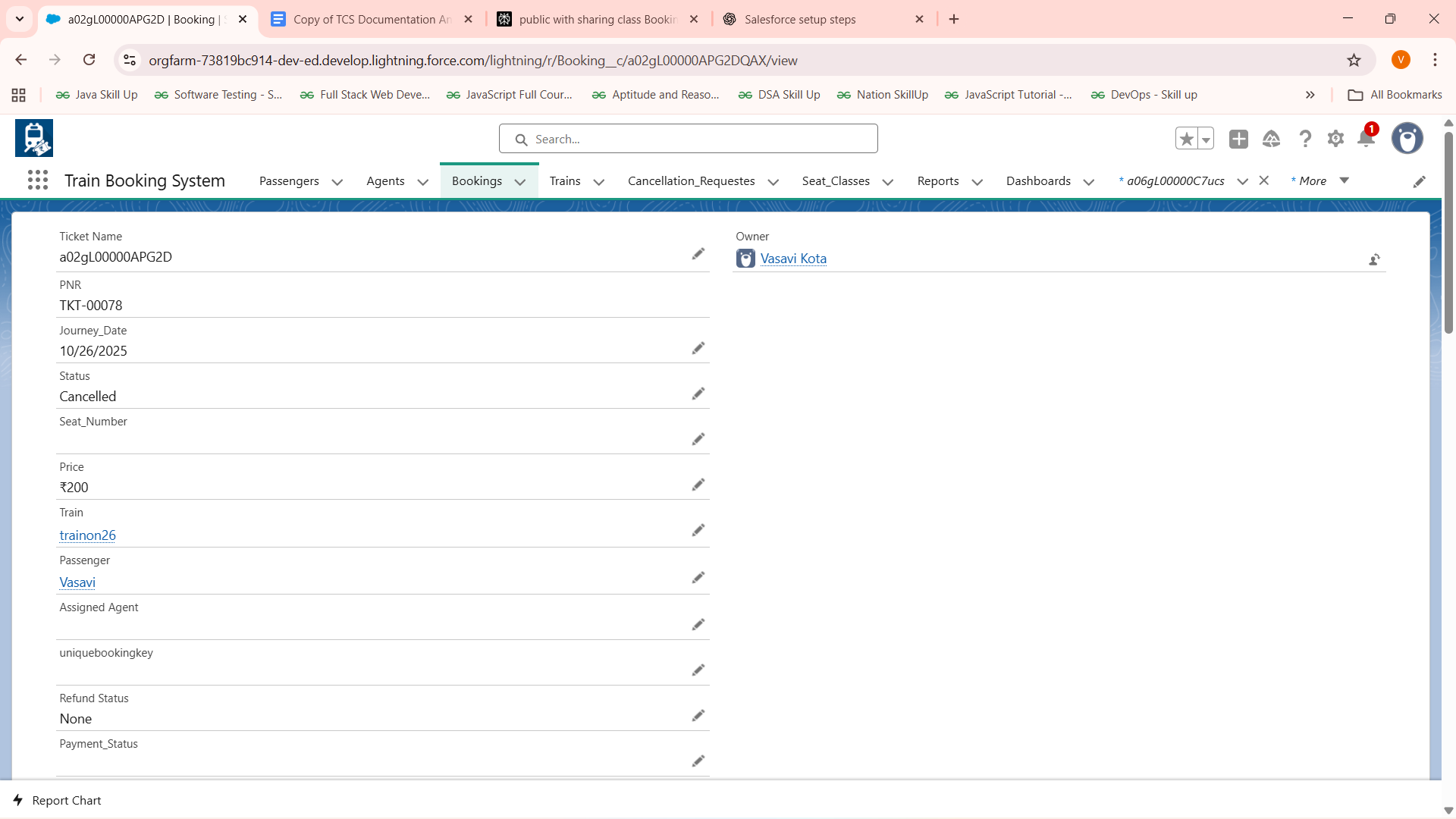
Actual Output: Ask for quitting waitlist and update Waitlist count if cancelled waitlist and updated status to cancelled



After Clicking Quit WaitList button :

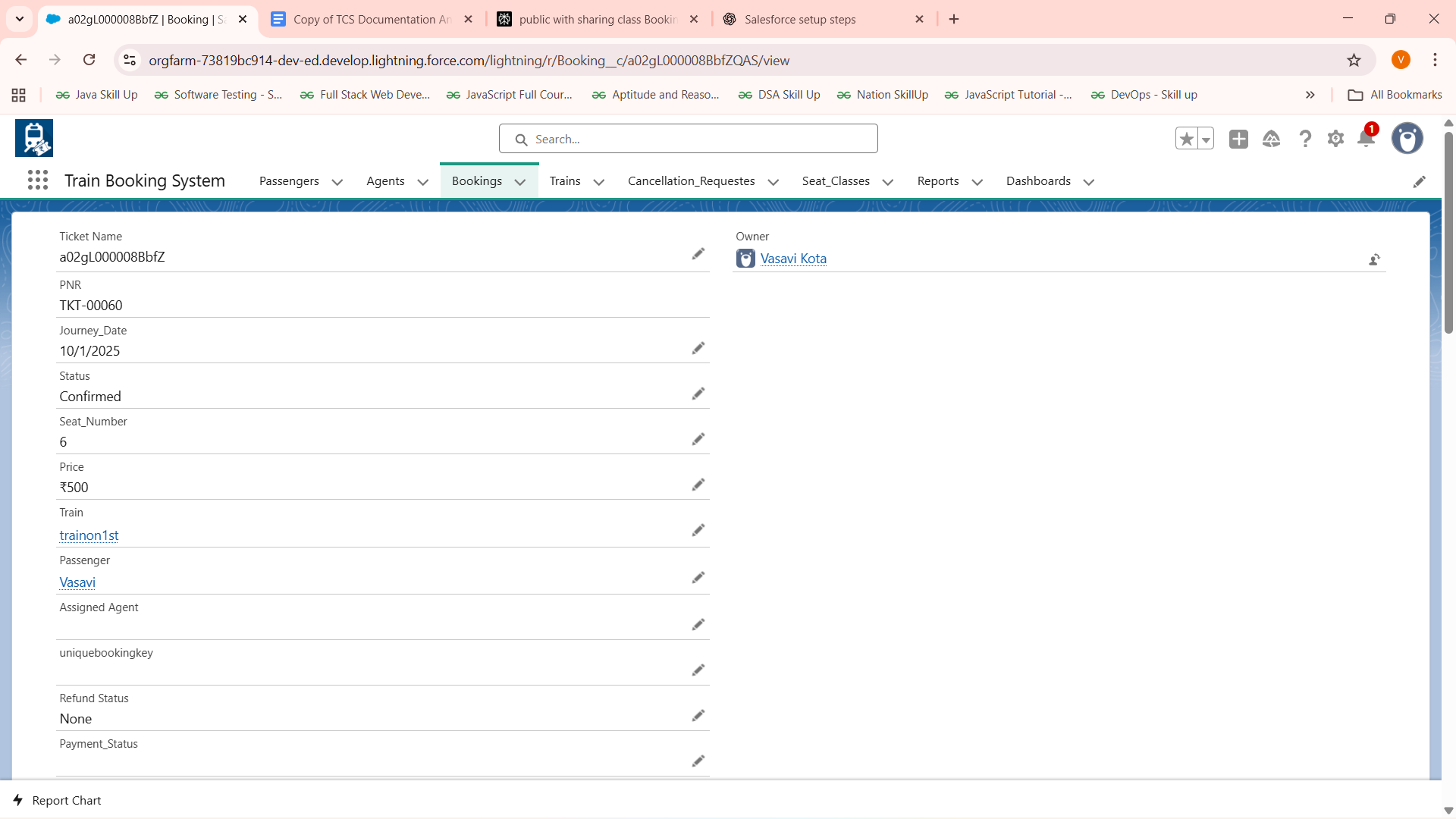


Status Changed to Cancelled :



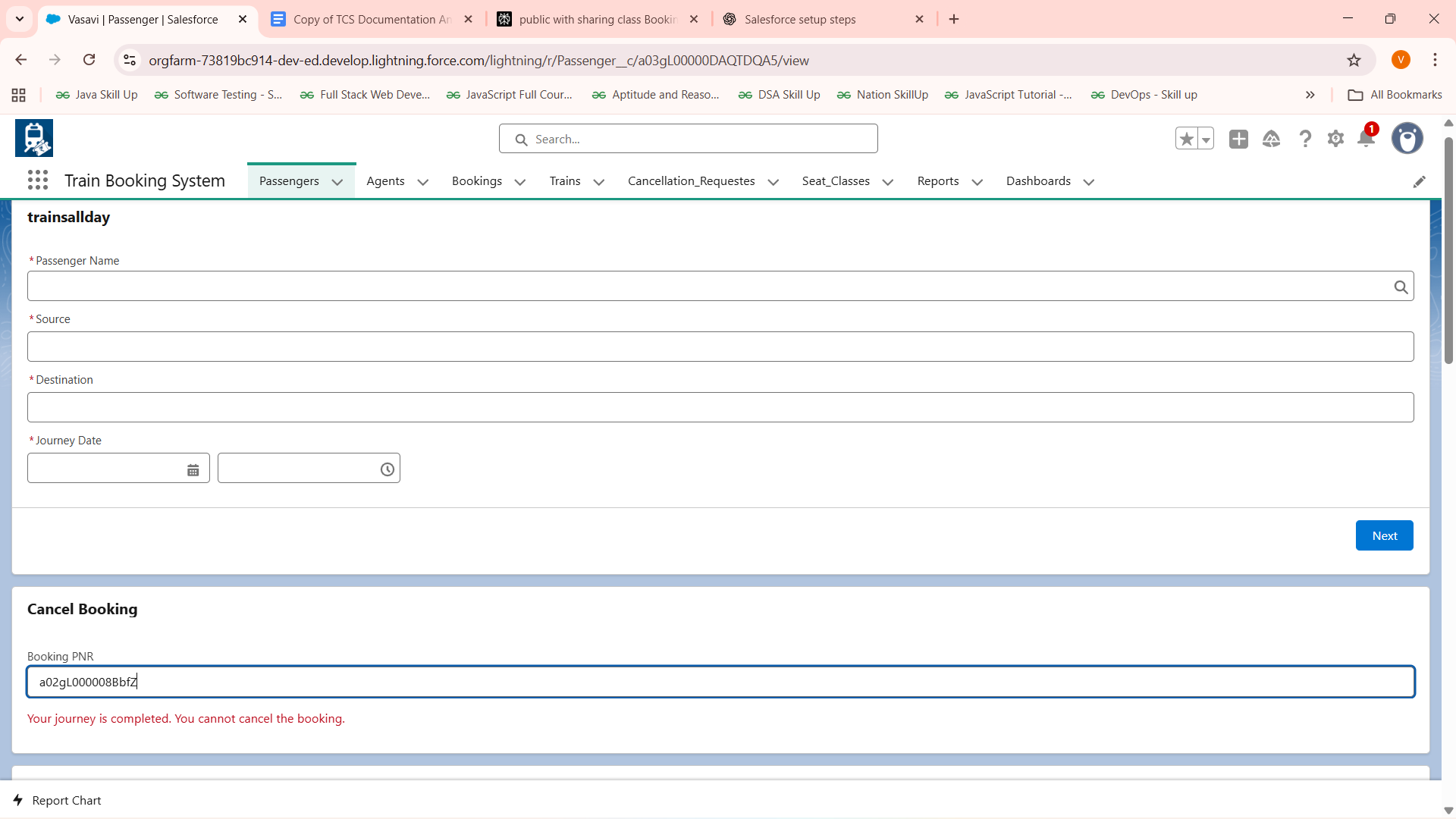
**TestCase-3:**

Input:Entering already journey completed PNR



Expected Output:Display error message

Actual Output:



**TESTCASE For Waitlist:**

Promoting Waitlist to confirmed and assigning seats when someone cancelles the ticket

Fig:They cancelled the booking and their seat no is 4

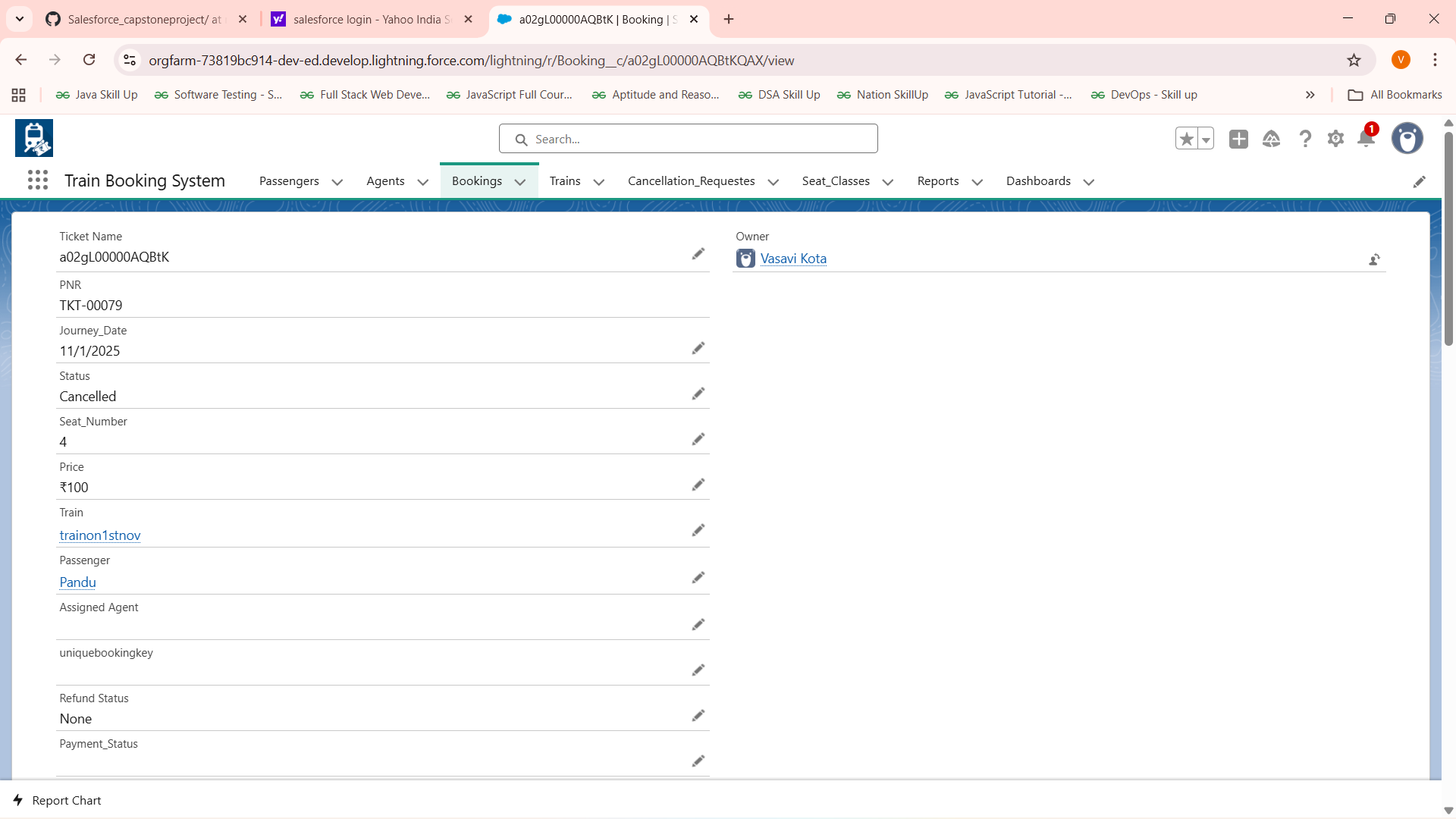
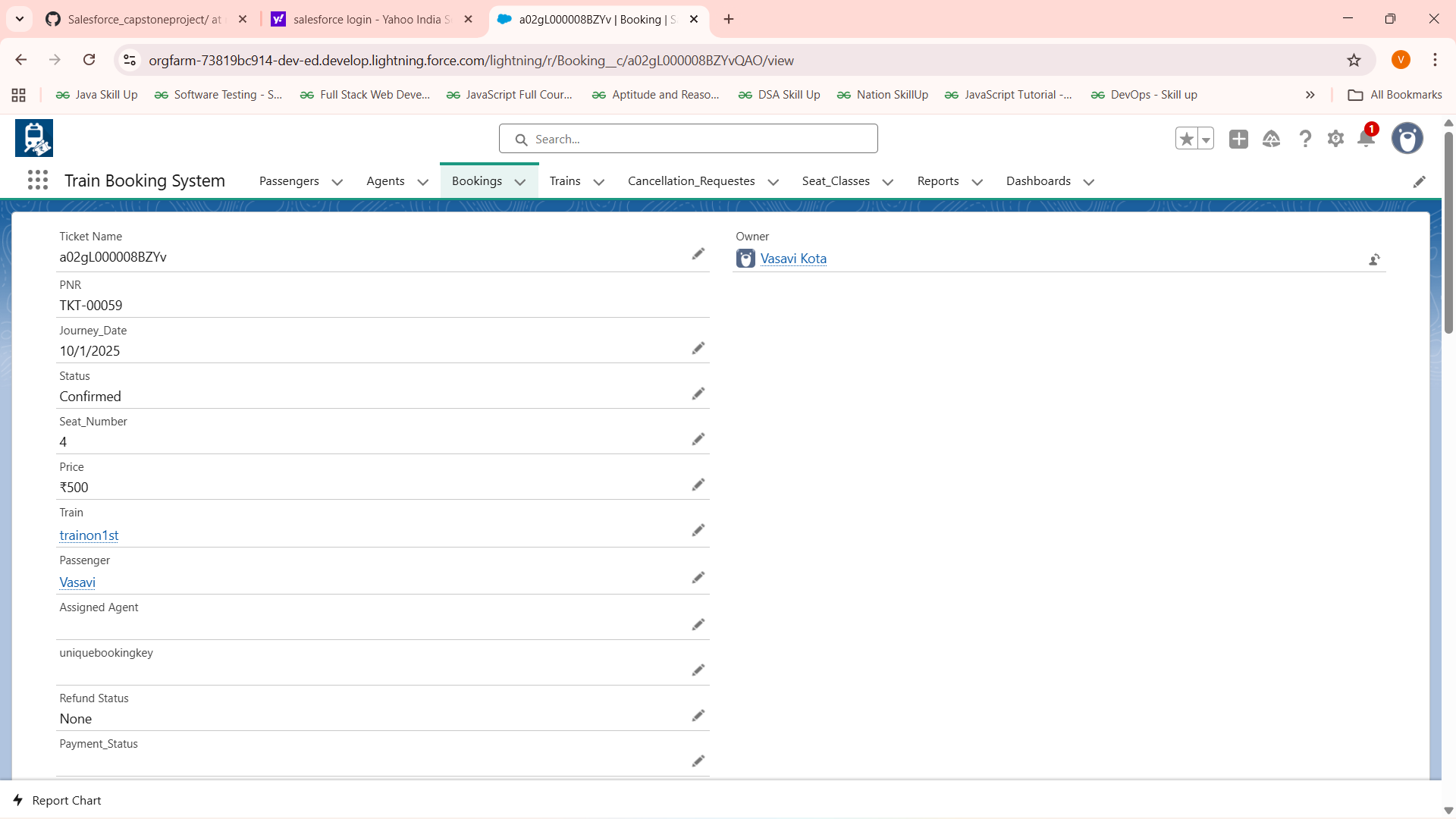


FIG:waitlisted booking changed to confirmed and assigned the seat number



**Future Enhancements :**

-Adding an Agent with whom passengers can ask their details regarding booking and can ask agent to send the cancellation requests ,know about different trains ,fairs,etc**(Started Implementing)**

-Adding further notification sending options

-Advanced payment methods