TRUCK BOOKING MANAGEMENT PROJECT

- **Aninda Banerjee and Team**

https://projectcom94-dev-ed.develop.my.site.com/s

Overview:

An application that manages truck booking. The customer details and the booking details are handled in the Salesforce org. It includes basic functionalities, business use cases, and everything else that an app needs.

Use Cases/Functionalities

1. App for Truck Booking should be accessible by all users.

2. Truck Managers should have access to all the records.

3. Financiers should have access to only the financial data.

4. Truck Drivers should view or edit only their bookings and view the related load information.

5. Customers should be able to create or modify their booking and related load information.

6. Customers should be presented with a UI to view the list of trucks along with Name, Image, Maximum Load, and Cost.

7. Filters should be available for pickup and drop points.

8. After selecting the truck a booking will be created and notification will be sent to the truck manager.

9. Truck Manager will be able to assign a driver to the truck.

10. The truck driver will be able to change the status of the booking from not started to in progress to completed to payment pending or payment received.

11. Customers will be able to add a load item for every booking with the approximate weight.

12.Payment is done offline, payment for every booking will be managed by financiers either as a full payment or advance payment or installments with the mode of payment.

Phase 1 : DATA MODELLING

1. ***Objects:***

a) **Custom** Objects:

i) Truck__c – what available to book.

Purpose:
Stores each truck's details.

Why needed?
Every booking needs a truck to be assigned. Keeping it separate helps reusability and prevents duplicates.

*Why custom?
No standard object for asset like Truck in core Sales Cloud.
Standard Asset Object exists, but it's used for products purchased by customers ----- here the Truck is owned by com, not sold – it's rented.

ii) Truck_Booking__c -- who booked which truck

Purpose:
Represents a booking made by a customer for specific truck.
Why needed?
It's the core object of the project --- connecting customer, truck, load and payment.
*Why custom?
Standard objects like Order or Opportunity don't fit ------

They assume product-sale context, they don't support route info, pickup/drop or truck assignments.
Our booking flow needs- pickup/drop formulas, availability validation, driver assignment. So custom needed.

iii**) Truck_Load__c** – what items are being carried

Purpose:
Stores load details for each booking.

Why needed?
A booking can have multiple load items, so we need a separate child object.

Why custom?
One booking – Many Loads (1-M rel)
Standard SF has no direct "child" object for this pattern.
CO allows – separate load weight validation, related list on booking, clear ds for reporting (total weight per booking etc).

****
Why we made another obj Truck Load instead of making text fields in Truck Booking?

- To particularly keep a booked trucks load info which we don't want to share with other users having access to Truck Booking, like they can see total load, but can't see the load items like that.

iv) **Payments__c -- how the customer pays**

Purpose:
Tracks payments(payment type, payment status , amount, instalments etc).

Why needed?
Financial tracking is done by financier, hence isolated from booking for clarity.

** Why custom when you already have a standard one?
It's tightly linked with **Invoices, Orders and Products from Billing cloud**, which we don't use.

It doesn't have an OwnerId field --- so you can't assign payments record to financier users, can't control visibility using owd, roles or profiles. We need **ownership-based access** -> not possible without ownerId.

2) Relationships:

i) Truck Booking (C)– Truck(P) -> LookUp : To know which truck is booked

1-many actually , but restricted to 1 active at a time.

Why LU not MD?

Logically , no need of TB if there is no T, but we must **balance business dependency vs data safety** ---

- Prevent data loss : accidental delete of truck record – if a Truck record gets deleted accidentally, in MD, you'll lose its Bookings, Loads and Payments – the entire booking/payment history.
- Truck can change : If a Truck breaks or replaced – Truck Manager can assign another truck in the booking lookup simply.
- Independent lifecycle
- Record Ownership: In LU, TB has its own OwnerId, independent of the Truck owner , so – Customer -> owns Booking
  Manager-> owns Truck
  Financier -> owns Payment

In MD , you'd lose this flexibility – all ownership would be controlled by the Parent Truck.

Ex: Think of OLA car booking – if your booked car got damaged, your booking record still exist, you may assign another car.

ii) Truck Load(c) - Truck Booking (P) -> LookUp - each load belongs to one booking

Why LU not MD?

we want to keep **data flexibility** and **safety** wo automatic deletion.

- Avoid accidental cascade delete – for audit, analytics
- Operational flexibility – like booking was cancelled and the load was rescheduled and reassigned to another booking.
- Independent ownership – Load record can be owned by same customer or driver for updates (weights, items).
- Report simplicity

Ex: A “Load” is like the shipment list for that trip.

Even if booking is cancelled, the com still wants to know what load was planned , its weight etc.

iii) Payment(C)– Truck Booking (P) -> LookUp – Each payment linked to one booking

Why LU not MD?

Same Reason as above. (Audit gaps, financial data loss)

Ex: Imagine you paid 5000 for a truck booking, later you cancel your booking. So, do you expect your payment record to disappear from the com db ?

No right? Finance needs to track that 5000 --- refund, adjust and audit later.

That’s why we need LU over MD.

3) Important Field Created :

Truck object:

Truck Id - auto num – unique identification

Truck Name – text – display name of truck

Maximum Load – Num – helps validation for weight check

Cost of Booking – Currency – used to calculate payable amount

Availability of Truck – checkbox – availability based on booking status

Truck Booking Object:

Truck Booking ID – auto num – unique id

Truck Name – LU(Truck) – to know which truck is booked

Customer - Lookup(Accounts) – to know who booked

Date Of Booking - Date & Time – booking time

Customer's Load - Number – helps compare with truck's max load

Payment Type - Picklist – upi/cod/installment

Status – picklist – booked/ in transit/ completed – tracks booking stage

Pickup State - Picklist

Pickup City - Picklist(Dependent on Pickup State)

Pickup Area - Text Area(255)

Pickup Address - Formula(text)

```
TEXT(Pickup_State__c) & ", " &
TEXT(Pickup_City__c) & ", " &
Pickup_Area__c
```

Drop State - Picklist

Drop City - Picklist(Dependent on Drop State)

Drop Area - Text Area(255)

Drop Address – Formula(text)

```
TEXT(Drop_State__c) & ", " &
TEXT(Drop_City__c) & ", " &
Drop_Area__c
```

Truck Load Object:

Booking ID - Lookup(Truck Booking) – connects load to booking

Truck Load Name - Text

Load Items with Respective Weights - Text Area – list of load items

Weight of the load(kgs) - Number – for load validation

Payment Object:

Payment Name - Text(80)

Booking ID - Lookup(Truck Booking)

Payment Type - Picklist – cod/upi/installment

Payable Amount - Currency

Payment Status - formula – `IF( Remaining_Amount_to_be_Paid__c =0,"paid","pending")`

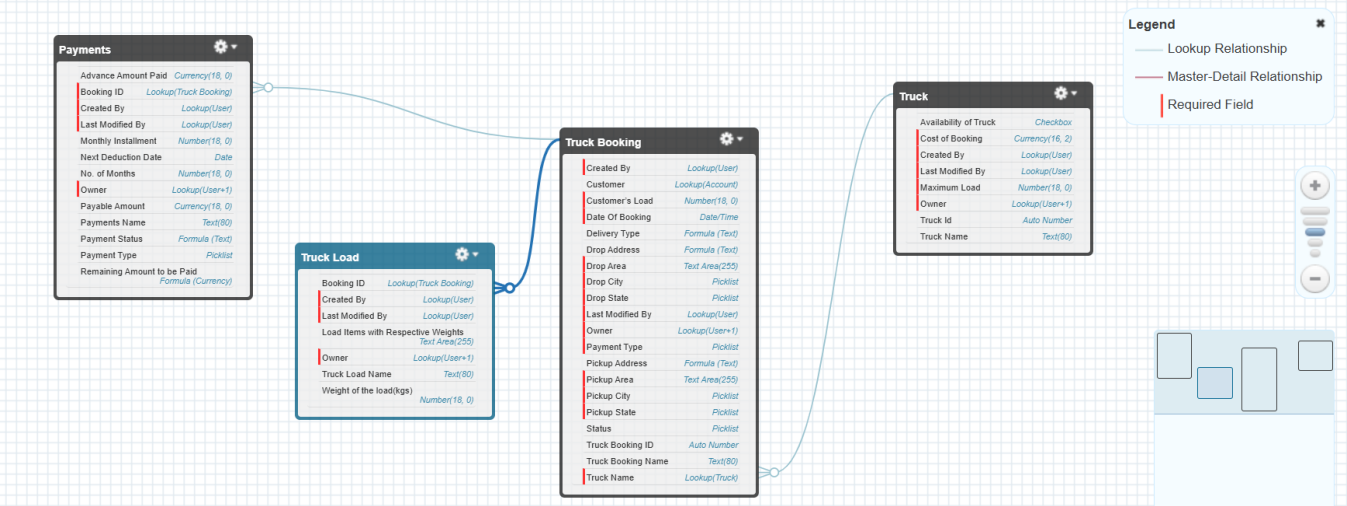
Advance Amount Paid - Currency

Monthly Installment – number

Next deduction date – date

No of months – num

Remaining amount to be paid – formula(currency) : Payable_Amount__c - Advance_Amount_Paid__c



4)Profile and Access Control (Data Security):

2 profiles:

System admin: TM

Standard User: Rest 3

4 USERS :

Internal:

Customer (ANKITA BANERJEE)

Truck Manager (VISHAL AGARWAAL)

Truck Driver (ANIL ROY)

Financier (ANUPAM JHA)

myManger (admin) -so that I can check Custom Notification on my real accnt i.e. Aninda(admin)

external:

customer1(Andy Baner) - License: Customer Community Plus Login

Object permissions (CRED):

Object CRUD	Customer 1	Truck Manager	Truck Driver	Financier
Truck	R	RCE	-	-
Truck Booking	RCE	RCE	RE	-
Truck Load	RE	RCE	RE	-
Payments	R	RCE	-	RCE

- Defines baseline access for each user type.

Field level:

profile select – fls- obj name-viewedit-

Truck Manager :

For all Objects : Read access & Edit access – for all the Fields

Financier :

Payments : Read access & Edit access – for all the Fields

No need to set FLS for rest objects as – **OLS is off** .

Truck Driver:

Truck Booking: Read access to all , Edit Access – By Default + Customer & Status

Truck Load: Read access to all , Edit Access – By Default + Booking Id & WOTL

Customer(external):

Truck: Read access to all , Edit Access – By Default

Truck Booking: Read access to all , Edit Access – By Default + Customer

Truck Load: Read access to all , Edit Access – By Default + LIWRW + WOTL

Payments: Read access to all , Edit Access – By Default

Record level access :

OWD-

Set up – sharing setting – owd for each user:

Truck : public read only (all can see available trucks)

Rest : private (so users see only their own record)

Role Hierarchy-

CEO - Sys Admin : my org user

Financier

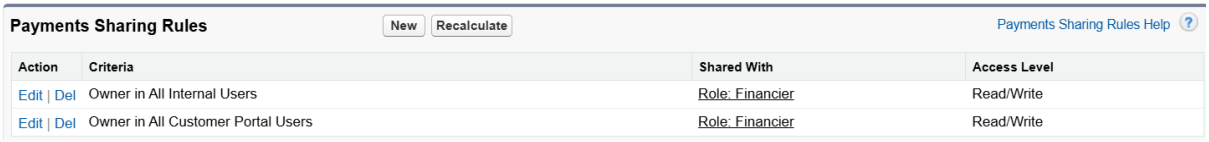
TM -

Customer

TD

Sharing Rule-

Role: Financier – share all record of payments



Payments Sharing Rules New Recalculate Payments Sharing Rules Help

Action	Criteria	Shared With	Access Level
Edit | Del	Owner in All Internal Users	Role: Financier	Read/Write
Edit | Del	Owner in All Customer Portal Users	Role: Financier	Read/Write

Manual sharing-

TM -> Truck Driver

5) Validation Rules:

Workflow Rules are designed to automate actions (like sending emails, updating fields), not to prevent data entry.

Validation Rules are specifically built to block saving records when certain conditions are not met.

So for our project need, we have to use **validation rule.**

Truck:

Positive_Load :

Error Condition Formula	Maximum_Load__c <= 0
Error Message	Load must be positive.

Truck Booking:

Rule Name	Load_Validation
Error Condition Formula	Customer_s_Load__c > Truck_Name__r.Maximum_Load__c
Error Message	Weight limit exceeded.

Rule Name	Booking_Date_Validation
Error Condition Formula	DATEVALUE(Date_Of_Booking__c) < TODAY()
Error Message	Booking date cannot be in the past.

Payments:

Rule Name	Advanced_Amount
Error Condition Formula	Advance_Amount_Paid__c > Payable_Amount__c
Error Message	Advance cannot be more that payable ammount.

Rule Name	Can Workflow Rule Replace It?	Why Not?
Positive_Load	❌	Workflow can't block record save
Load_Validation	❌	Workflow can't compare related fields or stop save
Booking_Date_Validation	❌	Workflow can't prevent past dates
Advanced_Amount	❌	Workflow can't validate numeric limits

6) Look-up Filter:

A Lookup Filter in Salesforce is used to restrict the list of records shown in a lookup field based on specific criteria. It helps users select only relevant and valid records, improving data accuracy and user experience.

Truck Booking Object -> Truck Name

Availibility of Trucks = true -> only available trucks will be shown.

Filter Type: Optional: Suggests available trucks, but users can still select unavailable ones. So that if a customer wants to update other fields like address ,then attempt to save, it will be allowed to.

7) Duplicate Rules:

Duplicate Rules in Salesforce are used to **prevent or manage duplicate records** across objects like Leads, Contacts, Accounts, or even custom objects.

Object: Truck

Rule Name: Truck Name

Action on create : Blocked

Action on edit : Allow – Alert

Alert Text: Record Already Exists.

Matching rule: Truck Name

Matching Criteria `Truck: Name` EXACT MatchBlank = FALSE

Matching Rule: Defines what makes records "duplicates".

Duplicate Rule: Controls what happens when a duplicate is found.

8) FEED TRACKING :

On Truck Booking : state -city -area + load + dob + status + truck booking name

9) DYNAMIC FORM:

On **Payments**-

If Payment Type = Installment : Monthly Installment

No. of Months

Next Deduction Date - will be shown.

10) FLOWS (For automation and exp cloud's home):

i> Avialability Of Truck

Your “Availability of Truck” flow is Record-Triggered because it automatically reacts to — it runs whenever a Truck Booking record is created or updated (no manual input).

Truck Booking = child

→ has Status (In Progress, Completed, etc.)

Truck = parent

→ has Availability__c checkbox

can only look up to fields on its own parent (not down to child records).

In other words:

Parent → can’t access Child fields directly.

Child → can access Parent fields easily (via lookup).

So, Truck can’t "see" the status of its bookings.

👉 Formula field can’t handle this scenario.

It won’t know when any booking changes status.

That’s why you use a Record-Triggered Flow on Truck Booking:

Triggered when a Truck Booking is created or updated

Checks the Status

Updates the related Truck’s Availability__c accordingly

ii> Set Payable Ammnt in Payment = Cost of Booking from Truck

🔹 Why it’s Record-Triggered (**Before Save**)

Because the Payment record should automatically get the Payable Amount as soon as a related Truck Booking is created.

⚡ Could You Do the Same Without a Flow?

Yes — here are your alternatives 👇

Formula Field (Simplest) If you only want to display the cost and don't need to store it: Truck__r.Cost_of_Booking__c Put this as a formula field on Payment → It'll always show the latest cost from the Truck. But drawbacks: It's read-only (you can't edit or override). It changes dynamically — if the Truck's cost changes later, the payment's amount changes too (may not be what you want). It's not "saved" value, just computed view.

iii> AutoCreatePayment

🚀 Why it's Record-Triggered

Because it needs to run automatically right after a Truck Booking record is inserted—no button click, no user action. A record-triggered flow fires when a record event happens (create, update, delete).

Here it's "on create", so the moment a Booking is saved,

Salesforce instantly: 1️⃣ Detects the new record 2️⃣ Executes the flow 3️⃣ Creates the Payment record in the same transaction

That's why it's record-triggered — it reacts directly to the creation event. It's record-triggered so that the moment a Truck Booking is saved, a Payment record is created automatically — no manual step needed. You could use Apex, but Flow is the best no-code, modern solution.

iv> Truck Load Automation

It's record-triggered because the flow must automatically run whenever a new Truck Booking record is created, creating its related Truck Load record without any manual input.

A formula field can't create records — only Flow, Process Builder, or Apex Trigger can.

v> setMonthlyInstallment

⚙️ Why it is **Before-Save** Because the flow is updating a field on the same record (Booking) So technically, yes, you could do this with a formula field, but in your case — where the value must be stored, stable, and editable — 👉 a Before-Save Record-Triggered Flow is the best and most efficient choice.

vi> CustomNotification

whenever a new booking will be created, a notification will be sent to manager via custom notification type object and custom action.

vii> Email Alert

whenever a new booking will be created, a email will be sent to manager via email template and Email Alert.

viii> Book a Truck

Screen flow for home page of the Exp cloud. Takes user input, shows available trucks, upon taking details from customer shows a success screen that the truck is booked.

Used Till Now:

- Custom Objects
- Custom Fields
- Formula Fields
- Relationships
- Data Security
    - Profiles
    - CRED
    - FLS
    - RLS
        - OWD

        - RH
        - SR
        - MS
- Validation Rules
- Look-Up Filter
- Duplicate Rules
- Dynamic Forms
- Feed Tracking
- FLOWS
- Experience Cloud for Website

Why haven't you used any Trigger till now?

- Dorkar poreni ekhono obdhi 😔 😔 😔 😔 😔 😔 😔 😔 😔 😔 .

Jokes apart –

I designed the Truck Booking Management System using Declarative Salesforce tools like Flows, Process Builder, and Custom Notifications because all the business requirements—like creating bookings, sending notifications to managers, calculating installments, and updating related records—can be handled without code.

---------- **INTERIM** ----------

Suggestion for Interim :

Prepare all admin topics like : Page Layouts – Interviewer ask scenarios based on page layouts, Record Types , Data Security , Relationships, Formula Fields, Validation rules, Duplicate Rules, Roles,profiles, users , FLOWS, Basic Apex . All admin Topics . Sales service and exp cloud basic knowledge , Apex scenarios basics. * Project full knowledge.

ADD LWC COMPONENTS INSTEAD OF THE SCREEN FLOW FOR UI

LWC COMPONENTS :

1. **Home1 – parent component**
2. **Booking1 – child component**
3. **myOrder – for showing my orders of the customer**

CODE :- C:\Users\2445097\Downloads\Truck.pdf

Experience Cloud :- https://cognizantonline-my.sharepoint.com/personal/2445097_cognizant_com/Documents/Desktop/SF/Experience%20Cloud%20App.pdf

---------- **FINAL** ----------

Suggestion for final :

Revise all admin topics, Deep dive into your project , you must know all things u used in the project. Apex Trigger scenarios , SOQL , LWC all topics in your project.

LWC is the key interview topic in finals , prepare well .