Marketplace Technical Foundation - Day-2

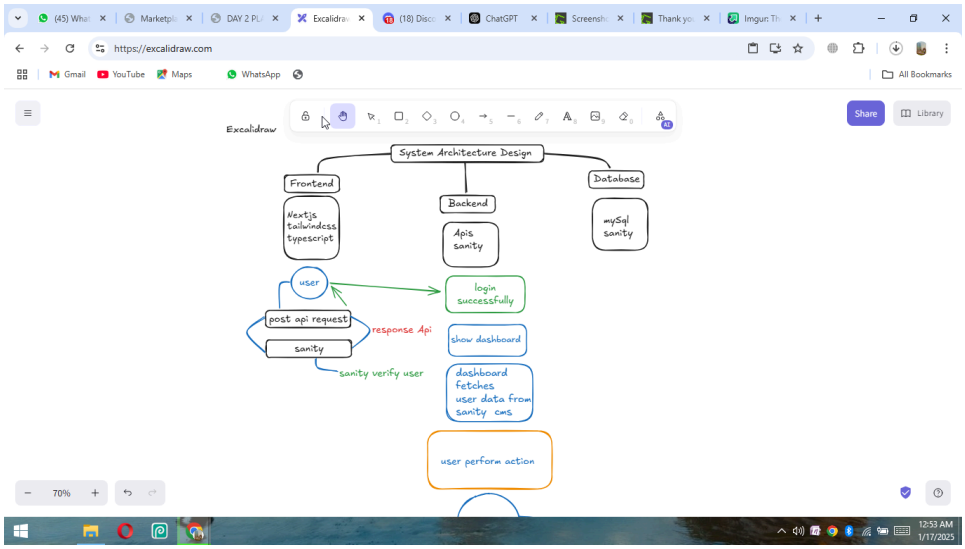
Day introduction:

Day 2 focuses on defining the technical foundation for our Rental E-commerce website. The includes designing the system architecture, defining workflow setting up sanity CMS schemas and planning Api endpoints

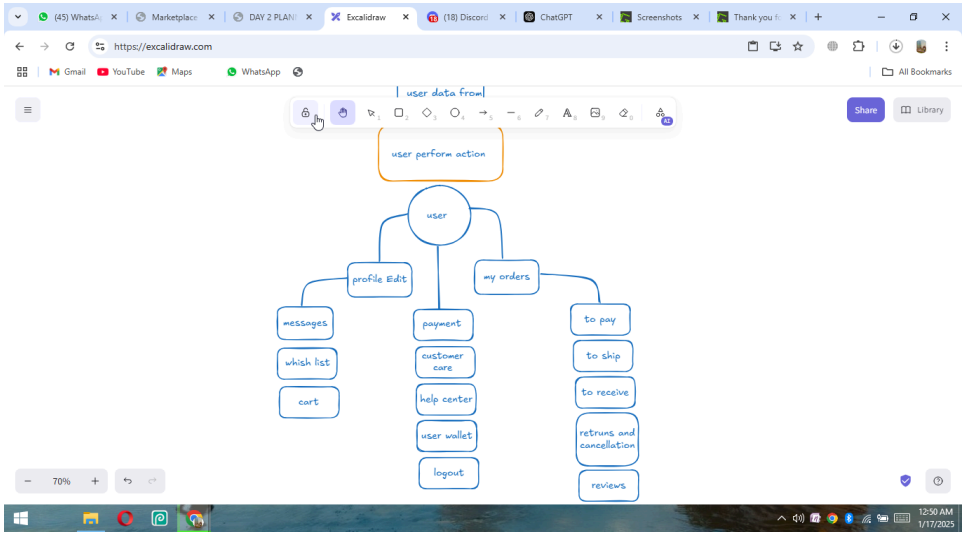
System Architecture:

Diagram :

Picture 1



Picture 2



System Architecture:

Description

- **Frontend :**

1. **Homepage: Displays car categories, featured deals, and search filters (location, date, car type).**
2. **Search and Booking: Dynamic forms for selecting rental dates, pickup/drop-off locations, and available cars.**
3. **Car Details Page: Displays car specs, images, rental rates, and availability.**
4. **Booking Summary: Shows selected car details, pricing breakdown, and user information form.**
5. **User Dashboard: For managing bookings, viewing rental history, and editing user profile.**
6. **Responsive Design: Mobile-friendly layout for seamless use across devices.**
7. **Payment Integration: Secure forms for payment options (credit cards, wallets, etc.).**
8. **Support Section: Includes FAQs, live chat, and contact forms.**

- **Sanity Cms :**

1. **Car Listings Management: Add, update, or remove car details (images, specs, pricing, and availability) through a user-friendly interface.**
2. **Location Data: Manage pickup and drop-off locations dynamically.**
3. **Deals and Offers: Easily create and schedule promotional banners and discounts.**
4. **Blog or News Section: Publish articles about travel tips, car maintenance, or seasonal promotions.**
5. **Dynamic Content Updates: Real-time updates for pricing, availability, and location changes without redeploying the website.**
6. **Multi-Language Support: Manage content in multiple languages for a broader audience.**
7. **Integration: Sync data with the frontend (e.g., Next.js) using Sanity's GROQ or GraphQL API.**

- **Apis : APIs include:**

- **Car Inventory API:** Provides real-time data on available cars, specifications, and pricing.

- **Search and Filter API:** Enables searching by car type, location, date, and other filters.

- **Booking API:** Handles car reservation, booking details, and updates.

- **Payment API:** Integrates secure payment gateways like Stripe or PayPal for transactions.

- **User Management API:** Manages user profiles, authentication, and rental history.

- **Location API:** Integrates services like Google Maps for pickup/drop-off locations and distance calculations.

- **Notifications API:** Sends booking confirmations, reminders, and
- alerts via email or SMS.

- **Review API:** Allows users to submit and view reviews for cars and services.

- **Admin API:** Enables administrators to manage inventory, pricing, and customer data.

Key workflow to include:

- ## User registration:

1. **Registration Form:** User enters details like name, email, password, and contact information.

2. **Email Verification:** An email is sent with a verification link to confirm the user's email address.

3 **Password Validation:** Ensures the password meets security criteria (length, complexity).

4 **Account Creation:** After verification, the user's account is created and stored in the database.

5. **User Authentication:** User logs in using the newly created credentials (email and password).

6. **Profile Setup:** Option to complete or edit profile (add payment details, preferences, etc.).

7. **Redirect:** User is redirected to their dashboard or main page.

- ## Product Browsing:

1. **Homepage Display:** Show featured cars, categories, and search bar.
2. **Search Filters:** User applies filters like location, dates, car type, or price range.
3. **API Call:** Backend fetches filtered car inventory from the database.
4. **Search Results:** Display matching cars with images, specs, and pricing.
5. **Car Details:** User clicks on a car to view detailed information (features, availability, reviews).
6. **Add to Booking:** Option to select the car and proceed to booking.
7. **Pagination or Infinite Scroll:** Smooth navigation for large inventories .

- ## Order Placement:

1. **Add to Cart:** User selects a car and adds it to the cart.
2. **Checkout:** User reviews the booking, enters details (dates, location, payment).
3. **Save Order:** Order details (car, user info, dates, total price) are saved in Sanity CMS.
4. **Payment:** User completes payment via integrated gateway.
5. **Confirmation:** Booking confirmation is sent to the user (email/SMS).

- ## Shipment Tracking:

1. **Fetch Status:** Order status updates are retrieved from a 3rd-party API.
2. **Update Display:** Real-time status (e.g., "In Transit," "Delivered") is shown to the user on the dashboard.
3. **Notifications:** Alerts sent to the user via email/SMS for status changes.

Apis Endpoint:

1. **Authentication:**
- **`POST /api/login`: User login.**
- **`POST /api/register`: User registration.**
- **`POST /api/logout`: End user session.**

2. **Cars:**
- **`GET /api/cars`: Fetch all available cars.**
- **`GET /api/cars/:id`: Get details of a specific car.**
- **`POST /api/cars`: Add a new car (Admin).**

3. **Bookings:**
- **`POST /api/bookings`: Create a booking.**

- **`GET /api/bookings/:id`: Fetch booking details.**
- **`PUT /api/bookings/:id`: Update booking status.**

4. **Payments:**

- **`POST /api/payments`: Process payment for a booking.**

5. **Locations:**

- **`GET /api/locations`: Fetch available pickup/drop-off locations.**

6. **User:**

- **`GET /api/user/profile`: Fetch user profile.**
- **`PUT /api/user/profile`: Update profile information.**

7. **Notifications:**

- **`POST /api/notifications`: Send booking confirmations or alerts.**

Sanity Schema :

1. Car Product Schema

Defines the cars available for rent, including details such as brand, type, and availability.

```
export default {

  name: 'car',
  type: 'document',
  title: 'Car',
  fields: [
     { name: 'name', type: 'string', title: 'CarName'},
     { name: 'image', type: 'image', title: 'Car Image',
options: { hotspot: true } },

    { name: 'brand', type: 'string', title: 'Brand' },
    { name: 'type', type: 'string', title: 'Type' },

    // SUV, Sedan, Hatchback, etc.

    { name: 'pricePerDay', type: 'number', title: 'Price
Per Day' },
    { name: 'availability', type: 'boolean', title:
     'Availability' },

    {
       name: 'features',
       type: 'array',
       title: 'Features',
       of: [{ type: 'string' }],

    },
```

```
    { name: 'description', type: 'text', title:
'Description' },
  ],
};
```

2. 2. Location Schema

Stores pickup and drop-off locations.

```
export default {
name: 'location',
type: 'document',
title: 'Location',
fields: [
 { name: 'city', type: 'string', title: 'City' },
{ name: 'address', type: 'string', title: 'Address' },
 { name: 'coordinates', type: 'geopoint', title: '
  Coordinates' },
],
};
```

3. User Schema

Captures user information for bookings and profile management.

```
export default {
name: 'user',
type: 'document',
title: 'User',
fields: [
{ name: 'name', type: 'string', title: 'Full Name' },
{ name: 'email', type: 'string', title: 'Email
Address' },
{ name: 'phone', type: 'string', title: 'Phone
Number' },
{ name: 'profileImage', type: 'image', title:
'Profile Image', options: { hotspot: true } },
],
};
```

4. Booking Schema

Tracks rental bookings, linking users and cars with rental details.

```
export default {

  name: 'booking',
  type: 'document',
  title: 'Booking',

  fields: [
        { name: 'user', type: 'reference', to: [{ type:
'user' }],
        title: 'User' },
        { name: 'car', type: 'reference', to: [{ type:
'car' }],

      title: 'Car' },
      { name: 'startDate', type: 'datetime', title:
'Start Date' },
      { name: 'endDate', type: 'datetime', title: 'End
Date' },
     { name: 'totalPrice', type: 'number', title: 'Total
Price' },

    {

      name: 'status',
      type: 'string',
      title: 'Status',
      options: {

        list: [

          { title: 'Pending', value: 'pending' },
          { title: 'Confirmed', value: 'confirmed' },
          { title: 'Completed', value: 'completed' },
          { title: 'Cancelled', value: 'cancelled' },
        ], },

    },

  ],

}
```

5. Order Schema

Handles payment and order details for the bookings.

```
   export default {

   name: 'order',
   type: 'document',
   title: 'Order',
   fields: [

    { name: 'user', type: 'reference', to: [{ type:
'user' }], title: 'User' },
    { name: 'booking', type: 'reference', to: [{ type:
'booking' }], title: 'Booking' },
    { name: 'orderDate', type: 'datetime', title: 'Order
Date' },

    {
      name: 'paymentStatus',
      type: 'string',
      title: 'Payment Status',
      options: {
      list: [

          { title: 'Pending', value: 'pending' },
          { title: 'Completed', value: 'completed' },
          { title: 'Failed', value: 'failed' },
       ],
      },
    },

    { name: 'transactionId', type: 'string', title:
'Transaction ID' },

    { name: 'amountPaid', type: 'number', title: 'Amount
Paid' },

  ],

};
```