CODE DEVILERABLES

Key Component Code Snippets

1. HOMEPAGE COMPONENT

```
'use client';
import Image from "next/image";
import Link from "next/link";
import { Card, CardContent, CardDescription, CardFooter, CardHeader, CardTitle } from
"../components/ui/card";
import { client } from '../sanity/lib/client';
import { urlFor } from '../sanity/lib/image';
import { useEffect, useState, Suspense } from 'react';
import { useSearchParams } from 'next/navigation';

interface Car {
  _id: string;
  name: string;
  type: string;
  image: any;
  pricePerDay: number;
  fuelCapacity: string;
  transmission: string;
  seatingCapacity: string;
}

function CarList() {
  const [currentPage, setCurrentPage] = useState(1);
  const carsPerPage = 12; // Number of cars to display per page
  const [cars, setCars] = useState<Car[]>([]);
  const searchParams = useSearchParams();
  const searchQuery = searchParams.get('search') || '';

  useEffect(() => {
    const fetchCars = async () => {
      let query = '*[_type == "car"]{_id, name, type, image, pricePerDay, fuelCapacity,
transmission, seatingCapacity}';
      const data = await client.fetch(query);
      setCars(data);
    };

    fetchCars();
  }, []);

  // Filter cars based on search query
  const filteredCars = cars.filter((car) =>
    car.name.toLowerCase().includes(searchQuery.toLowerCase())
  );

  // Pagination logic
  const indexOfLastCar = currentPage * carsPerPage;
  const currentCars = filteredCars.slice(0, indexOfLastCar);
  const totalPages = Math.ceil(filteredCars.length / carsPerPage);

  const handleShowMore = () => {
    if (currentPage < totalPages) {
      setCurrentPage(currentPage + 1);
    }
  };
```

```
  return (
    <div className="sec grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-4">
      {currentCars.map((car) => (
        <Card key={car._id} className="w-full max-w-[304px] mx-auto h-auto flex flex-col
justify-between">
          <CardHeader>
            <CardTitle className="w-full flex items-center justify-between">
              {car.name}
            </CardTitle>
            <CardDescription>{car.type}</CardDescription>
          </CardHeader>
          <CardContent className="w-full flex flex-col items-center justify-center gap-4">
            <Image src={urlFor(car.image).url()} alt={car.name} width={220} height={68} />
            <div className="flex justify-between w-full">
              <div className="bg-gray-200 p-2 rounded text-center">
                {car.fuelCapacity}
              </div>
              <div className="bg-gray-200 p-2 rounded text-center">
                {car.seatingCapacity}
              </div>
              <div className="bg-gray-200 p-2 rounded text-center">
                {car.transmission}
              </div>
            </div>
          </CardContent>
          <CardFooter className="w-full flex items-center justify-between">
            <p>${car.pricePerDay}/day</p>
            <Link href={`/cars/${car._id}`}>
              <button className="bg-[#3563e9] p-2 text-white rounded-xl w-[140px] h-[56px]">
                Rent Now
              </button>
            </Link>
          </CardFooter>
        </Card>
      ))}
      {/* Show More Button */}
      <section className="button w-full text-center mt-4">
        <button onClick={handleShowMore} disabled={currentPage === totalPages} className="bg-
[#3563e9] px-4 py-2 text-white rounded-md">
          Show More Cars
        </button>
      </section>
    </div>
  );
}

export default function Home() {
  return (
    <div className="bg-[#f6f7f9] min-h-screen p-4 sm:p-6 lg:p-20 flex flex-col gap-10 font-
[family-name:var(--font-geist-sans)]">
      {/* Popular Cars Section */}
      <section className="popular w-full flex flex-col gap-4">
        <h1 className="text-gray-500 text-lg sm:text-xl">Popular Car</h1>
        <Suspense fallback={<p>Loading cars...</p>}>
          <CarList />
        </Suspense>
      </section>
    </div>
  );
}
```

2. CHECKOUT COMPONENT:

```
'use client';

import React, { useEffect, useState } from "react";
import { useForm, SubmitHandler } from "react-hook-form";
import { zodResolver } from "@hookform/resolvers/zod";
import { z } from "zod";
import { Card, Input, Button } from "'components/ui"; // Combined imports for brevity
import { useCart } from "'context/CartContext'";
```

```
const combinedSchema = z.object({
  name: z.string().min(1, "Name is required"),
  phone: z.string().min(10, "Phone number must be at least 10 digits").regex(/^\d+$/,
"Phone number must contain only digits"),
  address: z.string().min(1, "Address is required"),
  city: z.string().min(1, "City is required"),
  pickupLocation: z.string().min(1, "Pickup location is required"),
  pickupDate: z.string().min(1, "Pickup date is required"),
  dropoffLocation: z.string().min(1, "Dropoff location is required"),
  dropoffDate: z.string().min(1, "Dropoff date is required"),
});

export default function PaymentPage() {
  const { cart, totalCost } = useCart();
  const { register, handleSubmit, formState: { errors } } = useForm({ resolver:
zodResolver(combinedSchema) });
  const [loading, setLoading] = useState(false);

  const handleCheckout = async () => {
    setLoading(true);
    // Checkout logic...
  };

  const onSubmit: SubmitHandler = async (data) => {
    // Form submission logic...
    await handleCheckout();
  };

  return (
    <div className="w-full p-4 flex flex-col lg:flex-row gap-6">
      <form onSubmit={handleSubmit(onSubmit)}>
        <Card>
          <CardHeader>
            <CardTitle>Billing Info</CardTitle>
          </CardHeader>
          <CardContent>
            <Input {...register("name")} placeholder="Your Name" />
            {errors.name && <p>{errors.name.message}</p>}
            {/* Other input fields... */}
          </CardContent>
        </Card>
        <Card>
          <CardHeader>
            <CardTitle>Rental Info</CardTitle>
          </CardHeader>
          <CardContent>
            {/* Rental info fields... */}
          </CardContent>
        </Card>
        <Button type="submit" disabled={loading}>{loading ? "Processing..." :
"Checkout"}</Button>
      </form>
      <Card>
        <CardHeader>
          <CardTitle>Rental Summary</CardTitle>
        </CardHeader>
        <CardContent>
          {cart.map(item => (
            <div key={item.id}>
              <p>{item.name}</p>
              <p>${item.totalPrice}</p>
            </div>
```

```
          ))}
          <p>Total: ${totalCost}</p>
        </CardContent>
      </Card>
    </div>
  );
}
```

3. USER PROFILE COMPONENT:

```
'use client';

import { useUser } from '@clerk/nextjs';
import { useEffect, useState } from 'react';

interface Car {
  name: string;
  model: string;
}

interface Rental {
  _id: string;
  car: Car;
  startDate: string;
  endDate: string;
  duration: number;
  totalPrice: number;
  status: string;
  paymentStatus: string;
}

export default function ProfilePage() {
  const { user } = useUser();
  const [rentals, setRentals] = useState<Rental[]>([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    if (user) {
      const fetchRentals = async () => {
        const response = await fetch(`/api/rentals?userId=${user.id}`);
        if (!response.ok) throw new Error('Failed to fetch rentals');
        const data = await response.json();
        setRentals(data.rentals);
        setLoading(false);
      };
      fetchRentals().catch(err => {
        setError('Failed to fetch rentals');
        setLoading(false);
      });
    }
  }, [user]);

  if (!user) return <div>Loading...</div>;
  if (loading) return <div>Loading rental history...</div>;
  if (error) return <div>{error}</div>;

  return (
    <div className="p-4">
      <h1 className="text-2xl font-bold">Profile</h1>
      <p><strong>Name:</strong> {user.fullName}</p>
```

```
      <h2 className="text-xl font-bold">Rental History</h2>
      {rentals.length > 0 ? (
        <ul>
          {rentals.map(rental => (
            <li key={rental._id}>
              <p><strong>Car:</strong> {rental.car.name} {rental.car.model}</p>
              <p><strong>Total Price:</strong> ${rental.totalPrice}</p>
            </li>
          ))}
        </ul>
      ) : (
        <p>No rental history found.</p>
      )}
    </div>
  );
}
```

4. CART COMPONENT

```
'use client';

import { useCart } from "../../context/CartContext";
import DatePicker from "react-datepicker";
import "react-datepicker/dist/react-datepicker.css";
import Link from "next/link";

const CartPage = () => {
  const { cart, removeFromCart, updateRentalDates, totalCost } = useCart();

  const handleDateChange = (id: string, startDate: Date | null, endDate: Date | null)
=> {
    if (startDate && endDate) {
      updateRentalDates(id, startDate.toISOString(), endDate.toISOString());
    }
  };

  return (
    <div className="max-w-7xl mx-auto p-4">
      <h1 className="text-2xl font-bold mb-6">Your Cart</h1>
      {cart.length === 0 ? (
        <p>Your cart is empty.</p>
      ) : (
        <div className="space-y-6">
          {cart.map(item => (
            <div key={item.id} className="border p-4 rounded-lg">
              <img src={item.image} alt={item.name} className="rounded-lg w-full h-48
object-cover" />
              <h2 className="text-xl font-bold">{item.name}</h2>
              <DatePicker selected={new Date(item.rentalStartDate)} onChange={date =>
handleDateChange(item.id, date, new Date(item.rentalEndDate))} />
              <p>Total: ${item.totalPrice}</p>
              <button onClick={() => removeFromCart(item.id)} className="bg-red-600
text-white">Remove</button>
            </div>
          ))}
          <div className="flex justify-between mt-6">
            <p>Total: ${totalCost}</p>
            <Link href="/payment">
              <button className="bg-green-600 text-white">Proceed to Checkout</button>
            </Link>
          </div>
```

```
        </div>
      )}
    </div>
  );
};

export default CartPage;
```

5. PRODUCT DETAIL CMPONENT

```
'use client';

import { useEffect, useState } from "react";
import { useParams } from "next/navigation";
import { client } from "../../../sanity/lib/client";
import { urlFor } from "../../../sanity/lib/image";
import { useCart } from "../../../context/CartContext";
import DatePicker from "react-datepicker";
import "react-datepicker/dist/react-datepicker.css";
import Link from "next/link";

const CarPage = () => {
 const params = useParams<{ id: string }>();
 const id = params?.id || "";
 const [car, setCar] = useState<Car | null>(null);
 const [loading, setLoading] = useState(true);
 const [error, setError] = useState<string | null>(null);
 const { addToCart } = useCart();

 useEffect(() => {
  if (id) {
   const fetchCar = async () => {
    const carQuery = `*[_type == "car" && _id == $id][0]{_id, name, model, year, type,
image, brand, fuelCapacity, transmission, seatingCapacity, pricePerDay}`;
    const fetchedCar = await client.fetch<Car>(carQuery, { id });
    setCar(fetchedCar);
    setLoading(false);
   };

   fetchCar().catch(err => {
    setError("Error fetching car data");
    setLoading(false);
   });
  }
 }, [id]);

 if (loading) return <div>Loading...</div>;
 if (error) return <div>{error}</div>;
 if (!car) return <div>Car not found.</div>;

 return (
  <div>
       <h1>{car.name}</h1>
       <p>Price: ${car.pricePerDay}/day</p>
       {/* Additional car details and rental form */}
     </div>
 );
```

```
};

export default CarPage;
```

API Integration & Dynamic Routing

```
import { NextResponse } from 'next/server';
import { client } from 'sanity/lib/client';

export async function GET(request: Request) {
 const { searchParams } = new URL(request.url);
 const userId = searchParams.get('userId');

 if (!userId) {
  return NextResponse.json({ success: false, error: 'User ID is required' }, { status: 400 });
 }

 try {
  const rentals = await client.fetch(`*[_type == "rental" && userId == $userId]{
       _id,
       car->{
         name,
         model
       },
       startDate,
       endDate,
       duration,
       totalPrice,
       status,
       paymentStatus
     }`, { userId });
  return NextResponse.json({ success: true, rentals }, { status: 200 });
 } catch (error) {
  console.error('Error fetching rentals:', error);
  return NextResponse.json({ success: false, error: 'Failed to fetch rentals' }, { status: 500 });
 }
}

export async function POST(request: Request) {
 try {
  const { userId, carId, startDate, endDate, duration, totalPrice } = await request.json();

  const rental = {
   _type: 'rental',
   userId: userId,
   car: {
    _type: 'reference',
    _ref: carId,
   },
   startDate: startDate,
   endDate: endDate,
   duration: duration,
   totalPrice: totalPrice,
   status: 'Pending',
   paymentStatus: 'Unpaid',
```

```
  };

  const createdRental = await client.create(rental);
  return NextResponse.json({ success: true, rental: createdRental }, { status: 201 });
 } catch (error) {
  console.error('Error creating rental:', error);
  return NextResponse.json({ success: false, error: 'Failed to create rental' }, { status: 500 });
 }
}
```

Dynamic Routing Code Snippet

This snippet illustrates how to implement dynamic routing for rental details based on the rental ID.

```
import { NextResponse } from 'next/server';
import { client } from 'sanity/lib/client';

export async function GET(request: Request, { params }: { params: { id: string } }) {
 const { id } = params;

 try {
  const rental = await client.fetch(`*[_type == "rental" && _id == $id]{
       _id,
       car->{
         name,
         model
       },
       startDate,
       endDate,
       duration,
       totalPrice,
       status,
       paymentStatus
     }`, { id });

  if (!rental) {
   return NextResponse.json({ success: false, error: 'Rental not found' }, { status: 404 });
  }

  return NextResponse.json({ success: true, rental }, { status: 200 });
 } catch (error) {
  console.error('Error fetching rental:', error);
  return NextResponse.json({ success: false, error: 'Failed to fetch rental' }, { status: 500 });
 }
}
```