

# Car Rental Web Application (About My Angular Project)

THE **CAR RENTAL WEB APPLICATION** BUILT USING ANGULAR PROVIDES A SEAMLESS PLATFORM FOR USERS TO EXPLORE AND RENT CARS EFFICIENTLY. IT FEATURES SECURE AUTHENTICATION, DYNAMIC CAR LISTINGS WITH FILTERS, RENTAL BOOKING AND MANAGEMENT FOR USERS, AND A SUPPORT SECTION FOR USER QUERIES.

ANGULAR'S **COMPONENT-BASED ARCHITECTURE**, **TWO-WAY DATA BINDING**, AND **EFFICIENT ROUTING** ENSURE A **RESPONSIVE, SCALABLE, AND USER-FRIENDLY EXPERIENCE**, MAKING IT IDEAL FOR MANAGING THE **CAR RENTAL PROCESS** EFFECTIVELY. THIS PROJECT INCORPORATES **LOCAL STORAGE**, **API CALLS**, **FORM HANDLING**, **ROUTING GUARDS**, AND **CUSTOM PIPES** TO ENHANCE FUNCTIONALITY.

# Features Used in This Project

The **Car Rental Web Application** is built with Angular and incorporates several key features to ensure a smooth user experience and efficient car rental management.

## 1. Components

- Modular and reusable UI elements are created using Angular components.
- Example: HeaderComponent, FooterComponent, CarListComponent, BookingComponent, ContactComponent, etc.

## 2. Parent-Child Components

- Child components receive data from parent components using @Input().
- Example: Car details are passed from CarListComponent (parent) to CarCardComponent (child).

## 3. Directives (Structural & Attribute)

- **Structural directives** (\*ngIf, \*ngFor) are used to dynamically render content.
- **Attribute directives** (ngClass, ngStyle) modify the appearance of elements dynamically.
- Example: Displaying available cars dynamically in CarListComponent.

# Features Used in This Project

## 4. @Input and @Output Decorators

- **@Input**: Passes data from parent to child components (e.g., sending car data to car cards).
- **@Output**: Emits events from child to parent (e.g., booking confirmation button click).

## 5. Routing and Routing Parameters

- Implemented RouterModule for navigation between pages.
- Used **route parameters** to fetch and display details for a selected car (/booking/:id).
- Example: Clicking on a car redirects to BookingComponent with the selected car's details.

## 6. Local Storage

- Stores user session data and rental history for a seamless experience.
- Example: Saving login session, booking details, and previously viewed cars.

# Features Used in This Project

## 7. Routing Guard

- AuthGuard ensures that only authenticated users can access the booking page.
- Example: If a user tries to access /booking without logging in, they are redirected to the login page.

## 8. API Call

- Integrated with a mock JSON API to simulate real-time car listings and bookings.
- Example: Fetching car details from car.json.

## 9. Form Handling (Template & Model-Based Approach)

- **Template-driven forms** for login and contact page.
- **Reactive forms** for car booking with validation.
- Example: BookingComponent uses form validation to ensure correct user inputs.

# Features Used in This Project

## 11. Inbuilt Pipes

- Used Angular pipes like uppercase, currency, and date to format data dynamically.
- Example: Displaying car prices with currency pipe (₹5000/day).

## 12. Custom Pipes

- Created a custom pipe to format car names (CarNamePipe).
- Example: Transforming "honda-civic" into "Honda Civic".

## 13. Authentication System

- Users log in using email or phone number and password.
- Login persists using local storage to prevent repeated logins.

## 14. Responsive UI with Bootstrap

- Used Bootstrap for a **modern, red-and-white theme**.
- Header and Footer** are consistent across all pages.

# Component Architecture

- **Modular Design:** The application is divided into multiple reusable components for better organization and scalability.
- **Header & Footer Components:** Provide a consistent layout across all pages.
- **Feature-Specific Components:** Includes Home, Car List, Booking, and Contact components for individual sections of the application.
- **Parent-Child Communication:** Uses **@Input** and **@Output** decorators to pass data between components.
- **Directives:** Utilizes **structural directives** (\*ngIf, \*ngFor) for dynamic rendering and **attribute directives** for UI enhancements.
- **Routing & Route Guards:** Implements navigation between pages with Angular Router and secures routes where necessary.
- **Services & Observables:** Manages API calls and data sharing across components efficiently.
- **Scalability & Maintainability:** Follows best practices to ensure future enhancements and code reusability.

# Installing Required Dependencies

- ✓ **angular.json** does not manage dependencies directly.
- ✓ Dependencies are installed and managed through the **package.json** file.
- ✓ After installing, Angular uses **angular.json** to define:
  1. **Styles** (CSS, SCSS)
  2. **Scripts** (JS, external libraries)
  3. Asset paths, build options, etc.

**To add any dependency to an Angular project:**  
(npm install package-name)

- ✓ This adds the package to dependencies in **package.json** file.
- ✓ With npm install command Node Modules will be installed in the project

# Using Standalone Component

- Standalone components allow you to create Angular components **without the need for NgModules**.
- They simplify the structure by removing the need for **app.module.ts**.
- Introduced in Angular 14+ and now fully supported in Angular 17+.
- To generate a Standalone Component:  
**ng g c my-component --standalone**

## Benefits of Using Standalone Components:

- No need to create app.module.ts
- Faster startup and smaller bundle size
- Improved maintainability and modularity
- Easier to manage dependencies

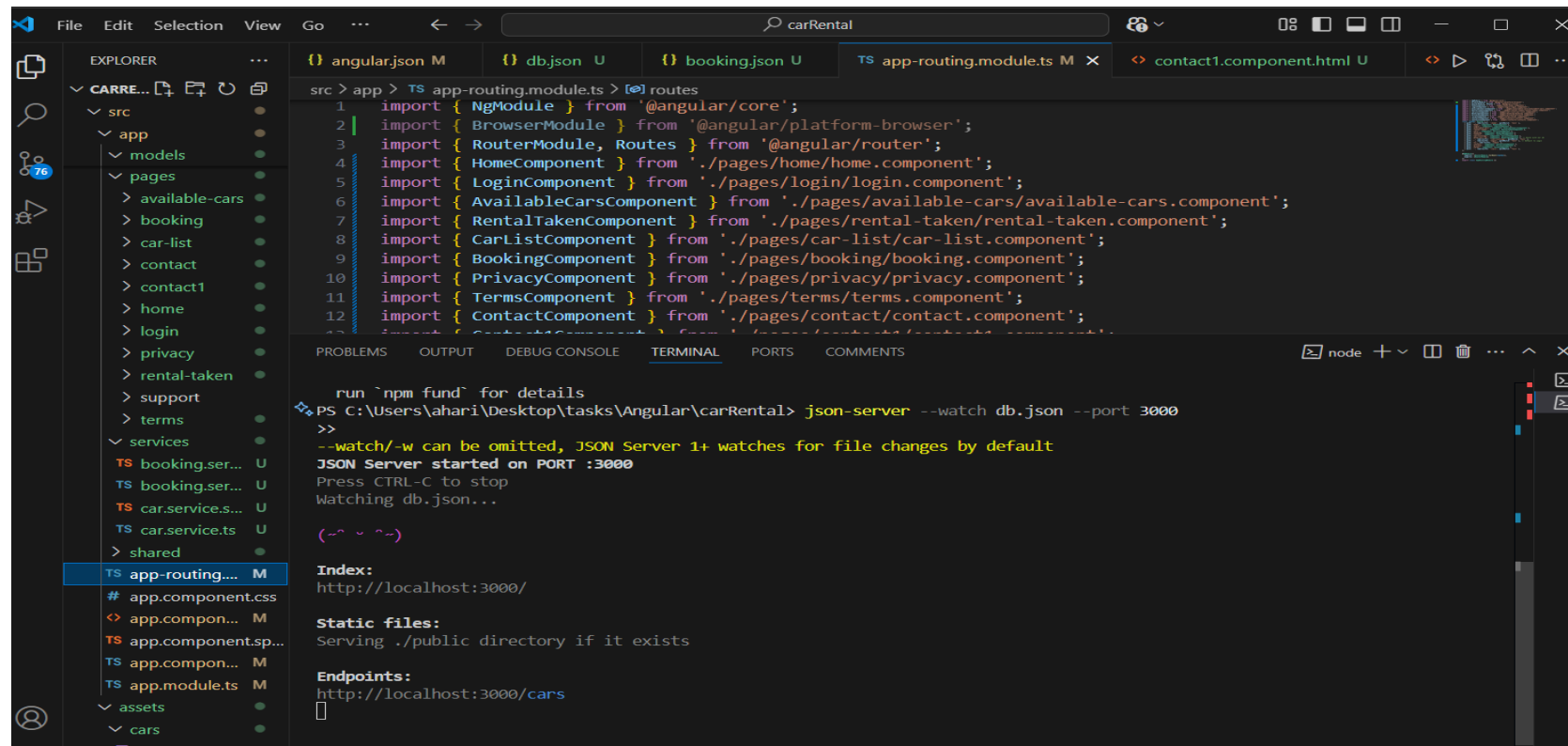


# Bootstrapping in the Car Rental Application

Bootstrapping is the process of initializing an Angular application by loading the root component. Traditionally, this is done using an AppModule, where the @NgModule decorator includes a bootstrap array that specifies the root component, such as AppComponent. However, Angular also supports standalone bootstrapping using bootstrapApplication(), which eliminates the need for an AppModule, allowing components to be bootstrapped directly. This approach reduces load times, minimizes bundle size, and simplifies project structure.

In our **Car Rental Application**, we follow the traditional bootstrapping method, ensuring a modular and scalable architecture. This structured approach helps maintain the application efficiently, allowing for better organization of services, components, and dependencies.

# Running the Angular Project by starting a json server:



The screenshot shows a code editor with the following components:

- EXPLORER:** Displays the project structure. The `src` directory contains `app`, `models`, and `pages`. The `pages` directory contains several component files like `available-cars`, `booking`, `car-list`, `contact`, `contact1`, `home`, `login`, `privacy`, `rental-taken`, `support`, `terms`, `services`, and `shared`. The `services` directory contains `booking.service.ts`, `car.service.ts`, and `car.service.ts`. The `assets` directory contains `cars`.
- EDITOR:** Shows the `src > app > TS app-routing.module.ts` file. The code defines the routes for the application, importing various components from the `pages` directory and the `angular/core` and `angular/router` packages.
- TERMINAL:** Shows the command `run `npm fund` for details` and the output of the `json-server` command. The output indicates that the JSON server is started on port 3000 and is watching for file changes in `db.json`.

```
src > app > TS app-routing.module.ts > routes
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { RouterModule, Routes } from '@angular/router';
4 import { HomeComponent } from './pages/home/home.component';
5 import { LoginComponent } from './pages/login/login.component';
6 import { AvailableCarsComponent } from './pages/available-cars/available-cars.component';
7 import { RentalTakenComponent } from './pages/rental-taken/rental-taken.component';
8 import { CarListComponent } from './pages/car-list/car-list.component';
9 import { BookingComponent } from './pages/booking/booking.component';
10 import { PrivacyComponent } from './pages/privacy/privacy.component';
11 import { TermsComponent } from './pages/terms/terms.component';
12 import { ContactComponent } from './pages/contact/contact.component';
```

```
run `npm fund` for details
PS C:\Users\ahari\Desktop\tasks\Angular\carRental> json-server --watch db.json --port 3000
>>
--watch/-w can be omitted, JSON Server 1+ watches for file changes by default
JSON Server started on PORT :3000
Press CTRL-C to stop
Watching db.json...

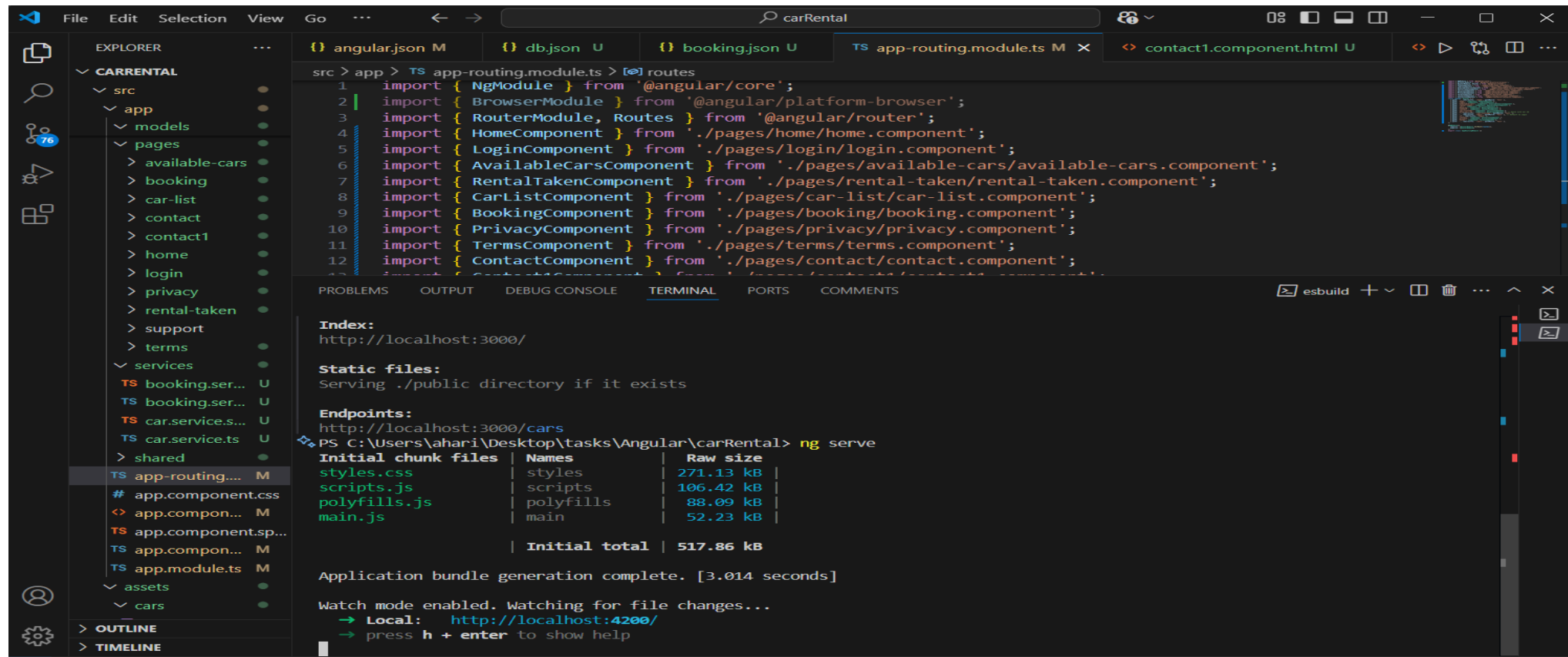
(~^ ~ ~)

Index:
http://localhost:3000/

Static files:
serving ./public directory if it exists

Endpoints:
http://localhost:3000/cars
```

# Running the Angular Project After Starting JSON Server



The screenshot displays the Visual Studio Code interface for an Angular project named 'carRental'. The Explorer sidebar on the left shows the project structure, including the 'src' directory with 'app', 'models', 'pages', and 'services' subdirectories. The Editor window shows the 'app-routing.module.ts' file, which imports various Angular modules and components. The Terminal window at the bottom shows the output of the 'ng serve' command, indicating that the application bundle generation is complete and the application is running on 'http://localhost:4200/'.

```
src > app > TS app-routing.module.ts > [0] routes
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { RouterModule, Routes } from '@angular/router';
4 import { HomeComponent } from '../pages/home/home.component';
5 import { LoginComponent } from '../pages/login/login.component';
6 import { AvailableCarsComponent } from '../pages/available-cars/available-cars.component';
7 import { RentalTakenComponent } from '../pages/rental-taken/rental-taken.component';
8 import { CarListComponent } from '../pages/car-list/car-list.component';
9 import { BookingComponent } from '../pages/booking/booking.component';
10 import { PrivacyComponent } from '../pages/privacy/privacy.component';
11 import { TermsComponent } from '../pages/terms/terms.component';
12 import { ContactComponent } from '../pages/contact/contact.component';
```

**Index:**  
http://localhost:3000/

**Static files:**  
Serving ./public directory if it exists

**Endpoints:**  
http://localhost:3000/cars

PS C:\Users\ahari\Desktop\tasks\Angular\carRental> ng serve

Initial chunk files	Names	Raw size
styles.css	styles	271.13 kB
scripts.js	scripts	106.42 kB
polyfills.js	polyfills	88.09 kB
main.js	main	52.23 kB
<b>Initial total</b>		<b>517.86 kB</b>

Application bundle generation complete. [3.014 seconds]

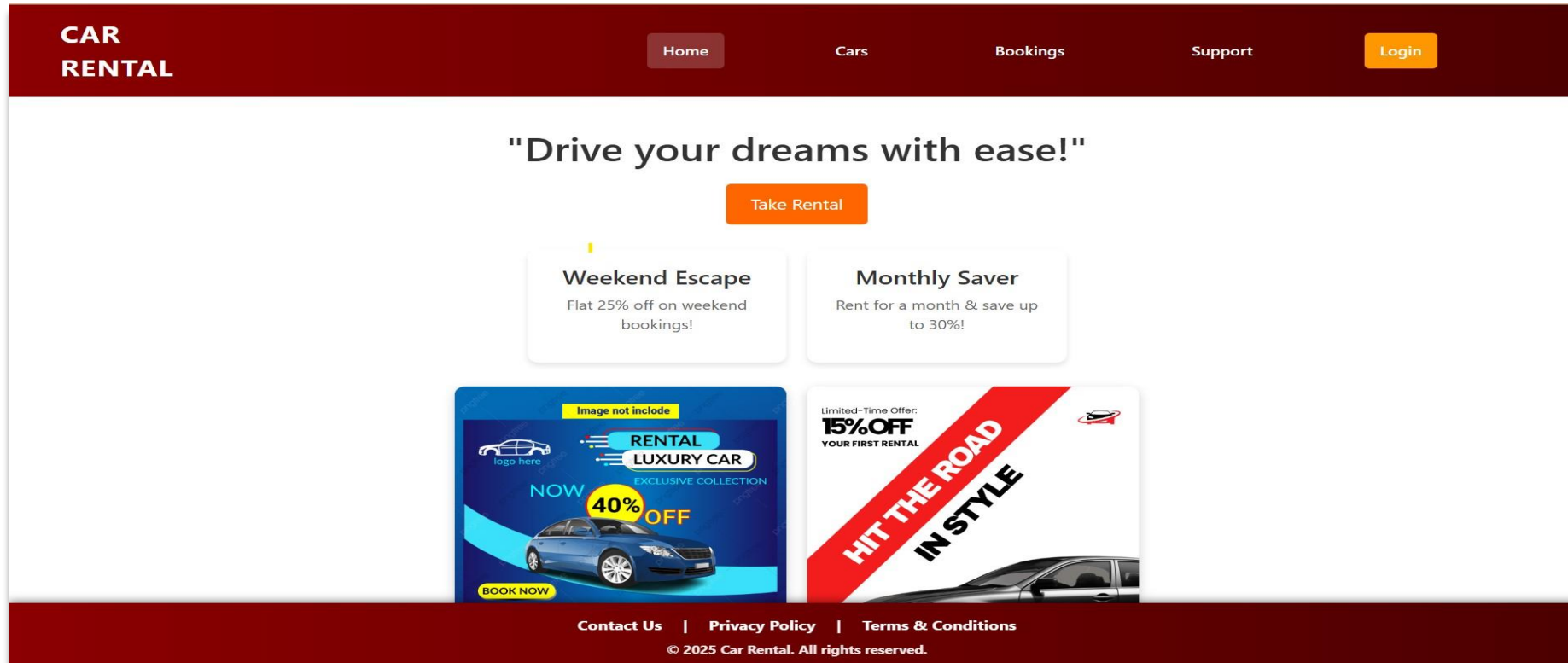
Watch mode enabled. Watching for file changes...

→ Local: <http://localhost:4200/>  
→ press **h + enter** to show help

# Pages Included

- **Login Page** – Allows users to log in before booking a car.
- **Home Page** – Displays a motivational quote, offers, and advertisements.
- **Available Cars Page** – Lists all available cars with images, prices, and booking options.
- **Rental Taken Page** – Shows rented cars (initially empty).
- **Contact Page** – Displays contact info, social media links, FAQs, and customer reviews.

This project **follows best practices** for **scalability, modularity, and maintainability**, making it a complete **Single Page Application (SPA)** with **efficient routing and dynamic data handling**.



## Home Page:

The **Home Page** serves as the main landing page, designed to engage users with:

A **Quote** related to travel and adventure, Featured **offers and advertisements** for promotions, A **navigation bar** with links to other sections (e.g., Available Cars, Contact, Rentals)., Uses **Angular Components** to structure sections dynamically. Employs **custom styling (CSS/Bootstrap)** for an attractive layout.

AVAILABLE CARS FOR RENT



shutterstock.com - 2484157269

Toyota Corolla

Brand: Toyota

Price Per Day: \$50

Rent Now



Honda Civic

Brand: Honda

Price Per Day: \$55

Rent Now



BMW X5

Brand: BMW

Price Per Day: \$120

Rent Now



Hyundai Elantra

Brand: Hyundai

Price Per Day: \$45

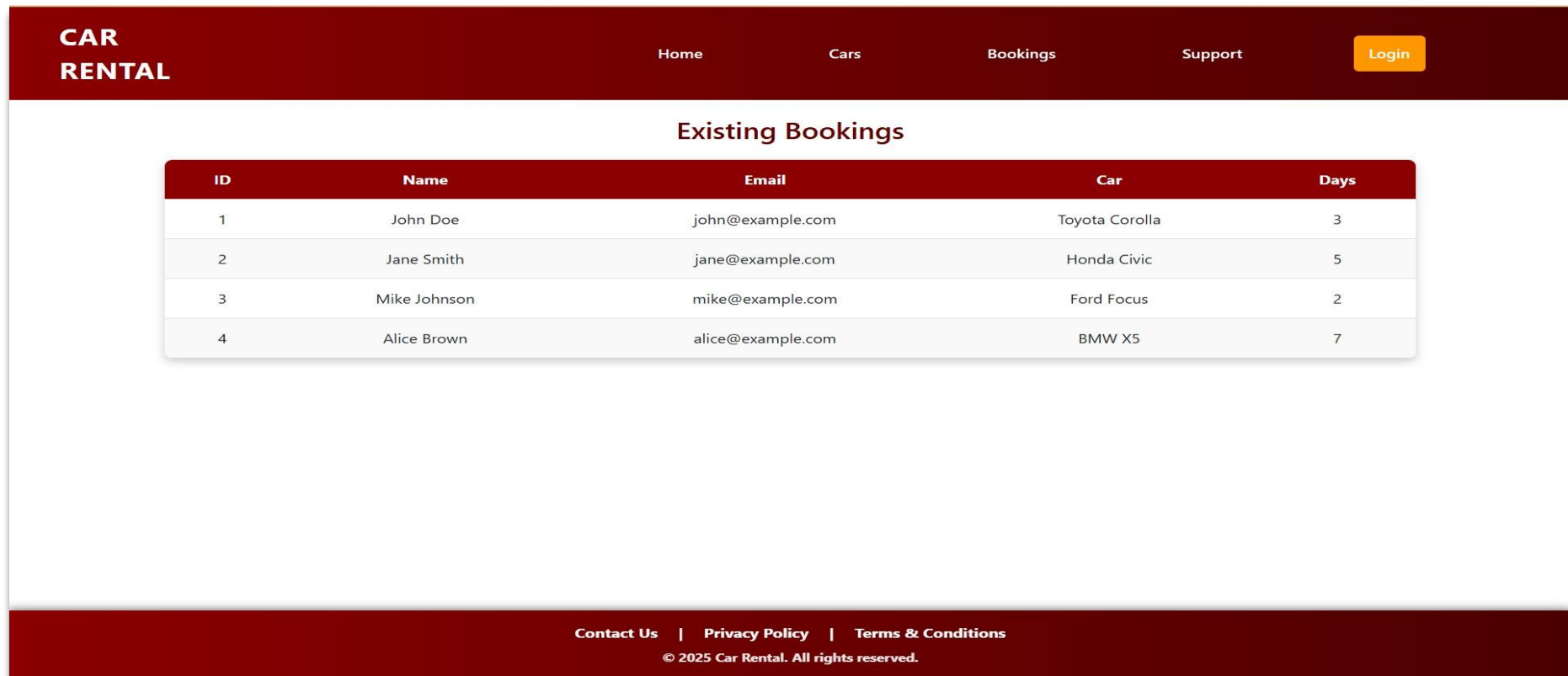
Rent Now



[Contact Us](#) | [Privacy Policy](#) | [Terms & Conditions](#)

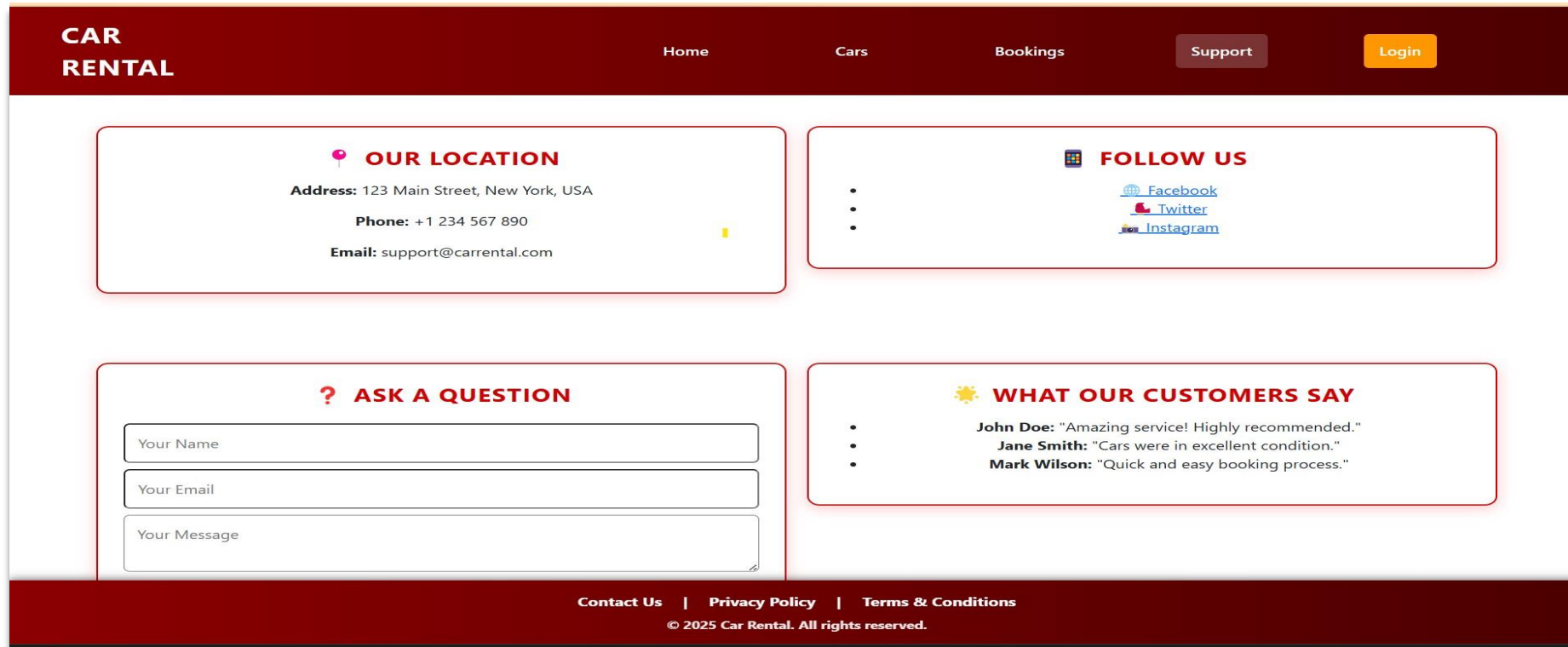
© 2025 Car Rental. All rights reserved.

The **Available Cars Page** displays a list of all cars available for rent, showcasing images, names, prices, and booking options. Each car has a "Book Now" button, allowing users to proceed with reservations. The data is fetched using Angular Services and HTTP Requests from a mock JSON Server API. Structural directives like `*ngFor` dynamically render the car listings, while attribute directives such as `ngClass` and `ngStyle` enhance UI elements to indicate availability.



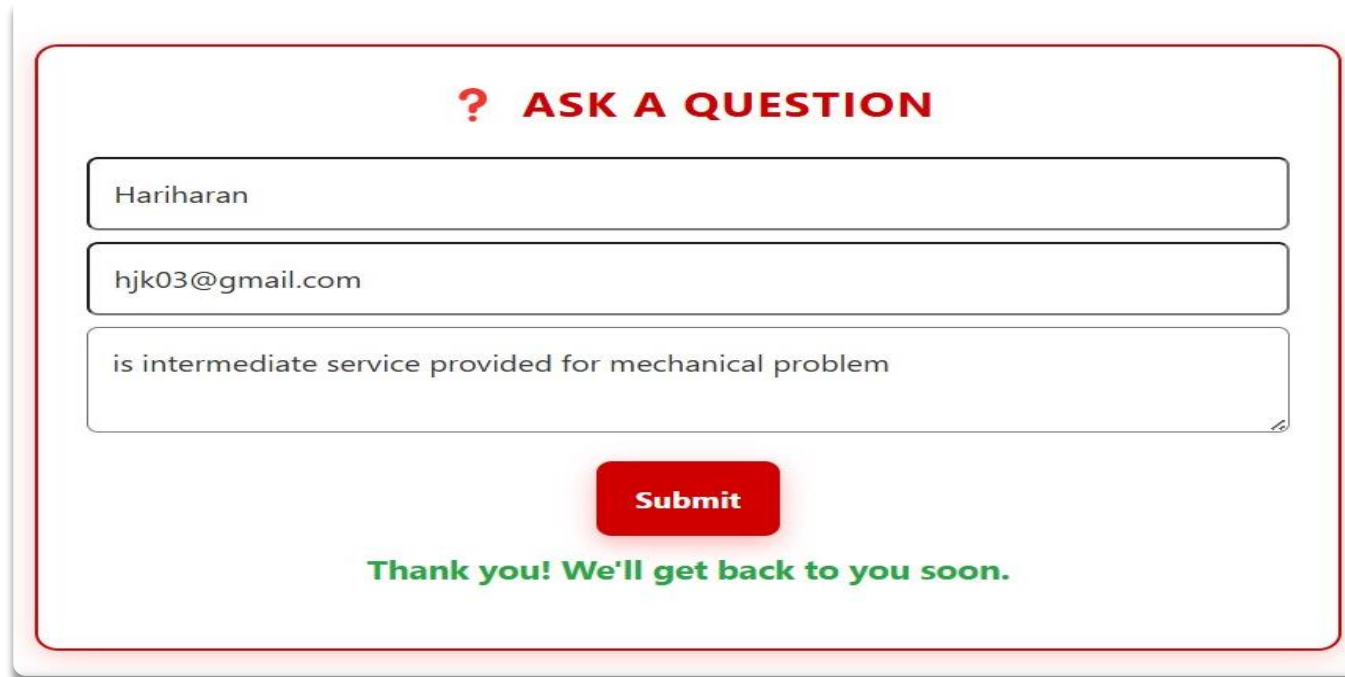
The **Bookings Page** allows users to review and manage their car rental reservations. It displays details such as car name, booking ID, rental duration, and total cost. Users can view their past and current bookings, ensuring a smooth tracking experience. This page utilizes Angular's **Services and Observables** to fetch booking data and display it dynamically. Local Storage is also used to retain user booking details. Angular's **Routing Parameters** enable navigation to detailed booking information, and form handling is implemented for modifying or canceling bookings in future enhancements.





The **Support Page** provides essential customer support information, including phone numbers, email addresses, and office locations. It also features social media links for quick access, an FAQ section answering common user queries, and a customer review section displaying feedback with ratings. Using Bootstrap cards and a structured Angular component layout, this page ensures users can easily find support and stay connected with the service.





**? ASK A QUESTION**

Hariharan

hjk03@gmail.com

is intermediate service provided for mechanical problem

**Submit**

**Thank you! We'll get back to you soon.**

The **Ask a Question Card** on the **Contact Page** allows users to submit their queries regarding car rentals, services, or any other concerns. This feature is implemented using **Angular Forms (Template-Driven or Reactive Forms)** to collect user input, such as name, email, and question. The form uses **two-way data binding with ngModel** for real-time updates and validation to ensure correct input. Upon submission, the data can be stored locally or sent to a backend API for further processing. This enhances user engagement and provides a direct communication channel with the service provider.



**Sedan X**

Price: \$50/day

Take Rental



**SUV Y**

Price: \$70/day

Take Rental



**Hatchback Z**

Price: \$40/day

Take Rental

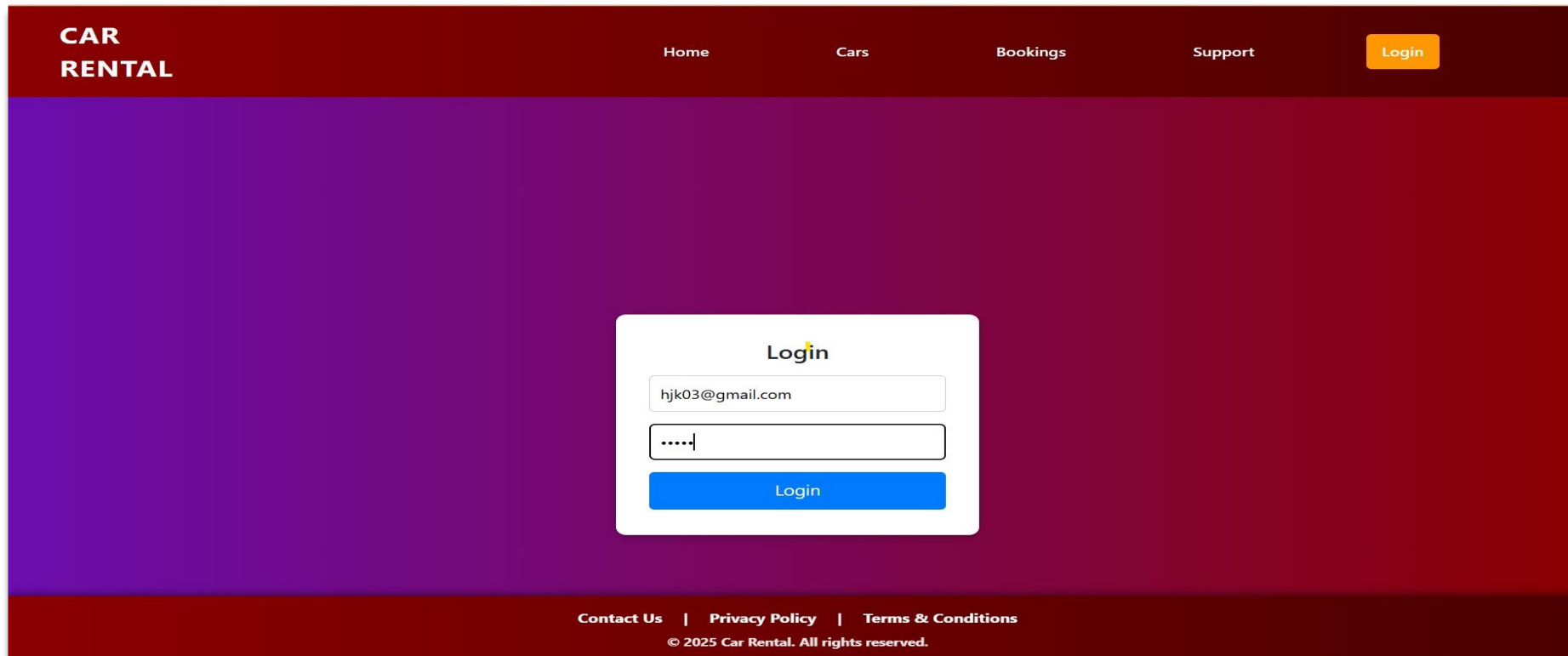


**Convertible A**

Price: \$90/day

Take Rental

The **Rental Taken Page** keeps track of the cars that users have booked. Initially empty, it updates dynamically once a user completes a booking. The page leverages Angular's Local Storage or Services with Observables to persist and retrieve booking data. It also incorporates @Input and @Output decorators for seamless component communication, allowing users to view their active rentals. Future additions could include a return or cancellation feature for modifying bookings.

The image shows a web application interface for a car rental service. The top navigation bar is dark red with the text 'CAR RENTAL' on the left and links for 'Home', 'Cars', 'Bookings', 'Support', and a 'Login' button on the right. The main content area has a purple-to-red gradient background. In the center is a white login form titled 'Login' with a yellow star icon. It contains two input fields: the first has the email 'hjk03@gmail.com' and the second has masked characters '.....'. Below the fields is a blue 'Login' button. The footer is dark red and contains links for 'Contact Us', 'Privacy Policy', and 'Terms & Conditions', followed by the copyright notice '© 2025 Car Rental. All rights reserved.'

CAR RENTAL

Home Cars Bookings Support Login

### Login

hjk03@gmail.com

.....

Login

Contact Us | Privacy Policy | Terms & Conditions

© 2025 Car Rental. All rights reserved.

The **Login Page** serves as the authentication gateway for users before they can access the booking features. It includes input fields for email or phone number and password, along with a login button to authenticate users. Upon successful login, users are redirected to the home page. The form utilizes Angular's Template-driven or Reactive Forms for validation, ensuring secure user authentication. Future enhancements could include a registration option for new users.

# Conclusion

THE **CAR RENTAL WEB APPLICATION** IS A DYNAMIC AND USER-FRIENDLY PLATFORM BUILT USING **ANGULAR**, DEMONSTRATING KEY FRONTEND DEVELOPMENT CONCEPTS. IT FEATURES **COMPONENT-BASED ARCHITECTURE, DYNAMIC DATA HANDLING WITH JSON SERVER, SECURE ROUTING, AND INTERACTIVE UI ELEMENTS**. THE PROJECT SEAMLESSLY INTEGRATES **SERVICES, OBSERVABLES, API CALLS, AND LOCAL STORAGE**, ENSURING EFFICIENT DATA FLOW AND STATE MANAGEMENT.

THROUGH STRUCTURED NAVIGATION AND MODULAR DESIGN, THE APPLICATION PROVIDES AN INTUITIVE EXPERIENCE FOR USERS TO **BROWSE, BOOK, AND MANAGE CAR RENTALS**. THIS PROJECT SERVES AS A **SCALABLE AND MAINTAINABLE FOUNDATION** FOR FUTURE ENHANCEMENTS, OFFERING REAL-WORLD APPLICABILITY IN **RENTAL SERVICES AND BOOKING SYSTEMS**.