

Piri Reis University

Faculty of Economics and Administrative Sciences

**Management Information Systems Department**

**GRADUATION THESIS – YBS421**

**Rent A Car Website**

**İlayda Paslı**

**201835009**

Advisor: Asst. Prof. Dr. ALİ KILINÇ

İstanbul (May), 2024

FOREWORD

The inspiration for this thesis began during my undergraduate studies when I first encountered the challenges and opportunities within the car rental industry. My passion for web development and a keen interest in solving real-world problems motivated me to pursue this project. The goal was to create a modern, efficient, and user-friendly daily car rental platform that leverages the latest web technologies to improve the rental experience for both users and administrators.

This work would not have been possible without the guidance and support of several individuals. I am deeply grateful to my advisor, Dr. Ali Kılınç, for his invaluable feedback and encouragement throughout the research process. I would also like to thank my family and friends for their unwavering support and patience. Additionally, I appreciate the assistance of my colleagues, who provided constructive critiques and insights that significantly contributed to the refinement of this project.

During the development of this platform, I encountered numerous technical challenges and learning opportunities. These experiences not only enhanced my technical skills but also deepened my understanding of the complexities involved in building scalable and secure web applications.

This thesis is organized into several key sections. The first chapter provides an overview of the current state of the car rental industry and the technological advancements driving it. Subsequent chapters delve into the design and implementation of the platform, including detailed discussions on the RESTful architecture, security measures, and user interface design. The final chapter presents the findings, conclusions, and potential future directions for this project.

I hope this work will contribute to the field of web development and offer valuable insights for those interested in developing similar applications.

ABSTRACT

This thesis explores the development of a daily car rental platform using ASP.NET Core Web API and Angular. The primary goal of the research is to design a modern, efficient, and user-friendly system that facilitates the process of renting vehicles on a daily basis. The platform leverages a RESTful architecture to ensure flexibility and scalability, incorporating advanced features such as secure online payment, dynamic vehicle filtering, and an administrative panel for effective management of rentals and users. The methodology involves a detailed analysis of user requirements, followed by the implementation of various front-end and back-end technologies to create an integrated solution. Key findings indicate that the platform significantly improves the user experience by offering a seamless and secure rental process, supported by robust security measures and a high-performance infrastructure. The results demonstrate the platform's potential to meet the growing demand for reliable and accessible car rental services. This research contributes to the field of web development and e-commerce by presenting a comprehensive approach to building a scalable and user-centric car rental system.

**Keywords: Web Programming, .Net Core, Web API Project**

ÖZET

Bu tez, ASP.NET Core Web API ve Angular kullanarak günlük araba kiralama platformunun geliştirilmesini incelemektedir. Araştırmanın birincil amacı, araçların günlük olarak kiralanmasını kolaylaştıran modern, verimli ve kullanıcı dostu bir sistem tasarlamaktır. Platform, esneklik ve ölçeklenebilirlik sağlamak için RESTful mimarisinden yararlanarak güvenli çevrimiçi ödeme, dinamik araç filtreleme ve kiralamaların ve kullanıcıların etkili yönetimi için bir yönetim paneli gibi gelişmiş özellikler içermektedir. Metodoloji, kullanıcı gereksinimlerinin detaylı bir analizini ve entegre bir çözüm oluşturmak için çeşitli ön uç ve arka uç teknolojilerinin uygulanmasını içermektedir. Temel bulgular, platformun güvenli ve sorunsuz bir kiralama süreci sunarak kullanıcı deneyimini önemli ölçüde iyileştirdiğini göstermektedir; bu, güçlü güvenlik önlemleri ve yüksek performanslı bir altyapı ile desteklenmektedir. Sonuçlar, platformun güvenilir ve erişilebilir araba kiralama hizmetlerine yönelik artan talebi karşılama potansiyelini ortaya koymaktadır. Bu araştırma, ölçeklenebilir ve kullanıcı odaklı bir araba kiralama sistemi oluşturma konusunda kapsamlı bir yaklaşım sunarak web geliştirme ve e-ticaret alanına katkıda bulunmaktadır.

Anahtar Kelimeler: Restfull mimarisi, Web API projesi,Angular

ACKNOWLEDGMENTS

I would like to thank my project advisor Dr. Ass. Ali Kılınç for his support and encouragement.

TABLE OF CONTENTS

[FOREWORD v](#_Toc167108109)

[ABSTRACT iii](#_Toc167108110)

[ÖZET iv](#_Toc167108111)

[ACKNOWLEDGMENTS v](#_Toc167108112)

[TABLE OF CONTENTS vi](#_Toc167108113)

[ABBREVIATIONS viii](#_Toc167108114)

[LIST OF FIGURES ix](#_Toc167108115)

[1. INTRODUCTION 1](#_Toc167108116)

[1.1. The Aim of the Project 1](#_Toc167108117)

[1.2. Scientific Innovations and Contributions 2](#_Toc167108118)

[2. RENT A CAR PROJECT REVIEW 3](#_Toc167108119)

[2.1. Rent a Car Website 4](#_Toc167108120)

[2.2. Studies carried out in the Car Rental Sector 4](#_Toc167108121)

[3. CAR RENTAL PROJECT ANALYSIS 5](#_Toc167108122)

[3.1. The Technologies Used on Our Site 6](#_Toc167108123)

[3.2. Backend Side 7](#_Toc167108124)

[3.3. Frontend Side 8](#_Toc167108125)

[3.4. Database and Test Environment 8](#_Toc167108126)

[4. CAR RENTAL AND THE USE OF COMPUTER SYSTEMS IN CAR RENTAL 9](#_Toc167108127)

[4.1. Project Layers 10](#_Toc167108128)

[4.2. Diagrams and Tables 11](#_Toc167108129)

[5. SOFTWARE COMPONENTS 13](#_Toc167108130)

[5.1. Backend 14](#_Toc167108131)

[5.2. Frontend 21](#_Toc167108132)

[5.3. SQL Database 31](#_Toc167108133)

[6. INSTALLATION PACKAGES 36](#_Toc167108134)

[6.1. Test Environment and Post Men 38](#_Toc167108135)

[6.2. Project Outputs 45](#_Toc167108136)

[7. RESULTS AND RECOMMENDATIONS 50](#_Toc167108137)

[7.1. Result 50](#_Toc167108138)

[7.2. Recommendations 50](#_Toc167108139)

[REFERENCES 51](#_Toc167108140)

[CURRICULUM VITAE 52](#_Toc167108141)

ABBREVIATIONS

MVS : Microsoft Visual Studio

MVSC : Microsoft Visual Studio Code

SSMS : Sql Server Management Studio

EF : Entity Framework

MES : Microsoft.EntityFrameworkCore.SqlServer

MIT : Microsoft.IdentityModel.Token

MITJ : Microsoft.IdentityModel.Tokens.Jwt

OD :Over Design

FV : Fluent Validation

NSL : NetStandart.Library

ST : Security Token

LIST OF FIGURES

[Figure 4.1 9](#_Toc165238497)

[Figure 4.2 10](#_Toc165238498)

[Figure 4.3 11](#_Toc165238499)

[Figure 4.4 11](#_Toc165238500)

[Figure 4.5 12](#_Toc165238501)

[Figure 5.1 14](#_Toc165238502)

[Figure 5.2 15](#_Toc165238503)

[Figure 5.3 16](#_Toc165238504)

[Figure 5.4 17](#_Toc165238505)

[Figure 5.5 18](#_Toc165238506)

[Figure 5.6 19](#_Toc165238507)

[Figure 5.7 19](#_Toc165238508)

[Figure 5.8 20](#_Toc165238509)

[Figure 5.9 21](#_Toc165238510)

[Figure 5.10 22](#_Toc165238511)

[Figure 5.11 22](#_Toc165238512)

[Figure 5.12 23](#_Toc165238513)

[Figure 5.13 24](#_Toc165238514)

[Figure 5.14 25](#_Toc165238515)

[Figure 5.15 26](#_Toc165238516)

[Figure 5.16 27](#_Toc165238517)

[Figure 5.17 28](#_Toc165238518)

[Figure 5.18 28](#_Toc165238519)

[Figure 5.19 31](#_Toc165238520)

[Figure 5.20 31](#_Toc165238521)

[Figure 5.21 32](#_Toc165238522)

[Figure 5.22 32](#_Toc165238523)

[Figure 5.23 33](#_Toc165238524)

[Figure 5.24 33](#_Toc165238525)

[Figure 5.25 34](#_Toc165238526)

[Figure 5.26 34](#_Toc165238527)

[Figure 6.1 37](#_Toc165238528)

[Figure 6.2 38](#_Toc165238529)

[Figure 6.3 39](#_Toc165238530)

[Figure 6.4 39](#_Toc165238531)

[Figure 6.5 40](#_Toc165238532)

[Figure 6.6 41](#_Toc165238533)

[Figure 6.7 41](#_Toc165238534)

[Figure 6.8 42](#_Toc165238535)

[Figure 6.9 43](#_Toc165238536)

[Figure 6.10 43](#_Toc165238537)

[Figure 6.11 44](#_Toc165238538)

[Figure 6.12 45](#_Toc165238539)

[Figure 6.13 45](#_Toc165238540)

[Figure 6.14 46](#_Toc165238541)

[Figure 6.15 46](#_Toc165238542)

[Figure 6.16 47](#_Toc165238543)

[Figure 6.17 48](#_Toc165238544)

[Figure 6.18 48](#_Toc165238545)

[Figure 6.19 49](#_Toc165238546)

# INTRODUCTION

Our daily car rental platform, developed with ASP.NET Core Web API and Angular, is designed to enable users to quickly and reliably rent vehicles tailored to their needs. By utilizing modern web technologies, we have enhanced both the user experience and streamlined administrative processes. The platform is built on a RESTful architecture, offering a flexible and scalable structure. Users can filter vehicles by brand, model, price range, and rental duration, allowing them to find the desired car quickly. The secure online payment system ensures that users can safely complete their rental transactions using credit cards or other digital payment methods. Additionally, vehicle owners or administrators can easily upload and update photos of the vehicles, helping users to better understand the car they are renting and make informed decisions. The admin panel enables site administrators to efficiently manage vehicles, users, and reservations, significantly simplifying administrative tasks. The user-friendly interface developed with Angular allows easy navigation of the site and quick completion of tasks, thus maximizing user experience. With ASP.NET Core's robust security measures and high-performance, scalable infrastructure, we can effectively meet user demands. This platform provides a comprehensive and modern solution that allows users to easily find, securely rent, and enjoy a seamless rental experience. Our platform aims to stand out in the industry with its innovative approach, RESTful architecture, and user-centric features.

## The Aim of the Project

The primary aim of this project is to develop a modern, efficient, and user-friendly daily car rental platform utilizing ASP.NET Core Web API and Angular. This platform is designed to streamline the process of renting vehicles on a daily basis, catering to both users looking for convenient car rental solutions and administrators managing the rental operations. By leveraging the latest web technologies and a RESTful architecture, the platform aims to provide a seamless and intuitive rental experience, ensuring flexibility, scalability, and ease of use. The project seeks to address the current challenges in the car rental industry, such as inefficient booking processes and the need for dynamic vehicle management. Ultimately, this project aims to enhance user satisfaction and operational efficiency in the car rental business.

## Scientific Innovations and Contributions

This project introduces several scientific innovations and contributions to the field of web development and e-commerce:

1. Integration of RESTful Architecture:

The platform utilizes a RESTful architecture, which ensures a modular, scalable, and maintainable system. This architectural approach allows for seamless integration with other services and enhances the overall performance of the application.

1. Advanced Filtering and Search Capabilities:

The dynamic vehicle filtering feature allows users to search for cars based on various criteria such as brand, model, price range, and rental duration. This enhances the user experience by providing quick and relevant search results, thereby reducing the time and effort required to find the desired vehicle.

1. Virtual Payment System:

The platform includes a virtual payment system that simulates the payment process. This feature helps users understand how the payment would work in a real scenario, though it does not handle actual transactions. It serves as a prototype for integrating a secure payment gateway in the future.

1. Comprehensive Administrative Panel:

The development of a user-friendly administrative panel provides a powerful tool for managing vehicles, users, and reservations. This feature simplifies administrative tasks, allowing for efficient management of rental operations and improving overall business efficiency.

1. User-Centric Design:

The platform's front-end, developed using Angular, emphasizes a user-centric design approach. By focusing on usability and accessibility, the platform ensures that users of all technical levels can easily navigate and utilize the application, enhancing overall user satisfaction.

1. Scalability and Performance Optimization:

The project addresses scalability and performance optimization, ensuring that the platform can handle a growing number of users and transactions without compromising on speed or reliability. This is achieved through efficient coding practices, database optimization, and leveraging cloud-based services.

Through these innovations, the project not only provides a practical solution to the car rental industry but also contributes valuable insights and advancements to the broader field of web application development. The research and methodologies applied in this project can serve as a reference for future developments in similar applications, promoting best practices and encouraging further innovation.

# RENT A CAR PROJECT REVIEW

Technological developments, diversification of needs, increase in data /information flow for organizations due to reasons such as population growth and demands for processing, storage, frequent recall of these data have made it impossible to meet the requirements practically and economically by increasing the number of office employees only.

While the rapid expansion of computer use is transforming our age into the information age, all segments of society are greatly affected by these developments. Now, using information and obtaining new results has become one of the conditions of being an information society. The importance of information in the decision-making phase has started to direct organizations towards the creation of information systems.

It is aimed that the system offered in car rentals can be given in the best way, income and expenses can be monitored, expensive cars can be provided to the right customers (with customer score).

The customer can also make comparisons about the cars by accessing information about the offered vehicles, vehicle pictures, fees, and decide which one is more suitable for him.

## Rent a Car Website

Renting a car has an important place in the automobile industry today. In short, it is an advantageous financial event in which you provide yourself or your company with vehicle usage within the scope of mileage and monthly pay for a certain period of time (we have priced it depending on the day in our project). Today, all businesses, large or small, need passenger vehicles for functions such as sales, marketing and logistics, and they prefer to rent rather than buy cars in these companies. It has become a situation that even those who have a car prefer. For example, someone who is going out of town prefers to rent a car where he goes with other alternatives instead of his own car. The reasons for this are that a person does not want his car to do too many kilometers, we can list reasons such as going faster by plane to his destination. The main reasons for choosing a car rental site are the increasing car prices today, this is the beginning. As car prices have increased, people have started to prefer renting rather than buying a car. That's why I decided to do such a project, thinking that the intensity of the car rental sector will increase in the coming years.

## Studies carried out in the Car Rental Sector

A survey study was conducted for 379 customers who rent a car for a short term at Sabiha Gökçen Airport. The screening method, one of the quantitative research methods, was also used in the research. The research is limited to domestic and foreign passengers who rent a car from car rental companies located at Sabiha Gökçen international Airport in Istanbul. The research was conducted between Dec 30.05.2019 and Dec 15.06.2019. October et al. in the research. the service quality RENTQUAL scale was used in car rental developed by (2009). As a result of the research, it has been determined that the perceived quality perception of customers towards the services of car rental companies is high. As a result of the factor analysis, it was determined that the car rental service quality scale consists of two sub-dimensions, general service quality and car delivery service quality, and that general service quality and car rental service quality positively and moderately affect each other. In addition, differences have been found in the quality of vehicle delivery service between customers who rent a car for the first time and those who rent twice and more than four times. Dec. According to the findings obtained at the end of the research, recommendations for service quality have also been developed.

In line with the research conducted in my contribution to the project, in order to provide better service to customers, in order to help with the price issues that customers complain about, and when we look at other companies, most companies provide services under certain brands. Like Ford, Fiat, I'm aiming to get out of these brands. I have goals such as more different vehicle models, more technological vehicles, new vehicles.

# CAR RENTAL PROJECT ANALYSIS

My project was a car rental website. I used layered architecture when making our site. In the following sections, we will also discuss the layers of layered architecture, now if we briefly touch on it, we can say that it is the structure that allows us to develop our projects according to a certain standard and layout, increases the readability of the code, makes our projects more uncluttered and makes error management easier. Layered architectures consist of 3 layers;

Data Access Layer: Only database operations are performed at this layer. The task of this layer is to add, delete, update and withdraw data from the database. There are no other operations other than these operations in this layer.

Business Layer: We write our workloads on this layer. First of all, I should say that this layer is the layer that will process the data taken into the project by Data Access. We do not use the Data Access layer directly in our applications. By putting the Business layer together, we make sure that Business does it for us. Dec. The data from the user first goes to the Business layer and is processed from there and transferred to the Data Access layer. In this layer, we also specify who will have access to this data. For example, there is the IT and Accounting department. We want the IT department to make additions to the database, but if we want the Accounting department to only pull data, we do it at the Business Layer.

Presentation Layer: This layer is the layer where the interaction with the user is performed. This can also be a Windows form, it can be on the web, or it can be in a console application. The main purpose here is to show the data to the user and transmit the data from the user to Data Access via the Business Layer.

If we decipher the rest full architecture we use in our project, it is a service structure that allows quick and easy communication between the client and the server. So it does not store status information. Let's open it up a little more; In Rest standards, extra header information is not stored in the data moved between the client and the server, there are no details belonging to the client, this information is not moved between the client and the server. Decryption of Decryption is not possible. If we Compare Rest and Soap, Theoretically Soap is a protocol, Rest is a set of rules. Although it is wrong to compare these two terms directly, when we take into account the research conducted, we thought that rest was more efficient and we included it in our project. If we talk about rest full http methods;

* Get: Allows us to find out what the source on the web server is.
* Post: We use it so that we can create a new resource on the server.
* Put: We use it to update a resource located on the server.
* Delete: We use it to delete, delete a resource located on the server.

## The Technologies Used on Our Site

As a result of our research on the technologies we use on our site, we have used the most preferred, useful and most widely used programming languages today with their IDEs. For example, we have included Microsoft's ides and programming languages in the backend, frontend, database section. In this section, we will briefly touch on the technologies we use, and in the following sections, we will present these technologies in detail and with pictures. First of all, we used the solid principle when making our project, it is a principle that brings together the rules that a software developer must follow when making flexible and open to development object oriented programming. Dec. Solid consists of 5 different principles and takes the initial of each of them.

According to the Single Responsibility Principle, each method and class has a single task and responsibility.

According to the Open Closed Principle, every class should be open to development, but not closed to change.

According to the Liskov Substitution Principle, objects created from subclasses must exhibit the same behavior when they are replaced by objects of upper classes. In other words, derived classes must use all the properties of derived classes.

According to the Interface Segregation principle, every interface should have a specific purpose. Instead of using a single interface covering all methods, several interfaces are preferred, each of which serves separate method groups.

According to the Dependency Inversion Principle, High-Level classes should not be connected to Low-Level classes, the relationship should be provided using abstraction or interface, Abstraction should not depend on details, on the contrary, details should depend on abstraction.

## Backend Side

We made the backend side of our site with visual studio in C# language with asp.net using web api and we used the layered architecture that we mentioned and explained on the previous page. If we mention briefly:

Visual Studio: Visual Studio is an IDE, an integrated development environment where you can make programs, applications or websites using many programming languages. It is used to develop computer programs, websites, web applications, web services and mobile applications for Microsoft Windows.

C#: C#, the next generation programming language, brings innovations in programming. C# is designed for server and embedded systems. Developed by Microsoft, C# was developed for use in the .Net environment. Anders Hejlberg pioneered the design of the language. This programming language, which has developed over time, is now used in most places, whether desktop or web applications.

Asp.net: ASP.NET is a framework for building web and desktop applications. ASP software comes in the form of forms or web applications and these programs are developed in Visual Studio, Microsoft's developer platform. ASP software runs on a Windows desktop computer or web hosting server in corporate or small office environments.

Web APIs: Web APIs are defined interfaces where interactions take place between the organization and applications. An API approach is an architectural approach that provides a programmable interface to provide services that can be used by different applications serving different types of consumer communities. In the context of web development, an API is a description of a structure of request messages, typically Hypertext Transfer Protocol (HTTP), and response messages, typically in Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format.

## Frontend Side

We used Visual studio code on the frontend and angular library on the node.js platform.

Visual Studio Code: is a fast, lightweight tool that can run on Microsoft, Linux and Mac operating systems. The product, which appears as a simple text editor when installed, supports many programming languages such as Node JS, Ruby, Python, C/C++, C#, Javascript thanks to plugins. In other words, instead of incorporating everything like classic Visual Studio products, a logical move was made by starting with a core structure and adding the necessary parts to the product as plugins. A market environment where plugins can be downloaded has been created.

Node.js: Node.js is basically javascript running on the server side. Since it is based on javascript, it has a very dynamic and fast structure. We can also say that it is Google's open source javascript runtime environment.

Angular: is a Javascript-based framework used to create web, mobile and desktop applications. Angular applications include the use of Typescript, a language built on top of Javascript, together with Html, Css.

## Database and Test Environment

In this chapter, I used SSMS to store my SQL data and Postmen as a test environment because I think it is faster.

SQL Server Management Studio: Sql server management is a database editor developed by Microsoft. With the help of this editor, we can perform many different operations (such as data editing, reporting and analysis) on our data using T-SQL commands.

With these programs, we create an MSSQL type database, design the database, and make the necessary settings. In short, we can say that it is the platform where we design the database.

SQL SQL (Structured Query Language) is the most popular query language used to retrieve data from relational database management systems, to edit the data in the database or to enter data into the system. SQL was originally designed to support object-relational database management systems. But beyond this purpose, it has many features that are specified by ANSI and ISO standards.

Post men: It is a tool used to share, test, document and monitor APIs without the need for long code, which we can call "rest client".

# CAR RENTAL AND THE USE OF COMPUTER SYSTEMS IN CAR RENTAL

Car leasing is giving the lessor the rights to use the vehicle during the lease period. In the legal transaction based on the contract between the lessor and the lessee, the ownership remains with the leasing company, while the right to use the property belongs to the lessee. With the increasing importance of the tourism sector in today's world, car rental companies are gradually developing due to the transportation service and other benefits they offer. The increasing needs and expectations of customers for service quality in the car rental sector indicate that competition will intensify in the coming years. We can divide the car rental sector into two main classes. It is the daily car rental system that we apply on our own site. With this system, if the user (customer) wants to rent the vehicle for how many days, pricing is made according to the number of days.

## Project Layers

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 4.1

As we can see in our picture and mentioned in the sections above, we used layered architecture in our project. I have already explained the layered architecture and now I will explain the layers I use.

Business layer is our business layer. It is the layer that processes or checks the information coming from the presentation layer according to the necessary conditions and sends it to the database using the methods provided by the data layer. There are 6 folders named Abstract, BusinessAspects, Concrete, Constans, DependecyResolves, ValidationRules in this layer.

Core layer is a universal layer, we can call it a framework layer. It can be used in every project to be developed, it is very useful because of its naming rules and creation order. There are 7 folders in our Core layer: DataAccess, Entities, Utilities, Aspects, CrossCuttingConcerns, DependencyResolves and Estansions.

DataAccess layer is the data access layer. It is a computer program layer that provides simple access to data stored in a consistent repository in our Computer Software. We have 2 folders in our Data Access layer, Abstract and Concrete.

Entities layer is the layer we create for our database objects, we have 2 folders in our Entities layer, Abstract and Concrete.

We have defined our web api interface above and if we talk about its content, we have 2 folders, Properties and Controllers.

## Diagrams and Tables

metin, diyagram, dikdörtgen, plan içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 4.2

Our diagram above is our relational database, relational databases usually have multiple tables and these tables are associated with each other according to certain fields. We can say that they are databases where multiple tables are kept in a logical relationship. As can be seen in our diagram, we have associated our tables according to their subjects.

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 4.3

We see our car table in which we show the number, name, model year, daily rental price, feature and score of our car. It also shows that we associate our car table with brandId (brand number) and colorId (color number).

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 4.4

In our car rental table, we include the rental number, the start date of the rental and the end date of the rental. This shows that we associate our rental table with carId (car number) and customerId (customer number).

metin, ekran görüntüsü, sayı, numara, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 4.5

If we talk about the contents of our user table, userId, name, surname, email, password and status information (is the user active) welcome us.

# SOFTWARE COMPONENTS

We used n-tier solid architecture structure in our project and we have mentioned solid architecture in the sections above. Using EntityFramework, we made our car rental site, which works with the Wpf interface, where we do our Crud operations, register, login, provide security as a token with Jwt techniques, caching, validation, transaction, performance operations using the aspects we created with the Autofac package. In our project, data sources can be easily changed, new objects can be added, and the operations we will do can be done with continuous additions without breaking the old codes. Our project has become a sustainable project.

Entity Framework: It is a framework developed by Microsoft that provides an ORM (Object Relational Mapping) opportunity by eliminating the strict sql queries of software developers. We can say that this is a bridge that allows relational database management systems to be intervened through objects instead of direct intervention.

Crud: It is the place where we do Create (Add), Read (Read), Update (Update), Delete (Delete) operations that we use with Entity Framework.

Jwt: Json Web Tokens (JWT) are used to enable secure exchange between parties as Json objects. Being digitally signed makes JWT reliable and verifiable. In terms of structure, we can say that it consists of three parts: signature-based, encryption-based, header-based. When the user logs into the application, a token is given to the user. Since the token is also a credential, it should not contain any confidential data and should be well protected. When the user wants to access somewhere in the application (this can be a category or an API), the header section of the user request contains "Authorization: Bearer <user's token>". If the token sent is authorized for the requested operation, the operation is performed.

Caching: Cache is a high-speed data storage layer that stores a temporary subset of data. Caching enables efficient reuse of previously retrieved or computed data. The caching method results in higher performance when the relevant data is requested in a subsequent process, as the data is accessed from the primary storage location. Data in a cache is usually stored in hardware such as RAM and may require a connection through software to access the data. The purpose of caching is to improve data access performance by minimizing the need to access the slower storage layer below. If we talk about the advantages of caching, we can say that it increases application performance, reduces database cost, reduces the load on the backend, etc.

Validation: The purpose of the validation control is to check whether the data entered by users in Web applications meets the desired conditions before it is saved in the database. These conditions may be that the field where data is entered is not left blank, the data is in the desired format, the data is greater or less than any value, etc.

Transaction: We can say that it is a process that considers one or more Sql expressions as a whole and, if successful, permanently applies the desired change to the sql expressions or, in case of error, preserves the data in the same way without making changes.

Autofac We will talk about asutofac in more detail in our installation packages section.

WPF: WPF is a visual design environment offered to C# developers on Visual Studio. With this tool, users can develop applications that they can intervene in every stage visually. At this point, we come across XAML (extensible application markup language). Let's briefly answer the question of what XAML (zemıl) is by opening a short parenthesis here. XAML is actually a design language with spelling rules similar to XML language, which web developers are very familiar with. We can think of this language as CSS for HTML or as the form interface with the toolbox in Windows Form Applications.

## Backend

In our project, the backend side of our project starts after the frontend part of our site, which is the face that invites the user, is finished, this is the stage where the systematic, software part of the site is produced. The backend part consists of a server, an application and a database. With these parts being smooth, we ensure that our backend side works. In this part of our project, we used Mssql in the database architecture with Asp.net. We used Nuget in this part of our project. We will talk about Nuget in our installation packages title. Let's talk about the folders in the layers we explained in the above sections and what those folders do;

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.1

We created our Concrete folder to keep the properties of concrete objects coming from the database. We created our Dtos folder to use the relationship properties of objects that are related to each other in the database together.

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.2

In the folders inside this layer, the management of all the services with ready infrastructure was written. We can say that this is the layer where we can add our constantly changing business codes without changing the infrastructure. In this project where continuity is maintained, the fact that many changes are made only here facilitates management and continuous development.

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.3

Aspects folder is the folder where we prepare the Autofac attribute infrastructure of caching, validation, transaction, perfonmance operations. In ClassCuttingConcer folder, we generalize validation and cache management within the project so that it can work dynamically in the vertical layer. We make service configurations in the DependecyResolvers folder. In DataAccess folder we configure all crud operations and databases generically. In Extensions we have made extensions that make management easier for jwt. Entities folder has a class database table that implements IEntity in itself. In Utilities, we have facilitated the management of business method rules, document insertion operations, we have prepared the infrastructure for aspects to intervene, we have configured error management by setting up the result structure, and security has been configured with jwt and hashing techniques.

metin, ekran görüntüsü, menü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.4

We have folders that contain Abstract interfaces and Concrete classes. We perform Crud operations by inheriting from the core layer. We prepare the infrastructure for future business codes here. The databases that the objects will use for data transfers and the connections of the entities are configured in this layer.

metin, ekran görüntüsü, ekran, dizüstü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Api stands for application development interface and it is the interface where we define certain rules in order to open the service or data we have to the outside world and make it available to other applications-platforms.

The purpose of using Web Api is that in recent years, access to the internet has been provided by many different platforms, so that only websites cannot meet the needs of users. These platforms are computers, tablets and especially mobile phones that exhibit different capabilities from each other. Consequently, application developers are turning to developing APIs that are independent of the application in order to reach users in a platform-independent way. These APIs provide services using data types such as XML or JSON that all platforms and applications can read and make sense of, and in this way, they can serve end users in line with the purposes of the developers by freeing themselves from all dependencies. Since it is open source, a technology that is constantly being developed and features are being added, and as a result of our research, we thought that it is the best choice for developing restful services based on http. If we talk about the Program.cs file we received to the user is the code page that the user reads before opening the program when the program receives the run command. In other words, it is the page with the first commands processed before the program is opened.

ekran görüntüsü, metin, yazılım, bilgisayar içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.5

In our Entities layer, we see our car.cs source file in our concrete folder, where we are greeted by the car number, brand, color, car name, model year, daily rate, car information, car's point status.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.6

In our DataAccess layer, we see our RentAcarContext.cs source file in our EntityFramework folder in our Concrete folder. Here we are connecting our sql server database.

yazılım, multimedya yazılımı, metin, grafik yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.7

What we see here is the source file named customer.cs of our concrete folder in our entities layer. Here we see the customer number, user number, company name and findex score.

ekran görüntüsü, yazılım, metin, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.8

Here we see our source file named entityrepository.cs in the dataacces folder of our core layer. A generic structure welcomes us here. If we briefly talk about generic structures; We can say that it is a structure that allows the interface, class, method or parameters (arguments) we design to work not for a specific type but for every type that fits a template structure.

## Frontend

It's nice to have user data in the database as the back-end part, but on the other hand, what makes your website look good? This is where the frontend comes in. It creates a good look and feel for every page visited. In other words, it creates the colors, backgrounds and all other styles for every page that you interact with directly and can see with your eyes. It reflects the brand identity and provides an emotional experience to the website users. Now let's talk about the angular library that we use in the frontend part;

E2e: end to end folder. It is the necessary folder for unit tests and other tests.

Node\_modules: It is the folder with the packages we install with npm.

Src: This is the folder where the application runs. The components, assets files and config files etc. that we will add later will be located in this folder.

App: It is located under the src folder and holds the code components to be added to the application. (Component, Pipe, Service, quard, Models, interceptor) We add all the components and most of the scripts we will add for the project we will develop under this folder.

Now let's talk about our folders in our Frontend section:

Models folder: There is a models folder that we created in our project. In this folder, it is the part where we keep our classes corresponding to our tables in the Backend (our classes in the entities layer) and in our database and the properties of these classes. Since we have many tables in our project, we will take a base class and share our codes over this base class.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.9

As it can be seen here, we have created a car.ts. Then we named the columns corresponding to our table just as we keep them in the database. Here, the naming must be kept exactly as in the database, otherwise we may encounter errors when we want to access the data. As we do on the backend side, when we call our data, a data (data comes in an array), a success information and a message information comes. Instead of writing the code to call the data one by one for all our tables, we wanted to keep this data in a generic way, which caused us to get rid of code repetition. In the following 2 pictures, we show how we created this structure and how we created it with code.

metin, ekran görüntüsü, multimedya yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, multimedya yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.10

Figure 5.11

Service: In our Angular projects, components should not directly receive or save data. For example, it should focus on receiving data from a service and transferring it to the user or receiving data from the user and sending it to the service, and while developing Angular applications, we may encounter scenarios where we will use these services in more than one component when necessary. In order to create a service, open in integrated terminal on the folder you want to create, then come to the terminal and type ng g service service servicename. There is a services folder in our project. We created our services under this folder. Let's show the operations we do over a base service by sharing our codes. Here we have taken carService as base.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.12

Here we can say that this is the part where we call http requests from the webapi by the backend. In order to make an http request, the httpclient import on line 1 in the imports allows us to access the http from our api. Since we will use it globally after the definition process, we need to define it as a variable in the constructor. After performing this operation, we had many get and post operations that we wrote in the webapp, here I will explain what we did in getCars in line 20, we have performed the same operations in our other methods. The getCars():observable<ListResponse Model<Car>> we created here is our code that will allow us to access the car object we created in the model folder and the data, success and message information it returns. When we go into the method (referring to the code in line 21), here we use the get property of httpClinet to send our webapide address in the same way as we have stored it. Here, we must send the address in the same way as we created it in the webapp on the backend side, otherwise you will not be able to access the data. The same operations have been performed for all other methods.

Component: Component is a class that serves an html output. We can also think of components as blocks. For example, let's say we have a page with a list of products or a list of customers. On these pages, we need to pull the relevant data (data) with the component. We need to present the data we extract to the html of the relevant component. The html changes static or dynamic objects according to the data in the component according to the information in this component. Methods are found inside our components. If we want to summarize briefly, components are the part where we manage our data (component.ts) and write our codes that will display this data in the browser (component.html). In order to create a component, we create it by saying open in integrated terminal in the folder we want to create and then ng g component component component name in the terminal. Now let's share our codes by taking a basic component from the components we created in our own project. I will explain the components in the car folder in our project in a basic sense.

metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.13

In the picture we can see our folder for update-delete and vehicle details for the car object. As the name suggests, for example, in car-add-component.ts in the car-add folder, we manage our data in this component by using carservice (because we need to use it to access the required api url.) In car-add-copmenent.html in the same folder, we show that we can be given to the user using html and bootstrap in the html section using the data in car-add-component.ts. In the picture I shared below, we will share our car-component.ts and car-component.html codes as an example and we will touch on the important places on it.

metin, elektronik donanım, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.14

Here you see our codes in car-component.ts. For example, the carDto just below the Export class is necessary for us to access the properties of our class that we keep in the carDto in the previous models folder. We will call our data using the carDto we defined here using the foreach loop in the html section. If we return to component.ts again, we need to inject the activateRoute service in the constructor in order to use the params feature seen in the code written in ngOnInıt, we can send as many parameters as we want with the params feature, in the code seen, if the colorName parameter is called, we say that if none of the parameters in the ifs are present in this. getCarDetails, we have said that the code we call getCarDetails should bring the detail page about the vehicles, you will see their control just below the picture, we realize this event using the carService.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.15

Our codes that appear here are our html and bootstrap codes. The foreach code we use to access the features in the carDto I mentioned above is as written in lines 57 and 58 in the picture. In Angular, in order to access the data in the html section, we can say {{{data to be accessed}} open, close, close, open and close the fancy brackets and enter the name and property of the class you want to reach in the middle, for example, as in line 79.

Pipe: Pipes entered our lives with the name filter in angular js version. Since Angular 2 version, it is called Pipe. We can define pipes as filtering or transforming. We can transform and reuse any data on the template without writing code in the component. Moreover, with Pipes we can do this with high performance and immutable. In addition, Pipes allow data to be processed while a workflow is in progress. There are ready-made pipes that Angus offers us, let's share a few of them.

* DatePipe: formats a date value according to locale rules.
* UpperCasePipe: Converts text to all uppercase.
* LowerCasePipe: Converts text completely to lower case.
* CurrencyPipe: Converts the number to a currency string formatted according to locale rules.
* DecimalPipe: Converts a number to a string with decimal points formatted according to locale rules.
* PercentPipe: Converts a number to a percentage string formatted according to locale rules.

Pipes are used with | in Html. To create a pipe, we create it with the name se terminal da ng g pipe pipe. In our project, we created a pipe folder and kept our pipes here. For example, I will share our car-filter-pipe codes here.

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.16

In the codes that appear here, for the vehicle name, color name and brand name, we first convert our names to lower case using the toLocaleLowerCase property, and then we perform the filtering process according to the filterText sent.

Interceptors folder: Interceptors ensure that the requests sent by the client to the server are modified and reach the server. Thanks to interceptors, we intercept requests and this provides us with practicality at many points. Token Authentication is realized by passing the token obtained from the web api used by passing parameters to all requests via Intercaptor. Our code image is below.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.17

Guards folder: We want to prevent the user from accessing some pages in our Angular application without logging into our application. In this way, it is necessary to define a route guard to ensure page access, that is, security in the routes we create. Now let's share with you our guard code in our project and explain our codes.

metin, elektronik donanım, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.18

First of all, as can be seen in the code above, we define a class that we will use as a service that implements from the canActivate interface. In this class, we define our canActivate function to check whether the user is logged in or not, that is, when the relevant route is called. The login process is done according to the return value of the following isAuthenticated () function in Authservice. If the login process is successful, the token value is received and we make the page accessible. Let's show its usage in app-routing. module.ts in the following picture.

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

As it can be seen in the picture, canActive: [LoginGuard] as shown in the image.

* -Assets: It is located under the src folder and is the folder where files such as image, file, js will be kept.
* Environments: This file contains two files as follows. With these files you can set the configurations according to development and prod environments.

-environment.prod.ts

- environment.ts

* Index.html: We mentioned in the introduction that Angular SPA is a front-end framework where we create applications. For this reason, the application must have an input file and all rendering operations must take place in this file. That file is our index.html file. When you look at index.html, you will see an html tag called <app-root>. This component is actually our root component. The component that will return all routing and rendering operations. We will get into the details of this later.
* Main.ts: This is the start file of the application.
* Package.json: This file stores information about libraries added to the project. That is, when you run "npm install" from the terminal, it holds the package information to be installed into node\_modules. If we take a look at the file, you will see the list of packages it depends on. You may think "I already installed these packages, why should I install them again with npm install". But in enterprise software, we develop projects with teams and naturally we use version control and management systems like git, team foundation server, svn. We make improvements in your local environment and send them to the main repository. Since folders such as node\_module have high sizes and are standard, we ignore local and throw the code to the main repository. The person who wants to develop code for another issue or who pulls this code for the first time from the main repository to the local environment will need these packages to run the project. With the "npm install" command, you can download the dependencies whose references are already in the package.json file and quickly make the project ready for development and debugging.
* Angular.json: This is the component that allows us to add and use application settings and third-party installed javascript, css or our own custom javascript or css files.

## SQL Database

We mentioned that we used ssms id in our Sql database section. However, we preferred Mssql database in our project. MSSQL, or Microsoft SQL Server with its long name, is basically a system in which data is stored. In more detailed terms, it is an enterprise-wide relational database management system that allows data to be stored and accessed by multiple users at the same time. We can store a lot of data here, including the texts in a website, comments on articles, users' information. Now let's share our pictures of our database;

metin, diyagram, dikdörtgen, plan içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.19

We also include our diagram that we mentioned in the sections above. Here we have mentioned that we are using relational database and what relational database does, in addition to these, we can say that the Id's of our tables are Primary Key and that we only associate them with each other.

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.20

We have also included our car table here, while creating our table, we come to our databasem and enter DesignRentACar in it, then right click on the table option and call new table, then we create our car table. In our car table we see carid, brandid, colorid, car name, brand year, daily rate, car information and scoring.

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.21

Here we see our brand table and in our brand table we are greeted by brandid and brandname. As we can see in the picture above, when creating our table, we can either create it as a design or we can create our tables by coding, we used both of these options in our project.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.22

We see the user table in the section above. We log in to our project with user email and password. In this table, we are greeted with userid, name, surname, email and password.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.23

Here we are greeted by our payment page. Our payment page contains paymentId, rentalid, total amount and amount paid.

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.24

This table contains our car images. Here you will find carid, carid, image path (address) and upload date. We can add our images from here or from our project after running our project, that is, in 2 ways.

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.25

Here is our table with our car colors. Only colorid and color name are available here.

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.26

Here we are greeted by our customer table. Our customer table contains customerid, userid, company name and score status. Here we have associated customerid and userid with each other.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.27

Finally, we see our rental table. Here we are called rentalid, carid, customerid, rental period, which is the period from the start of the rental until the last day of the rental. After entering the pick-up and drop-off times, the program calculates the duration and fee for us. We have associated rentalid, carid and customerid here.

# INSTALLATION PACKAGES

* Nuget: It provides the management of external packages used when developing software with the .NET platform. The most important feature of the .NET platform is that it facilitates development with hundreds of ready-made classes. Now let's talk about our existing packages on our backend side.
* Autofac: It is an IOC container developed for the .Net Framework. In other words, it is a software design principle and in simple terms, it is responsible for the life cycle of objects throughout the application. If we explain the IOC container; It is the management of the life cycle of the objects to be created. In other words, it is the production of the relevant object instance on our behalf in a singelton way for each request under the specified conditions. This brings us convenience and a visible simplicity in the code.
* Autofac.Entensions.DependencyInjection: Adding dependencies in the .Net Framework is the Autofac implementation of our units in the abstraction.
* Autofac.Extras.DynamicProxy: Autofac extension we use to enable AOP with Caste.
* FluentValidation: Validation is a validation library that uses a fluent interface to create strictly defined validation rules.
* Microsoft.AspNetCore.Http: This is the package that contains the default http feature implementations.
* Microsoft.AspNetCore.Http.Features: Here we have our core http feature interface definitions.
* Microsoft.AsNetCore.Http.Abstractions: This is the core http object model for http requests and responses. It is also a common extension method for registering middleware.
* NetStandard.Library: is a set of NSL api prescribed for use and support
* Microsoft.EntityFrameworkCore.SqlServer: Mes is our Microsoft Sql Server database provider for Entity framework core.
* Microsoft.IdentityModel.Tokens: Contains types that provide support for ST encryption operations with the MIT package. Operations such as signing, verifying signatures, encryption.
* Newtonsoft.Json: It is a popular high-performance json framework in the project we are using.
* Microsoft.IdentityModel.Tokens.Jwt: This is our MITJ package with types that provide support for the creation, serialization and validation of json web tokens.

Now let's move on to our packages in our fronted section;

We have mentioned Angular in the sections above. In order to use Angular, we need to install some packages in VSC.

* Node: we install it to use package management. It will act as a server. So you need to install node.

Let's move on to our packages installed in Angular;

1. npm install -g @angular/cli this is our npm package that will allow us to install the necessary setup to write an angular project in code vs install.
2. ng new project name = to create a project to create, compile and deliver a basic Angular project.
3. ng serve - open: this process is not a package installation. It is the command that must be written to stand up the project, publish it (show it in the browser).
4. We installed the bootstrap package by saying 4-npm install bootstrap. This installation is the installation that will allow bootstrap css to work in html codes.
5. We install the query package by saying 5-npm install jquery. This package is a necessary package for the dropdown to work in the bootstrap we use.
6. npm install ngx-toastr this package installation allows us to provide information in the form of a message box in a certain part of the screen. Toastr gives a meaningful message to the user so that he/she can understand the status of his/her action.
7. npm install @angular/animations : since toastr uses angular's animation library, we need to add it for our toastr package to work properly and for us to use it.

## Test Environment and Post Men

We have talked about Post Men in the sections above. What it does and what it does, now we will explain with examples how we use postmen in our project and what we do with it. We can say that the reasons why we prefer Postmen are that it is more useful and there are more informative videos about postmen on the internet, etc.

metin, ekran görüntüsü, yazılım, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.1

Here in our test image in the postman test environment, we are listing the cars in our project, you will notice in the picture that after the first vehicle's properties are finished under the data, it will separate it with a comma and show the new data, but we were able to picture a single car due to the screen picture.

metin, ekran görüntüsü, yazılım, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.2

Here we list the cars in our test image in the postman test environment, the difference of this listing from the listing in car-getall is that we do not make a detailed listing, that is, we bring the picture of the car, we show the brand name with the brand name instead of showing the brand name with id, the same applies to the color.

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.3

Here in our postman image we list the properties of the car according to the id value of the car. In the test image here, we brought the car with vehicle id 1.

metin, ekran görüntüsü, yazılım, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.4

Here we are listing the vehicles with a brand id of 1 in our picture in the postman test environment. Here, since we have 1 ford branded vehicle, a single data came in our test, but if we had another ford branded vehicle, the information of that vehicle would have come because its brand id is the same.

metin, ekran görüntüsü, yazılım, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.5

Here in our postman image we list the properties of the car according to the id value of the car. In the test image here, we brought the car with vehicle id 1.

metin, ekran görüntüsü, yazılım, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.6

Here we are listing the vehicles with a brand id of 1 in our picture in the postman test environment. Here, since we have 1 ford branded vehicle, a single data came in our test, but if we had another ford branded vehicle, the information of that vehicle would have come because its brand id is the same.

metin, ekran görüntüsü, yazılım, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.7

Here we list the vehicles with color id 1 in our test image in the postman test environment. As you can see in the picture, since there are more than one vehicle with color id 1, we list all vehicles with id value 1.

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.8

Here, the picture in the postman test environment is our test picture where we list our users. Since we keep the encryption process in the database using the hash algorithm, as you can notice from here, we cannot see the open version of the password, how it looks in the database looks the same in the test environment.

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.9

Here we are pulling our data about the colors in our database in our image in the postman test environment.

metin, ekran görüntüsü, yazılım, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.10

In this postman test environment, we are pulling our data related to the brands in our database.

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.11

Here, finally, we have shown our data about car rental in the picture in the postman test environment. We can see the rental date and delivery date, which vehicle was rented and which customer rented it.

## Project Outputs

In this section, we will share screen shots of our project.

metin, yazılım, web sayfası, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.12

This is the screen where we register. After entering Name, Surname, E-mail, Password, registration is successful.

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.13

This is our login screen. After entering our e-mail and password, our process takes place.

metin, kara taşıtı, taşıt, araç, tekerlek içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.14

This is our admin page. Here Brand, color etc. It is the section where we add features.

metin, kara taşıtı, tekerlek, taşıt, araç içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.15

This is the section with member login. Users who log in can do brand color filtering here. The car he wants can move to the next page.

metin, taşıt, araç, ekran görüntüsü, araba içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.16

This page is the next stage after the customer selects the vehicle. Here, if the findeks score is sufficient and the car has not been rented on the dates of pick-up, he/she can proceed to the payment page.

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.17

We are making payment on this page. The customer can make the payment by entering their card information.

metin, araba, kara taşıtı, taşıt, araç içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.18

Here we see our home page, the screen that greets us when we first enter the project.

metin, kara taşıtı, araba, tekerlek içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.19

Here we see the administration panel. As we mentioned on our admin page, we can say that we briefly manage our website here. We do operations such as adding and deleting data. Inside the administration panel, Users, Customers, Rentals and adding new vehicles tabs welcome us.

# RESULTS AND RECOMMENDATIONS

## Result

1. The installations required to realize the project were successfully created.
2. Test phases were successfully realized.
3. The user interface of our project was designed successfully.
4. The project was successfully finalized in line with the desired goal.
5. Our car rental site has been prepared in a simple and elegant design considering the use of a customer. Care was taken not to be over design. Od. means (over design).
6. In our project, coding techniques were realized in accordance with Solid principles.

## Recommendations

* The design part of our project can be improved.
* Car rental payment parts are simulations. A real payment service is not used.
* My project is designed for a web platform. It can be developed as an application for mobile devices
* The above-mentioned suggestions could not be made due to time constraints.

REFERENCES

|  |  |
| --- | --- |
| [1] | <https://medium.com/codable/> |
| [2] | <https://blog.burakkutbay.com/postman-nedir.html/> |
| [3] | <https://docs.microsoft.com/en-us/ef/core/get-started/overview/first-app?tabs=netcore-cli> |
| [4] | <https://www.w3schools.com/js/js_api_intro.asp> |
| [5] | <https://angular.io/> |
| [6] | <https://medium.com/bili%C5%9Fim-hareketi/angular-front-end-ve-asp-net-core-web-api-back-end-ile-kimlik-do%C4%9Frulama-a0af1a49d42> |
| [7] | <https://getbootstrap.com/docs/5.0/components/navbar/> |
| [8] | <https://docs.microsoft.com/tr-tr/dotnet/api/system.io.filestream?view=net-5.0> |
|  |  |

CURRICULUM VITAE

**Ilayda**, **Paslı**

**Education :**

*Bachelors* : Piri Reis University

2017-2020 Management Information Systems

*High School*: Uğur Anatolian High School

**Work Experience / Internship: Vignold Turkey**

**Professional skill: .Net , Javascript, CSS, SQL, Java,**

**Email: pasliilayda@gmail.com**

Github Link: https://github.com/ilaydapasli/RentACARProject