Oct 28th, 8:00 AM - Oct 29th, 6:00 PM

Development of the application for cinema management with .net technology

Enise Hasanaj
University for Business and Technology - UBT, eh47364@ubt-uni.net

Lavdim Beqiri
University for Business and Technology - UBT, lavdim.beqiri@ubt-uni.net

Development of the application for cinema management with .net technology

Enise Hasanaj[1*], and Lavdim Beqiri[2]

[1]Kosovar, Peja, Kosova
eh47364@ubt-uni.net

[2] Kosovar, Peja, Kosova
lavdim.beqiri@ubt-uni.net

Abstract. The research discusses an internet platform dedicated to cinema management. The platform is built upon two key components: the user-facing part and the backend operations. The visual part, also known as the Frontend in the programming world, is developed using technologies such as HTML, CSS, and JavaScript, offering a simple and user-friendly interface. On the other hand, the server or backend where modifications and updates are made is built using ASP MVC Core and the C# programming language. For data management, SQL Server is used, which facilitates the staff's work in storing and accessing cinema-related information.
Overall, this system allows the administration to have complete control over all cinema functions. Responsibilities are divided among the main administrator, manager, and receptionist, making interaction with customers easier and more efficient. The platform primarily focuses on managing movie titles, screening rooms, movie schedules, reservations, ticket sales, and user management, providing a comprehensive solution for all these needs.
This online interface offers easy and quick access for both staff and customers to obtain the desired information. Another important advantage is that it helps minimize errors that can occur during various processes while significantly reducing operational costs. This platform is designed to be accessible by all team members, from the super-administrator who configures all system parameters to the customers who want to watch a movie, making the cinema available online 24/7.

Keywords: Internet platform, User-facing, Frontend, HTML, CSS, JavaScript, Backend operations, ASP MVC Core, C# programming language, Data management, SQL Server, Administration, Online interface.

1. Introduction

The cinema management system is an online platform built upon Microsoft technology, known as ASP.NET MVC Core and is coded in the C# language. This platform is adaptable and compatible with various browsers and different operating systems. This digital platform provides a convenient way to oversee and coordinate all actions and operations within a cinema, simplifying and automating many daily tasks, contributing to reducing the amount of work required to run the cinema. ((ZHANG, 2006)) It is a valuable tool for having comprehensive oversight and a unified intervention in all actions and activities taking place in the cinema environment. The platform for managing cinema operations is built upon a range of advanced technologies and tools. Here are the technologies and programming languages used to construct and optimize this system:

• Programming language C#
• Development framework ASP.NET Core MVC
• SQL database management system
• Object-relational mapping framework, Entity Framework
• Interface development languages, such as HTML, CSS, and JavaScript
• The library used for JavaScript manipulation is jQuery

• The stylistic and responsive framework, Bootstrap, for user interaction interfaces.

To effectively implement and manage these technologies, specific tools provided by Microsoft were used, which are:

• Development environment Visual Studio 2019
• Environment for managing and interacting with the database: Microsoft SQL Server Management Studio 18.

2. Analysis of the application

The application's structure is built upon a three-tier model, where each tier has its specific functions and responsibilities, thus simplifying and segregating responsibilities into different logical and physical layers. This structural model is well-known and frequently used in software development, especially in those

following the client-server model. This architecture aids in performance optimization by separating the application's logic in a way that each layer focuses solely on what it does best, and this segregation facilitates faster and more robust application development.

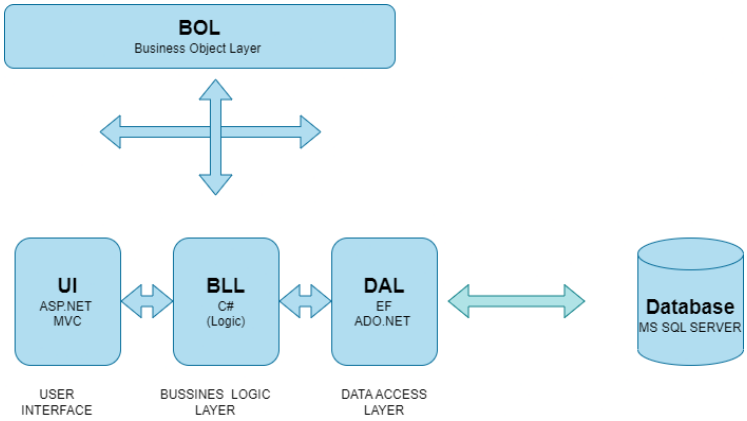

Fig 1. Application architecture

2.1 What is the 3-tier architecture?

The three-tier architecture is a well-structured and organized approach to software development, which is divided into three fundamental levels encompassing different aspects of the application. This division includes: the user interface level, responsible for displaying and user interaction with the application; the application logic level, where data processing and business logic occur; and finally, the data storage level, dedicated to storing, managing, and protecting the application's data. This approach offers a clear segregation of responsibilities and facilitates the management and expansion of the application in the future. (Cloud, 2001) Among the most evident advantages of the three-tier model is the ability to treat each level as an independent entity within the application's infrastructure. This separation consists of:

- The user interface level, which directly relates to how users view and interact with the software.
- The application logic level, where fundamental operations and business decision-making occur.
- And the data level, responsible for storing and managing information. (Hu, 2003)

With this structure, each layer can be treated as an independent project. This means that one development team might work on the user interface while another could focus on the application logic and another on data storage. Furthermore, changes or improvements to one layer can be made without significantly affecting the other layers, allowing for quick adaptation to changing needs or new technologies. This provides flexibility in development and maintenance of the software, making updates, expansions, or necessary changes without disruption or consequence to other parts of the application.

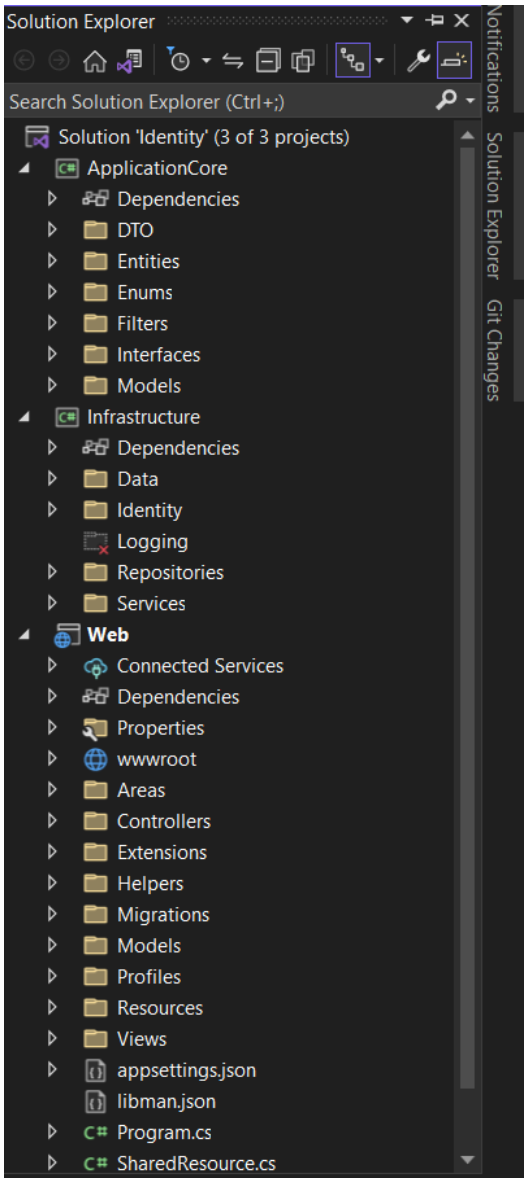

Fig 2. Project structure

The figure above illustrates the construction and organization of the project. This project is divided and structured into three main components:

Application Core - This is the heart of the application and contains the main logic and fundamental functionalities that determine how the application will operate.

Infrastructure - This component is responsible for connecting and integrating various technical resources, such as databases, external services, or any other tool that might be necessary for the proper functioning of the application.

Web - This part represents the interface and visual portion of the application, allowing users to interact and benefit from the functionalities the application offers.

This, dividing the project into these three components ensures that each part has its clear role and responsibility, enabling better management and more effective code development.

2.2 Details of the three layers

Presentation Layer- The layer that is directly experienced by the user is called the presentation layer. It serves as a "contact" between the user and the system and is responsible for presenting data and information in a clear and usable manner. This is where users can see, interact with, and provide feedback on the information presented by the application.

Primarily established to offer a visual and interactive experience for the user, this layer can be realized through a web interface, a desktop application, or an interactive graphical interface, otherwise known as GUI. When it comes to web interfaces, the most common technologies used to construct and style these interfaces are HTML for the structure, CSS for styling, and JavaScript for interaction and dynamics. Everything is designed to ensure a seamless and usable interaction between the user and the application. (IBM Cloud, 2021)

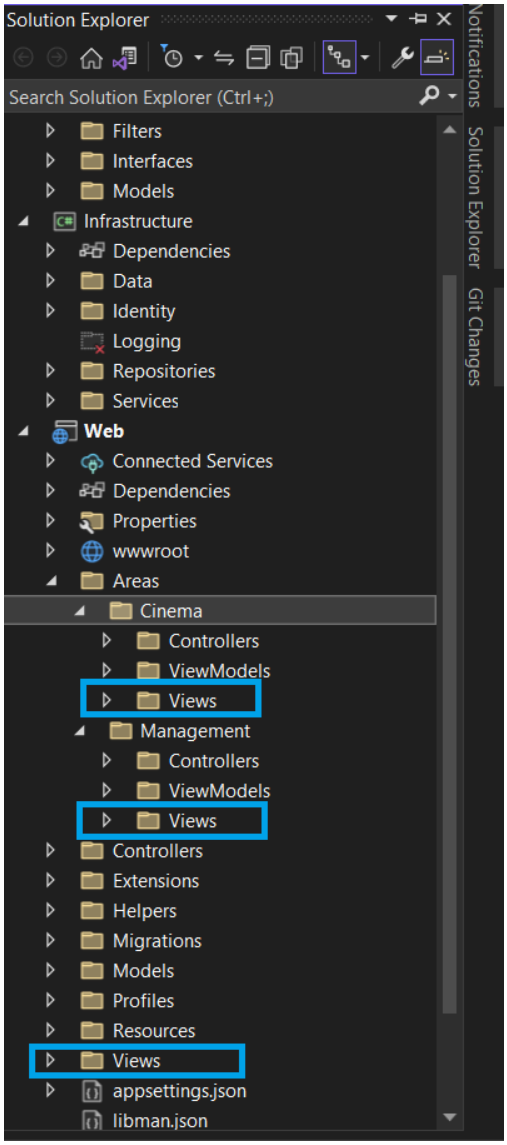

Fig 3. Presentation Layer

The figure shows details of the presentation layer, which I have named "Web". This is the section where information is structured and provided for viewing on a web page. More specifically, the presentation layer is further divided into several sub-layers, each having their specific functions and roles within the application.

These sub-layers are as follows:
User Management Sub-layer: This part is dedicated to the features and interfaces needed for users who are part of the management, providing them with tools and information specific to their role.
Customer Sub-layer: This section is focused on the interaction of customers with the platform, offering them a suitable and easy-to-use experience.
Common Sub-layer: This includes functionalities that are common and necessary for all users, such as authentication and identification processes on the website.
This detailed division and specification help us better understand how information and interactions are organized and provided for the various users of the application.

The application layer-also known as the business logic layer or the middleware layer, is the primary point of interaction and data processing within a computer system. It acts as a mediator between what the user sees and interacts with in the presentation layer and the data stored and managed in the database or the corresponding data layer.

In this area, data received from the user interface is processed and analyzed using a set of rules and logic specific to the application. This includes actions such as validating information, processing various requests, and interacting with the database to perform operations like adding, modifying, or deleting data.

This layer is crucial to ensure the application operates correctly, adhering to all business rules and requirements, and ensuring data is used and processed accurately and efficiently.

The data layer-also known as the primary information store or the data storage center, is the point where all application information is stored and managed. This layer often resides in a database management system, which might be of the relational type such as, for example, Oracle, MySQL, PostgreSQL, MariaDB, or Microsoft SQL Server. However, in cases where the data is less structured and more scalable, a NoSQL database like MongoDB, Cassandra, or CouchDB might be the appropriate solution. (IBM Cloud, 2021)

In a three-tier architecture, communication between the presentation layer and the data layer doesn't occur directly. Instead, every request and response must pass through the application layer, which acts as a mediator, ensuring all data is processed and validated before being stored or presented to the user. This strict separation aids in data protection and ensures the integrity of the information throughout the application. (IBM Cloud, 2021)

2.3 Functional and non-functional requirements

When it comes to building software, it's essential to understand and define the requirements it needs to meet. These requirements assist in determining what the software should do and how it should operate. Typically, these requirements are categorized into two main groups:

Functional Requirements: These are specifications that outline the primary functions and operations that software should provide. They are directly related to the actions and features that the software needs to perform, considering the needs and expectations of its users. This includes, for example, actions like creating an account, modifying a record, or searching for data.

Non-Functional Requirements: While functional requirements determine "what" the software does, non-functional requirements define "how" it performs those tasks. They relate to the quality and performance of the software. This might encompass issues such as how fast an application is, how secure it is from external attacks, or how the software scales when there's a large number of users.

For a more detailed overview of these requirements, I have presented them in a table where they are listed and described in detail.

Table 1. Functional requirement

Category	Functional requirement
Movie	• Complete movie registration details • Updating movie information • Deleting the movie from the database ○ Sorting and filtering movies
Schedules	• Updating the schedule's information • Deletion of the schedule lists • Organizing and filtering the schedule • Registration of the schedule for the entire week
Clients	• Registration of the client with complete details • Reverification and updating of client information

	• Removal of a client from the system • Organization, searching, and filtering of the client list • Client subscriptions • Client reservations • Viewing the ticket history • Changes and updates in the client's profile
Halls	• Registration of a new hall with all its dimensions • Modification and updating of the hall's information • Removal of a hall from the system • Management and organization of the halls
Tickets	• The capacity to create or generate tickets. • The tools to modify or update tickets. • The ability to remove or delete tickets. • The tools to organize and filter tickets.
Booking	• The capacity to make a reservation by selecting the client, schedule, and seat location. • The option to review and modify the information of a reservation. • The action to remove or cancel an existing reservation. • Tools that allow specific sorting and searching of reservations.
The manager	• The ability to analyze and view statistical data. • The capacity to produce and extract various reports. • The option to change and refresh details of the personal profile
Administrators	• The capacity to oversee and control financial transactions and payments. • The ability to administer and modify ticket information. • The option to coordinate and manage hall details. • The activity of managing subscriptions and their benefits. • The responsibility to organize and change movie data.

	• The possibility to create, modify, and manage various events. • The capacity to monitor and manage clients and their data. • The ability to plan and update schedules. • The option to change and refresh personal profile details.
SuperAdmin	• The ability to oversee and manage all aspects of the system. • Clear and unrestricted access to all functionalities and data of the system. • The capacity to control and manage all users and their roles within the system.
Roles	• The system consists of five user categories: the Client, the Manager, the Administrator, the Receptionist, and the Super Admin. • Responsibilities and access to the system should be divided and determined based on the specific role of each user. • A tailored interface is presented to the user based on their credentials during authentication.
The receptionist	• The capacity to administer and coordinate reservations. • The ability to register new clients in the system. • The option to create and issue tickets. • The activity of modifying and refreshing personal profile data.
Statistics	• All data related to the specific functions of the cinema are recorded and stored automatically. • The data is stored in a format that is suitable for analysis and creating statistics. • Automatic generation of reports and charts based on the gathered data for each functional section of the cinema is allowed.

Table 2. Non-functional requirements

Category	**Non-functional requirements**
Interface	• The user interface should provide familiar and intuitive functionalities for users,

	allowing for a simple and efficient use. The content and text presented on the interface should be clear and straightforward, ensuring that the user receives the intended message without any ambiguity. It is important for the interface to be "user-friendly", offering an impeccable user experience free from unnecessary complications. This assists in the intuitive and comfortable use of the application.
Software Specifications	• It is recommended to choose SQL Server as the base data platform for storage and management of information. This will ensure stability and high performance in data storage.
Availability	• It is expected that the system will be available and operational without interruption at all times, offering uninterrupted access 24 hours a day and 7 days a week. • Data preservation and recovery measures should be fortified and of high maturity, with a clear strategy for backup and restoration of the database. This aims to minimize the risk of information loss and reduce potential periods of application service interruption.
Performance	• It is essential that the system manages and stores a considerable amount of movies and clients efficiently, ensuring that there will be no interruptions or defects during operation. • When specific information is requested, the system's response time to display the data on the user interface should be fast and optimal, not exceeding a threshold of 3 seconds.
Safety	• The system is configured to accommodate different categories of users, and for each of them, there are specific levels of authorization and access. • Every transaction and communication within the system must offer a high level

	of security, relying on the most advanced and current technologies in the field of data security. • User passwords should be processed and stored in an encrypted (hash) manner, ensuring protection from unauthorized access. • The Super Admin figure will have the authority and capacity to intervene and manage all aspects of the system, allowing them to resolve any issue or problem that may arise.

2.4 Diagrams

Use Case diagram is an important tool for focusing on the main functionalities and the way users interact with a system or application. This diagram illustrates the primary behavior and functions of an application, reflecting specific use cases and the actors that interact with them. Through it, a clear picture is created of what the user can do with the system and how the system responds to these requests. ((Gemino, 2009)

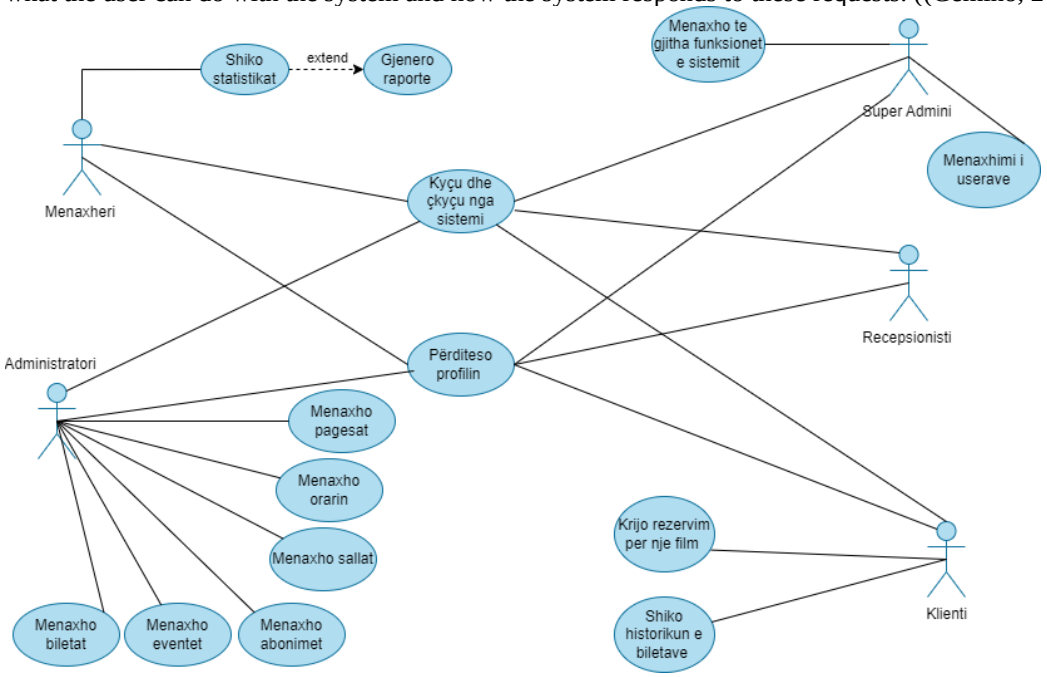

Figure 1: Use Case Diagram

The figure presents the Use Case Diagram for the entire system platform. Analyzing this diagram indicates that different users have varying levels of access and interaction depending on their function or role within the system. Meanwhile, there are five categories of users: four of them are responsible for managing and directing the system, while the fifth category is dedicated to clients who use the services provided by this system.

Class Diagram In the field of software engineering, a class diagram based on the Unified Modeling Language (UML) is a tool that represents the static construction and organization of a system. This diagram provides a detailed view of the structure of an application or system, revealing the classes that compose it, the specific features or attributes pertaining to each class, the functions or methods they perform, and the connections and relationships that exist between them. It is a visual representation that helps developers better understand and organize the interconnected components of the system they are building.

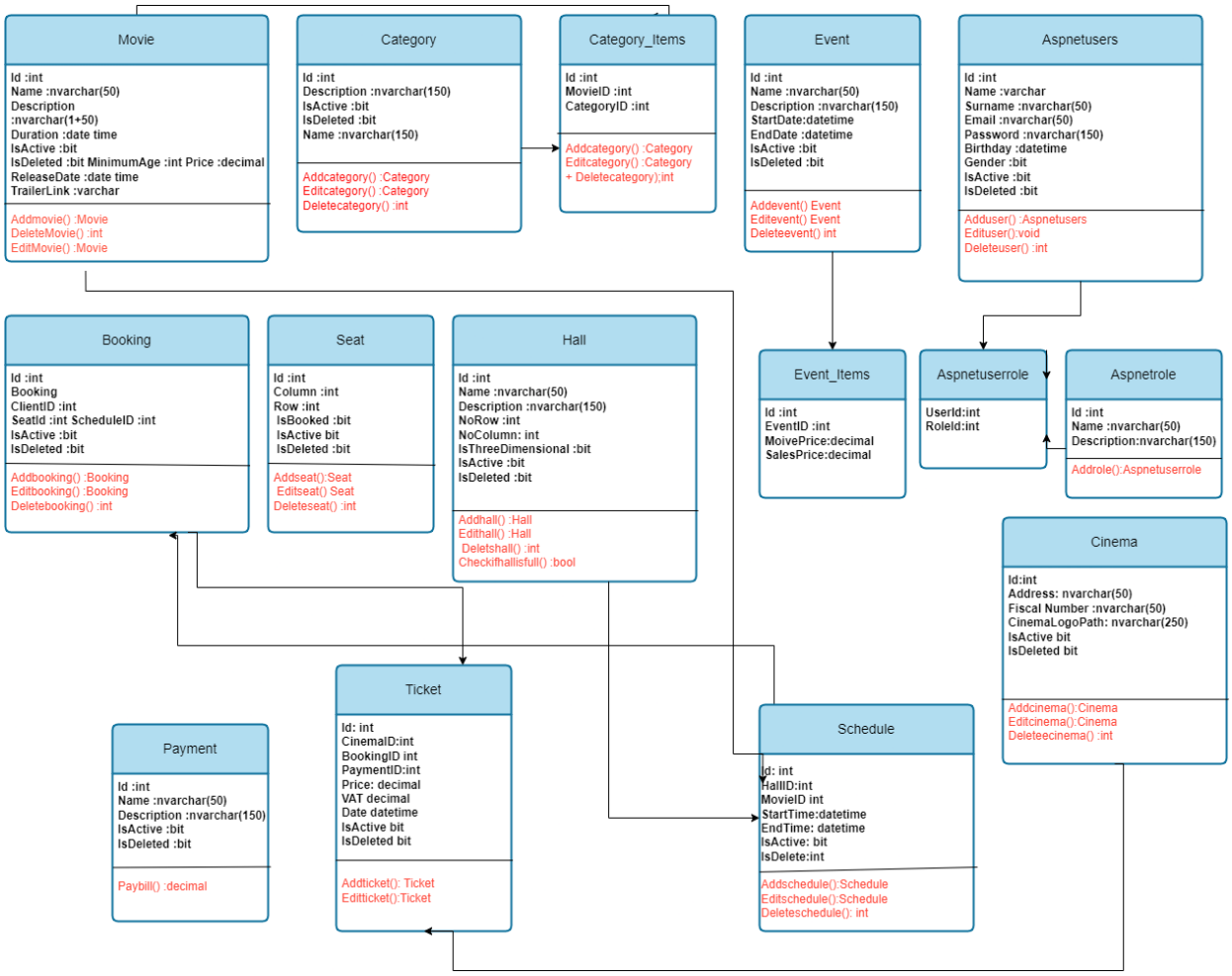

Figure 6. Class diagram

In the figure, the class structure diagram of the system is presented, revealing their fundamental components, such as the features and functions they include. This visualization brings out the interactions and connections between these classes, offering a clear view of how they are organized and interact within the overall system structure

The Entity Relationship Diagram visualizes how data is connected and structured amongst themselves. An entity represents a distinct unit or component of information. When we talk about a group of entities, we refer to a series of units with similar characteristics and features. Each entity is described by specific attributes that define its features or specifications.
By clarifying these entities and their connections, the diagram conveys a picture of the data's organization and interaction within a system. This type of diagram is a key tool to assist developers in building and optimizing a database structure

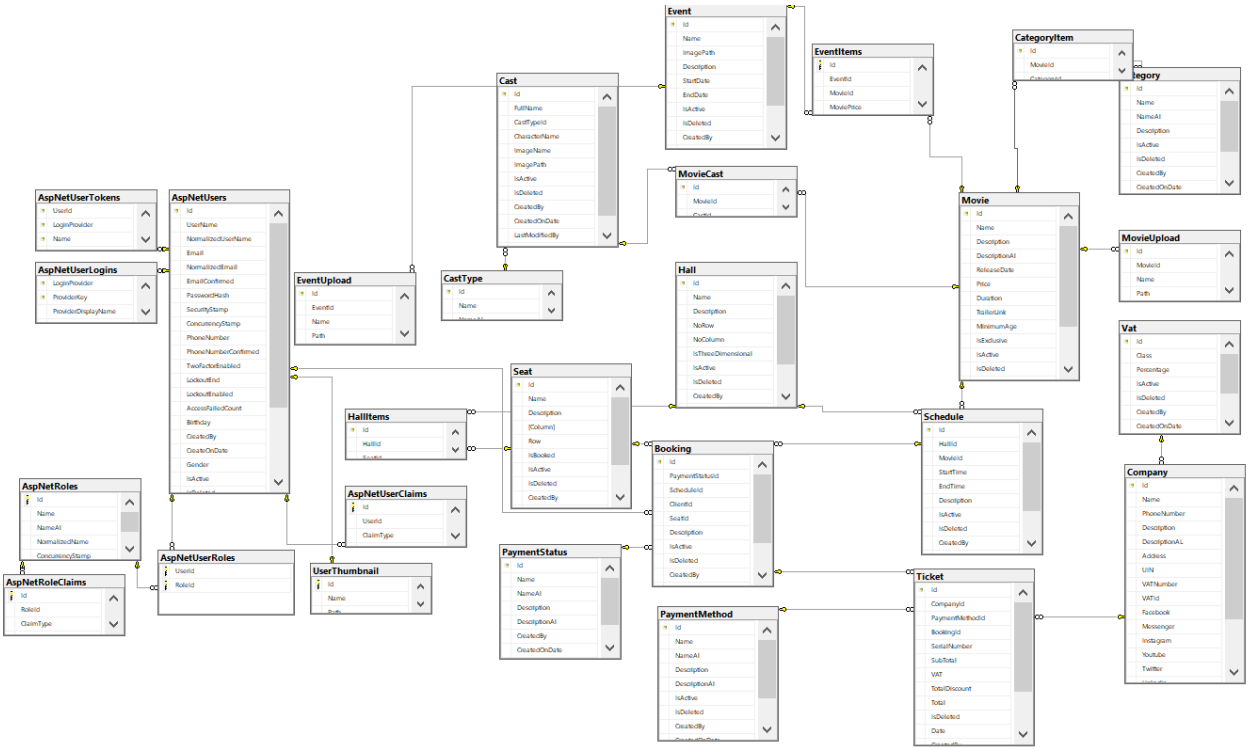

Figure 7: Entity Relationship Diagram

In this figure, the basic structure of the application's database is presented, where tables, along with their distinct characteristics, are distributed and connected. These reflective connections can be unique, multiple, or inverse, creating a network of links between the tables. This provides a clear understanding of how data interacts and is organized within the database system.

2.5 Configuration

The process of structuring the application is the responsibility of the super-administrator and includes three essential phases:

Setting the company parameters
Configuring the connections with social networks
Determining and configuring the system parameters.

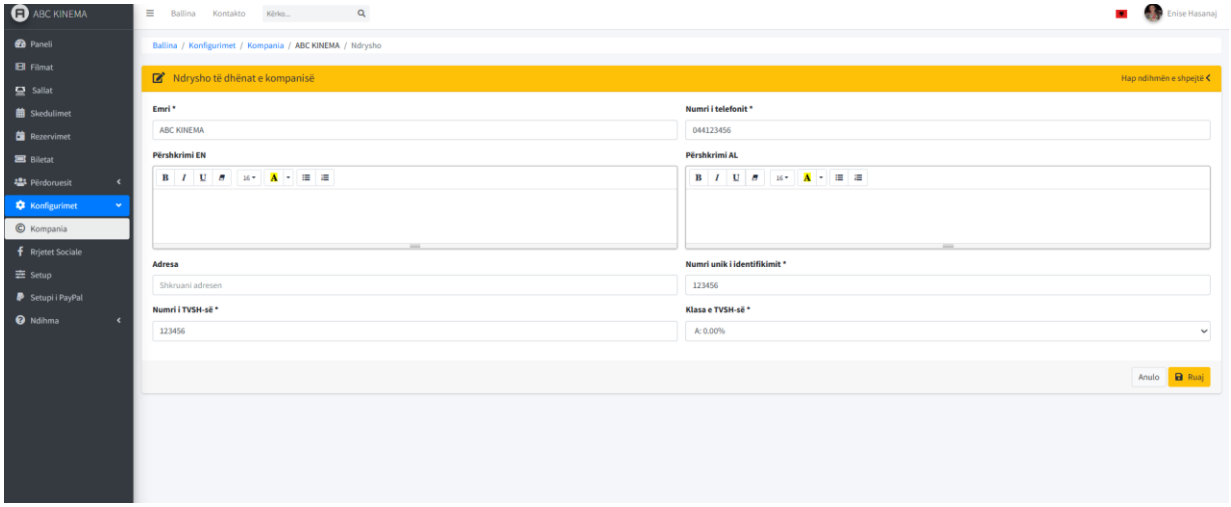

Figure 8: Setting the company parameters (Configuration)

In the figure, you can see the interface for entering the company's data, which will be the basis for calculating and invoicing tickets.

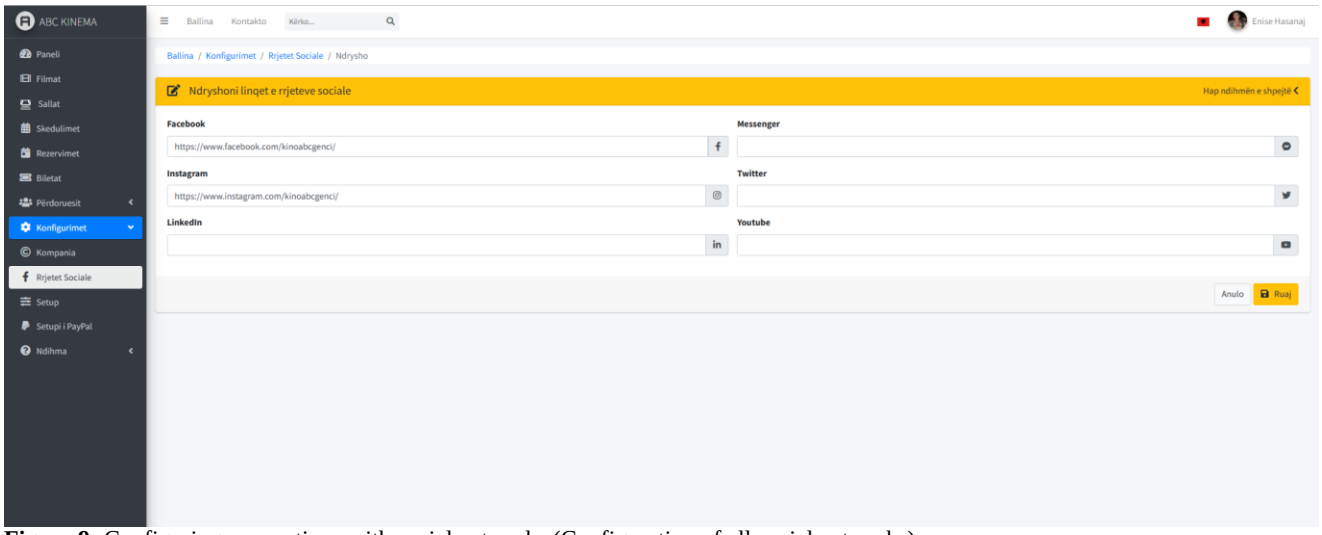

Figure 9: Configuring connections with social networks (Configuration of all social networks)

The figure displays the interface for setting up social network connections. These connections will be visible to clients in their interface.

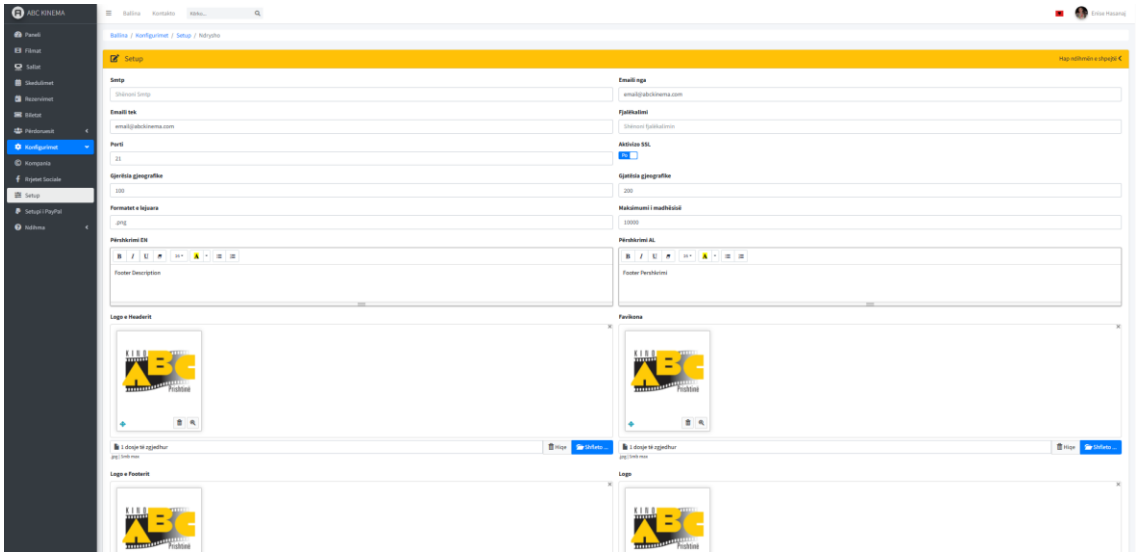

Figure 10: Determining and configuring the system parameters

The figure shows the interface where technical data for configuration is required. This includes details such as the SMTP server data for transmitting electronic messages, the geographical location of the company on the map, the file extensions that can be uploaded, the maximum upload capacity, the description and emblems of the company, and more.

3. RESULTS

After dividing the application into three main components - the part dedicated to management, the part addressing client needs, and the part functioning as a common platform for all functions - it's important to emphasize that this separation was done with the intent of optimizing and personalizing the experience for different users. Considering this division, below I have presented the visual aspect of each of these components, illustrating their interaction and specific function within the application's structure.

3.1 Management Part

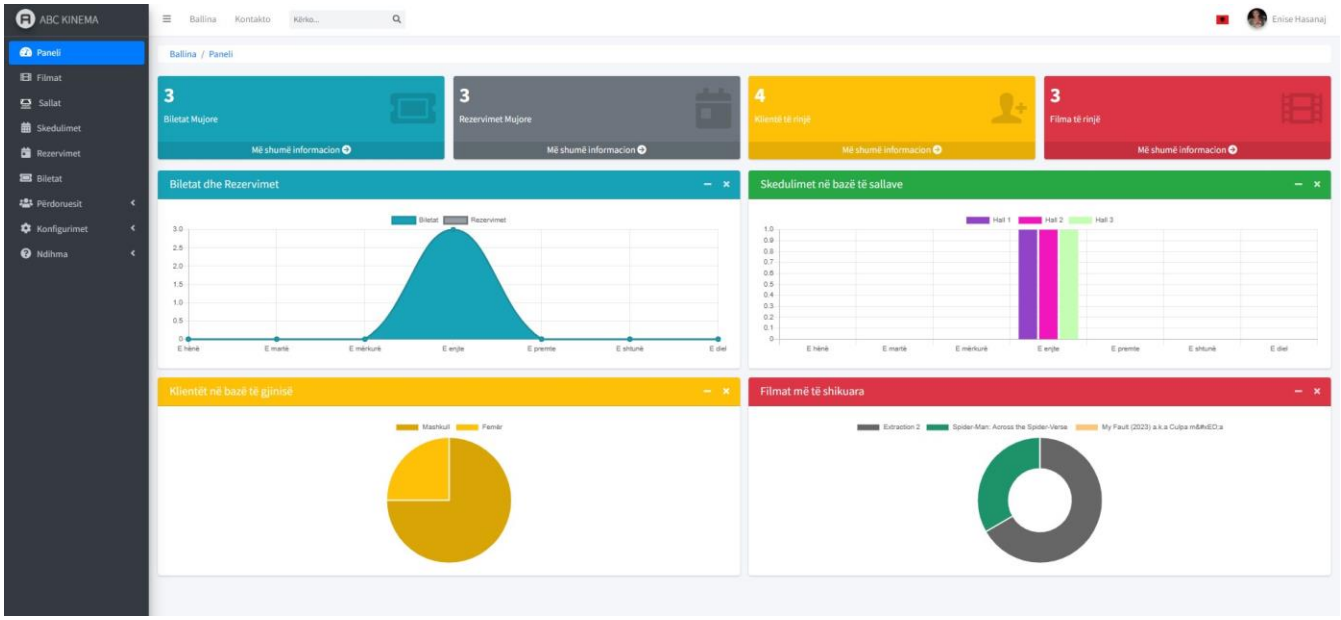

Figure 11: Control and Statistics Interface

In the figure, the control and statistics section is presented, which is an interface that allows managers and authorized users to view and analyze data in graphical and numerical form. This interface provides a clear overview of the system's performance, helping users make informed decisions based on concrete data.

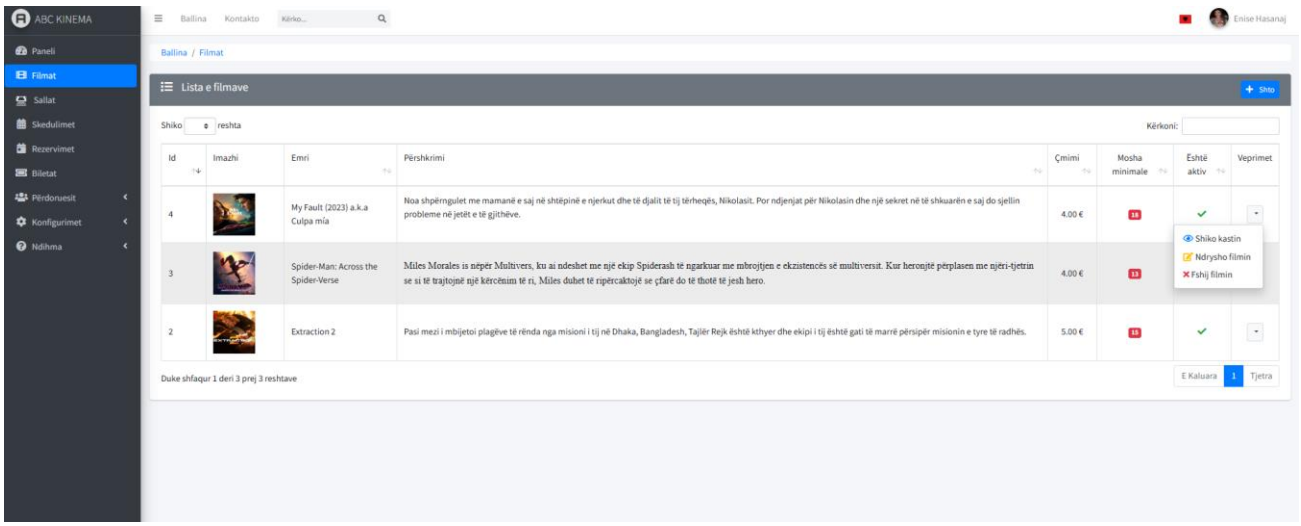

Figure 12: List of Movies

In this interface, there is a structured list of movies and their main details. For each movie, an image is displayed reflecting its content or the main scene, accompanied by a short description giving viewers a clear idea of what to expect from the movie. The movie's name is prominently displayed, while other details, such as the ticket price and the recommended viewer age, are clearly marked. Additionally, there's an indicator showing whether the movie is currently being shown or not. At the end of each row, there are additional options that allow actions such as editing, deleting, or adding new information for the specific movie.

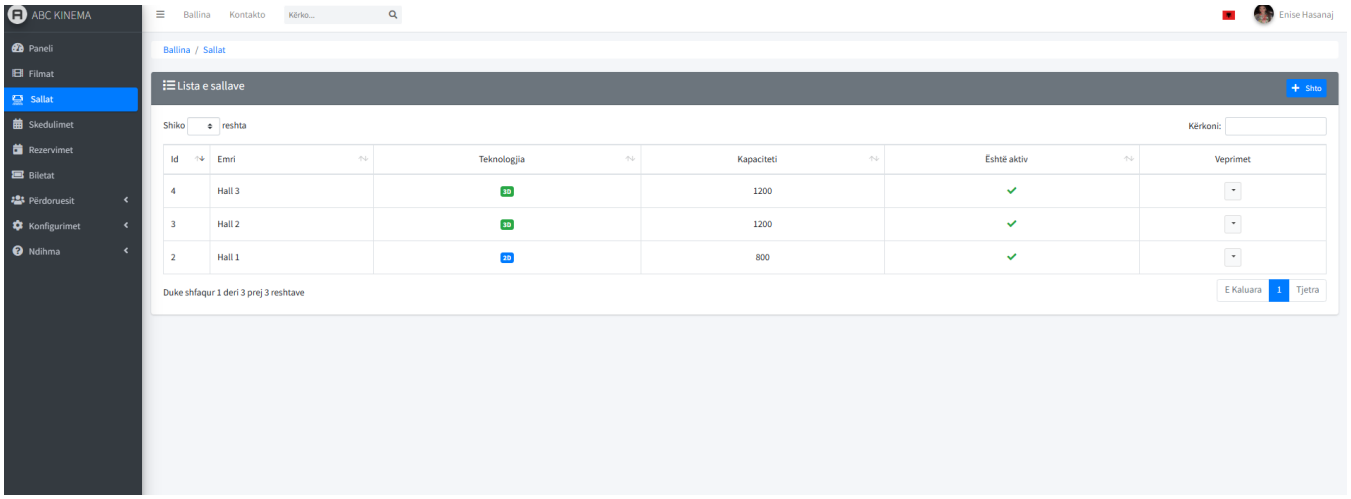

Figure 13: List of Halls

In this interface, a detailed list of halls and their specifications can be seen. For each hall, its name is clearly highlighted followed by the movie projection technology in 2D or 3D format. Next, the capacity of the hall shows the number of seats available for the audience. Also, a visual indicator shows whether the hall is currently active and available for screenings or not. For each hall, there are options available that offer the possibility to change the hall's data or, in certain cases, to delete the hall from the system.

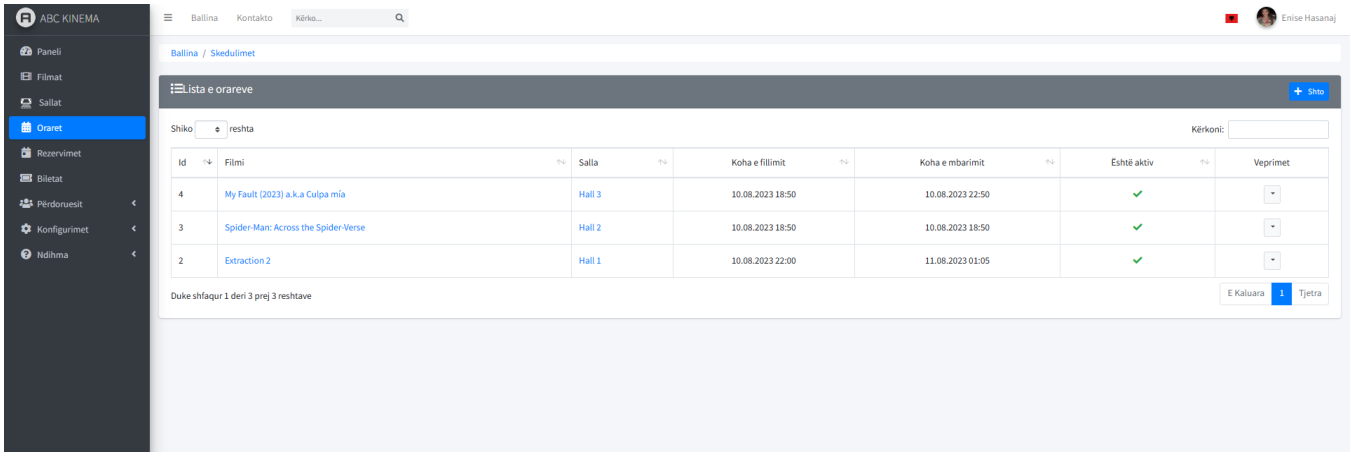

Figure 14: Schedule List

In this figure, a list of movie screening schedules and their specifications is displayed. For each screening, the movie title that will be shown is emphasized first. Subsequently, details about the hall where the movie will be screened are provided, giving an idea about the ambiance and the technology used. The start time of the movie screening is highlighted with the exact date and hour. Following that, comes the end time of the movie, indicating its duration. An indicator shows whether the movie screening is active or not. Additionally, options are integrated that allow changes to the schedules or other related data, as well as the ability to modify or delete schedules from the system

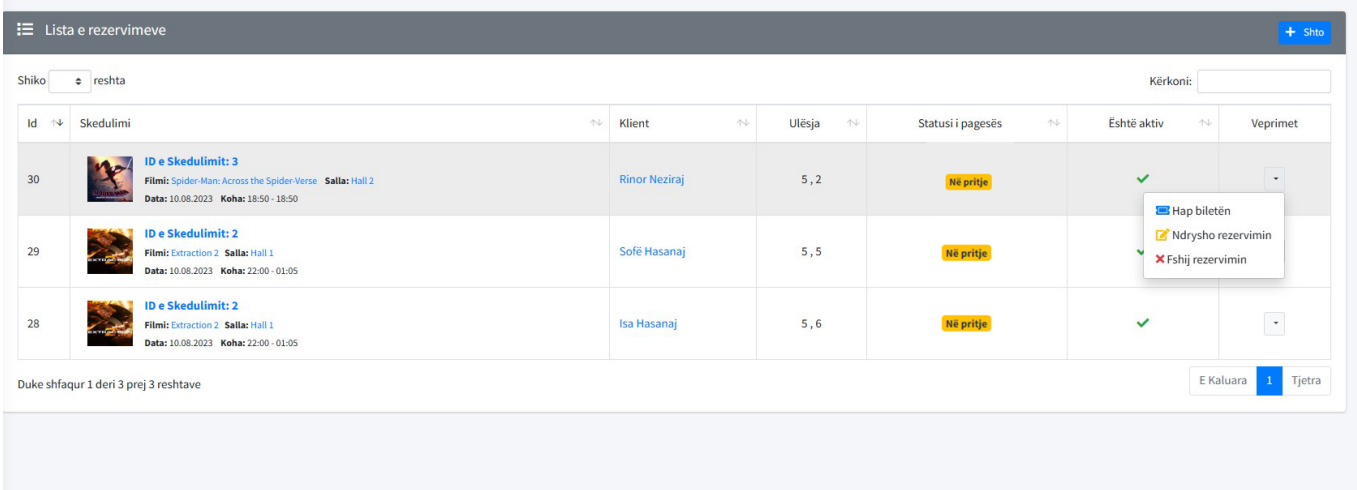

Figure 15: List of Reservations

In this figure, a detailed overview of reservations and the accompanying data is presented. Each reservation has an identifier known as "Reservation ID". This identifier is linked to specific details such as: the movie that's been reserved, the hall it will be screened in, as well as the date and time of the screening. The client who made the reservation is highlighted by name. On this list, users can also see the seat chosen by the client and the payment status, which can indicate whether the payment is still pending or has been completed. This list also provides information regarding the reservation's activity - whether it's still valid or not. To conclude, users have several options at their disposal to intervene in the reservation, including generating the ticket, modifying the reservation, or deleting it

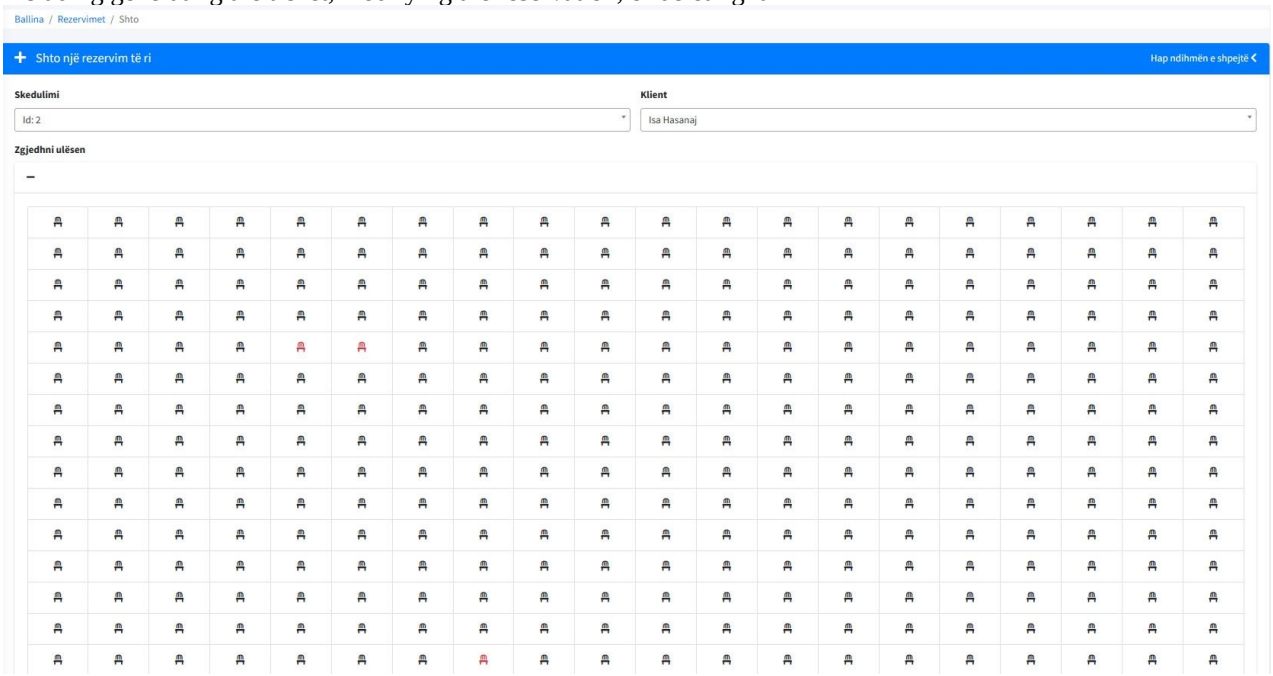

Figure 16: Interface for Adding Reservations

In this figure, the interface for adding reservations is displayed. All users who are part of the staff can access it. When clicking on "Add Reservation", the movie's ID and the name of the client wanting to make a reservation are shown. Also, the seat selection for the client is facilitated. Seats that are already reserved are displayed in red, while the available seats are marked in gray

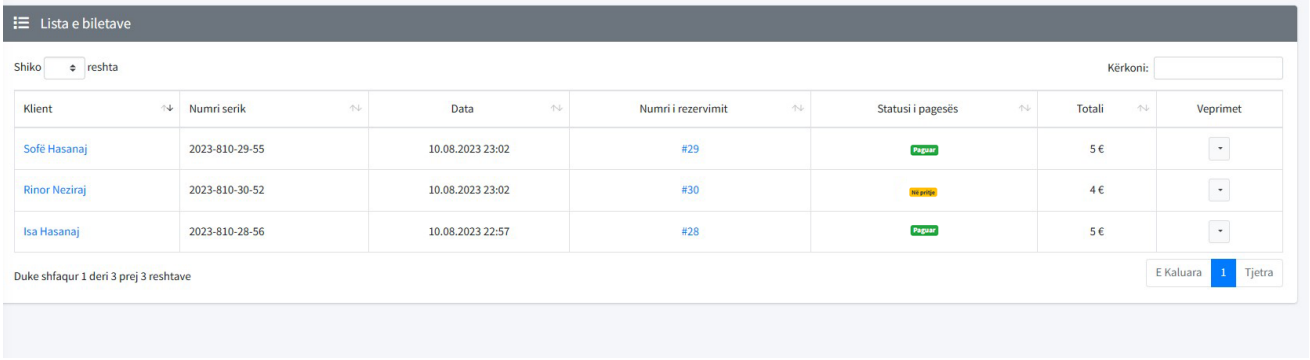

Figure 17: List of Tickets

In this figure, a list of tickets and their corresponding details is presented. Each displayed ticket has several essential pieces of information. This includes the name of the client who purchased the ticket, a serial number that uniquely identifies the ticket, the date and time when the ticket was issued, and a number associated with the anticipated reservation. The payment status is a key component of the list, indicating whether the ticket has been fully paid for or is still awaiting payment completion. The total amount paid or due is also highlighted. Lastly, users have several options available to intervene or take additional actions regarding the ticket, whether through modifications, verifications, or other action

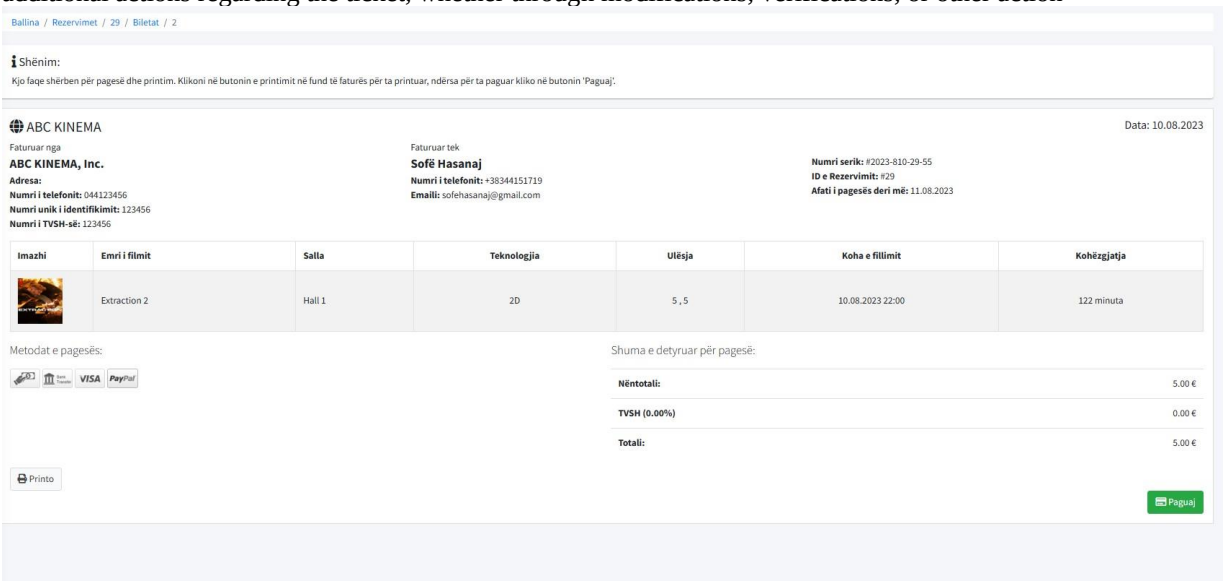

Figure 18: Ticket Generation

In this figure, ticket generation for all staff members is presented. They can generate a ticket, where during this ticket generation process all the cinema's details and the data of the client who made the reservation are displayed. A portion of the ticket displays the ticket's serial number, the reservation ID, and the payment deadline. Also, information about the movie the client has chosen to watch is presented, including elements like the movie poster image, the movie's name, the hall the client has chosen, and the selected technology for viewing the movie (2D or 3D). Additionally, the ticket is attached with information about the number and the seat chosen by the client, as well as the start and end times of the movie screening. Also, on the ticket, there is a payment option where users have several possible payment methods such as cash, Visa card, and PayPal, presented on the right side of the ticket. The obligated payment amount is clearly displayed, along with other options. Besides the aforementioned details, the ticket also displays an option to print the ticket.

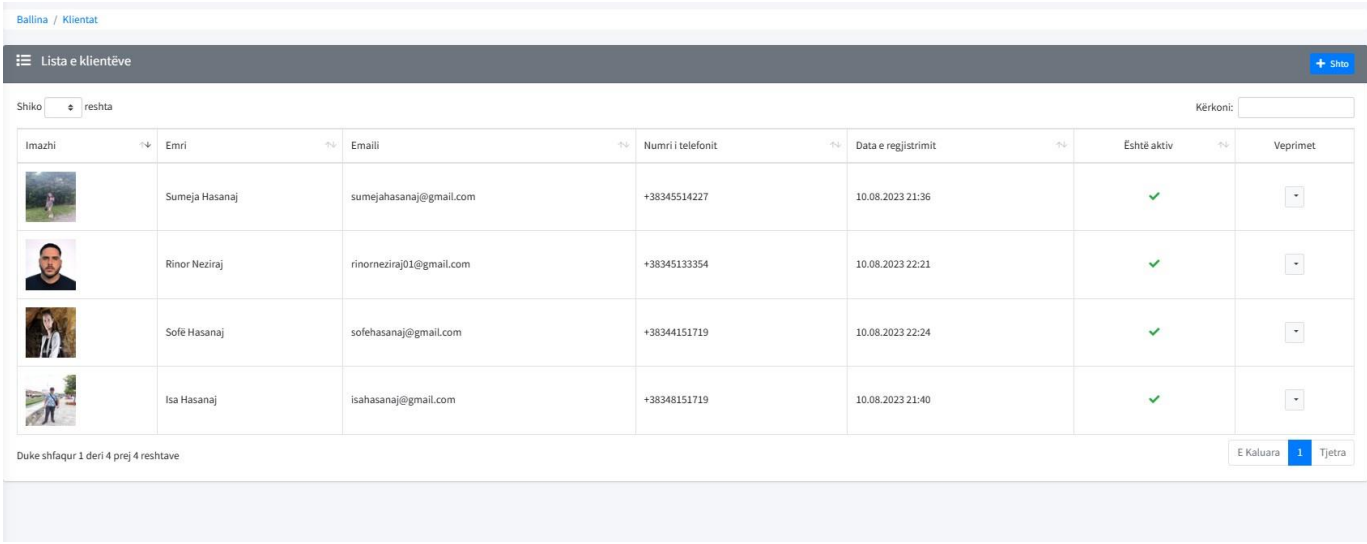

Figure 19: Client List

In this figure, an overview of clients and their defining data is seen. Each client on this list is identified through several key elements. Besides the client's image which offers visual identification, the first and last name are emphasized for quick recognition. The email and phone number provide means of communication with the client. The registration date indicates when the client first joined the service or platform. Additionally, the client's activity status is displayed to show whether their account is active or not. For each client, there are options for intervention or further modifications, which may include changes to their data or the deletion of their account

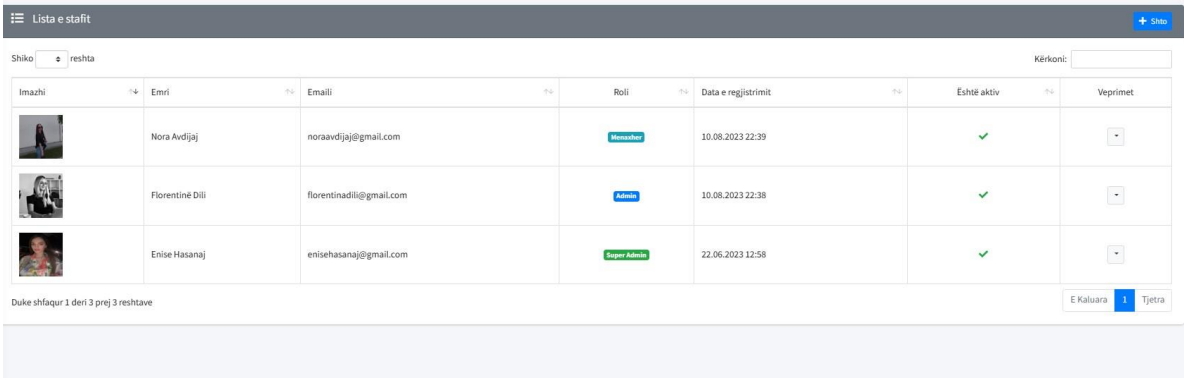

Figure 20: Staff List

In this figure, a comprehensive list of staff members and their specifics is displayed. Each member is represented with a photo that aids in quick visual recognition. The first and last names are clearly displayed for recognizing each staff individual. The email address assists in electronic identification and communication with them. Furthermore, the role of each member is specified, indicating which position they occupy, such as admin, superadmin, receptionist, or manager. The registration date has its significance, indicating when this member joined the team. Information about status indicates whether that staff member's profile is currently active or not. Additionally, there are options to intervene in each member's profile, altering or updating it as needed

3.2 Pjesa e klientit

Figure 21: Main Homepage

On the system's homepage, the client is presented with an attractive and functional interface that aids in simple and quick navigation. Upon entering the system, the first thing that catches the eye is a central section where several motivational quotes or brief information about the latest movies and suggestions to make a ticket reservation are placed.

Furthermore, a search engine is integrated. Clients can directly search for the movies they have in mind by typing the movie title or part of it. This facilitates a faster and clearer use of the platform.

On the right side of the screen, a special section is dedicated to upcoming movies in the cinema. This helps clients consider and plan their upcoming cinema visits.

One of the most distinctive features of the interface is the "subscribe" or subscription option. Clients who choose to subscribe can enjoy a range of exclusive advantages and benefits on the platform, such as discounts, early information on movie premieres, or special offers. On the main homepage interface, at the top, a structured horizontal menu offers straightforward navigation for the user. Following the "Home" option is the "Movies" option. When the user clicks on it, they are directed to a page where all the movies offered by the cinema are displayed. Movies can be sorted by release date, popularity, or category, and descriptions and showtimes can be viewed.

Next in the menu is the "About Us" option. This section provides a detailed overview of the

platform and its history, mission, and vision, as well as other information that can help clients familiarize themselves better with the company and the services it offers.

The "Contact" option is for users wishing to communicate with the cinema staff. This page may contain a form for sending messages, contact details like phone numbers, email address, and the cinema's physical address.

At the end of the menu, the "Join Us" or "Sign Up" option is crucial for new clients. Clicking on it takes them to a page where they can create an account and benefit from all the platform's features.

This complete and well-organized menu ensures that every client can easily find what they are looking for and benefit from a rich and swift experience on the cinema platform.

Through this interface, clients feel welcomed and informed, being offered an excellent and unparalleled user experience on this cinema platform.

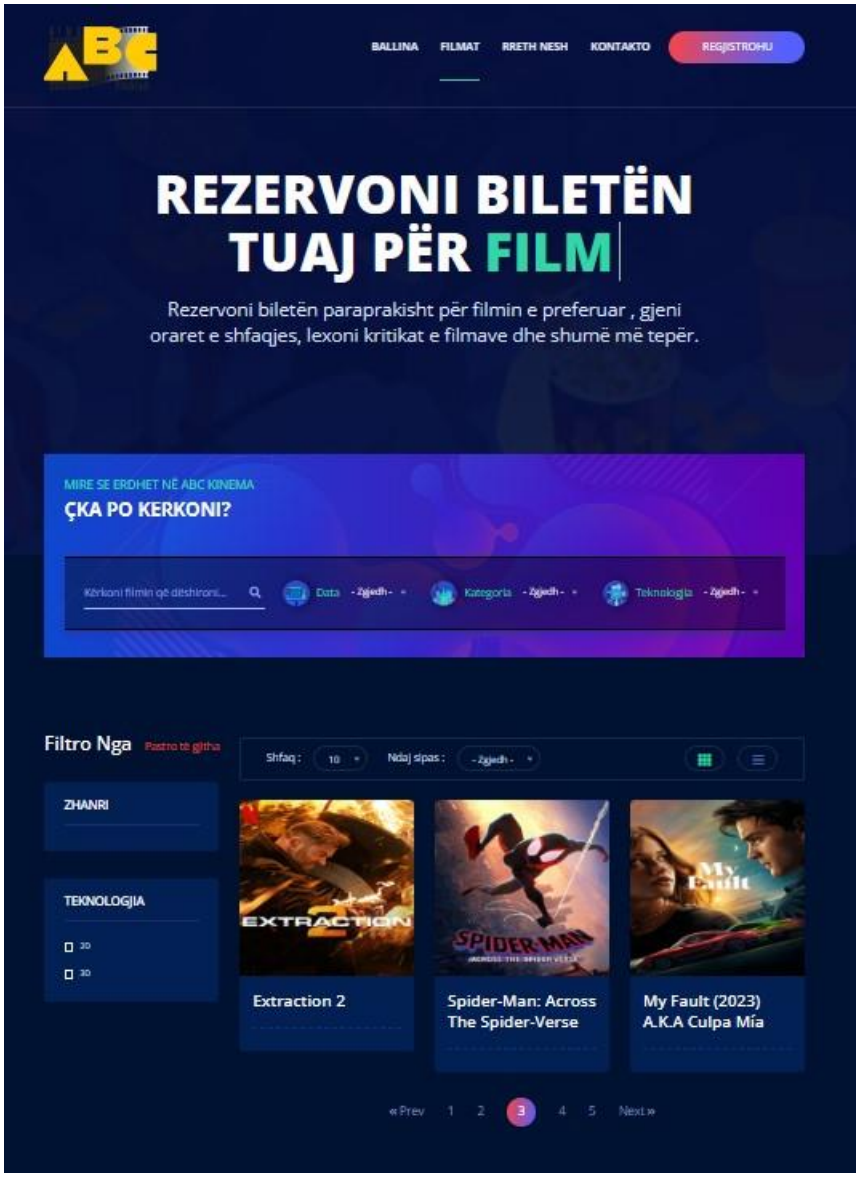

Figure 22: Movie Interface

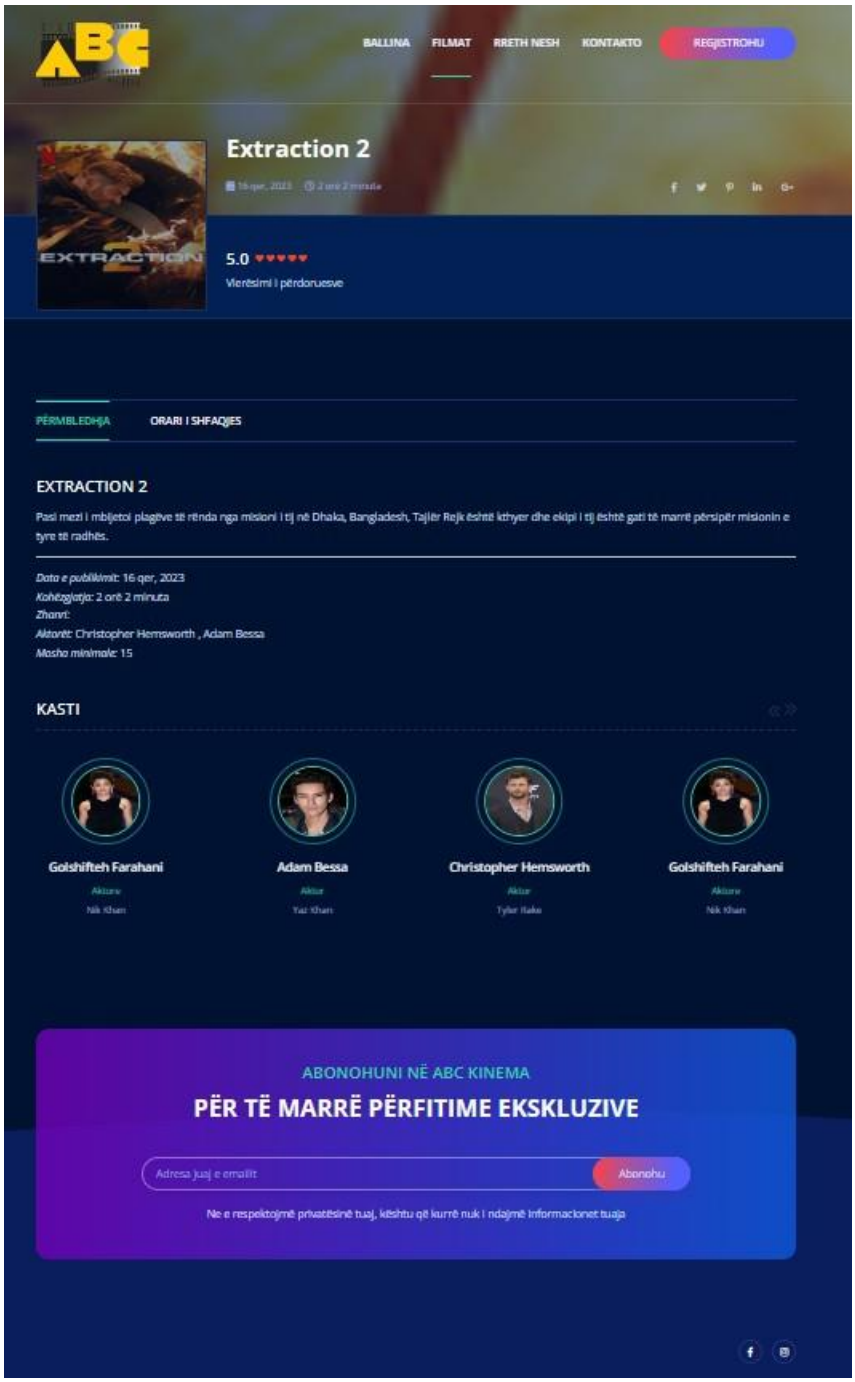

Figure 23: Movie Details

In the "Movies" menu option, users have the opportunity to explore the films offered by the cinema. When selecting a specific movie from the list, they are directed to a detailed page for that film. On this detailed page, at the top, the movie's name is displayed. Also prominently placed is an image of the film, usually the official poster or a primary scene. Below the image, users can see the genre of the film, which could be, for instance, action, drama, comedy, etc.
The release date and duration of the movie are situated in a separate section, giving users a clear idea of when the movie was released and its length.
The movie's description is located in a distinct part, usually below the main image or to its right. This description provides a brief overview of the film's main events and themes without giving away any spoilers.
In another section, the recommended age for viewing the film is displayed, assisting parents in making informed decisions about whether their children should watch the movie.
At the bottom of the page, a section is dedicated to the actors and the main crew involved in the film. Here, the names of the main actors are shown, and in some cases, their photographs, as well as the roles they play in the movie.

This detailed page offers a comprehensive overview to engage users and inform them about

various aspects of the movie before deciding to watch it in the cinema.

3.3 Common Parts

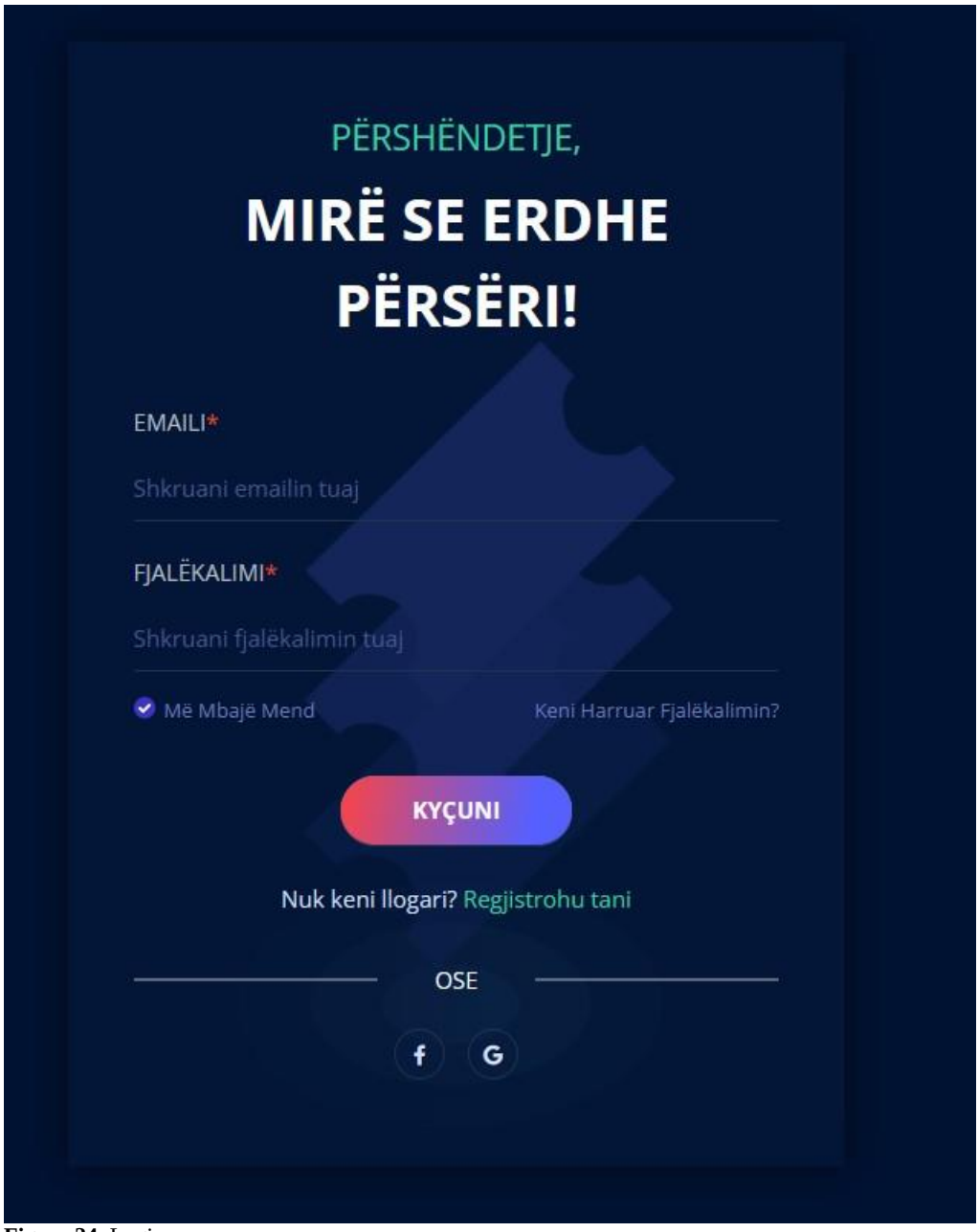

Figure 24: Login

In this figure, we see a user authentication interface. This interface is designed in a simple and effective manner, where users have the option to log into the system using their previously registered personal data. However, for those who want a faster and more secure method, there's also the option to authenticate using their Facebook or Google accounts. This offers an alternative and easy route for users who don't want to type in their credentials every time they wish to access the application.

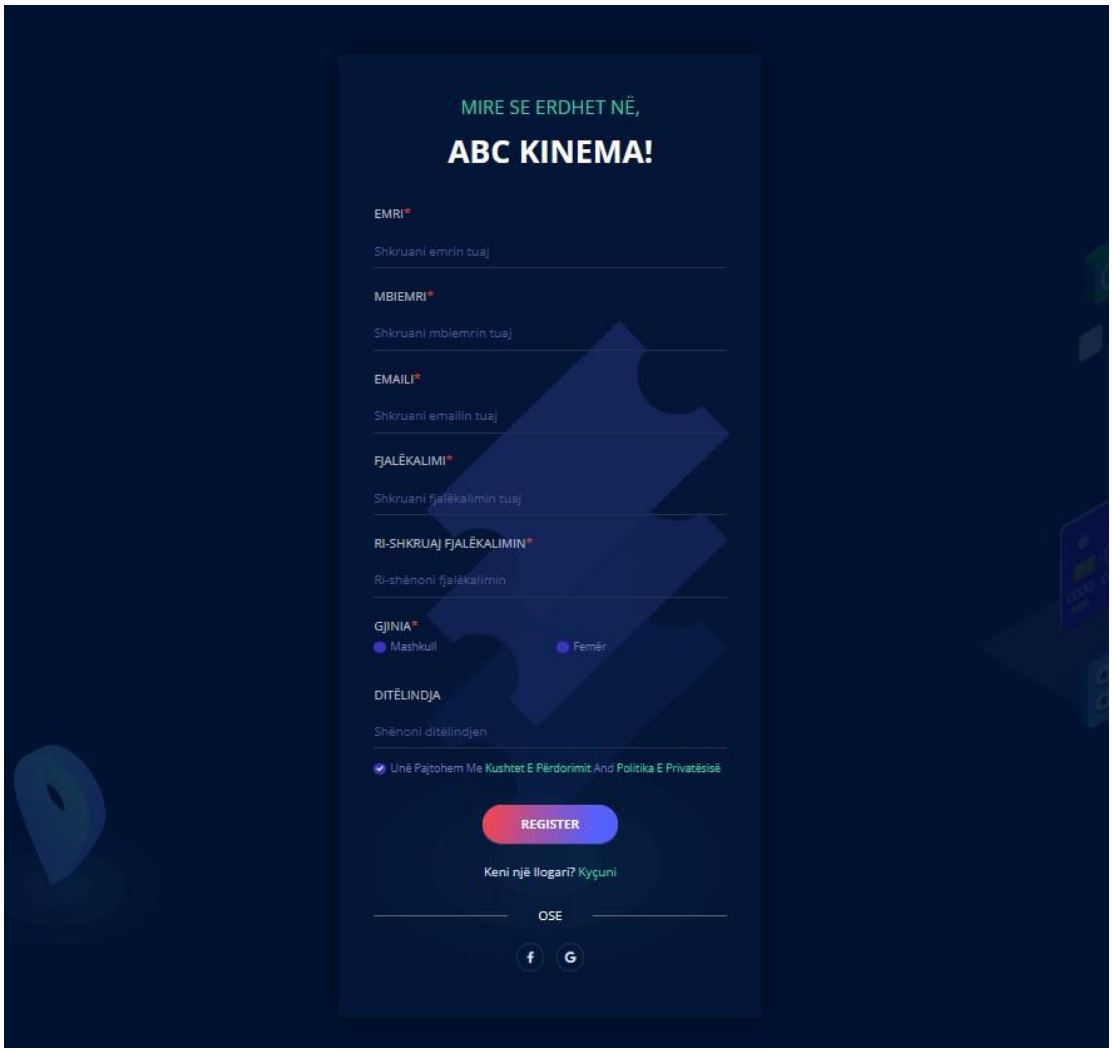

Figure 25: Registration

In this figure, an interface dedicated to creating a new account in the system is presented. The interface offers a simple and intuitive way to register the user's basic information. In addition to the possibility of using existing data from Facebook or Google accounts, users have the option to fill out manual fields such as: first name, last name, email, and password. An important feature of this form is the requirement to enter the password twice, a step that ensures users are aware of their choice and prevents potential mistakes. Additionally, the interface requests the user to specify their gender and date of birth, helping in creating a more complete and personalized user profile

4. Conclusion

Through the development of this project, I have reached a digital solution that assists us in managing all operations within the cinema, in an impeccable manner. This new technology allows us to coordinate and control all aspects of film operations with ease, while simultaneously reducing the workload within the work environment and contributing to the simplification of all procedures.

This project serves as an advanced tool to optimize and automate all processes. It provides us with a comprehensive solution for management, enabling control over every aspect of operations taking place within the cinema. From ticket purchases to the organization of screenings, from monitoring variable inventory to coordinating staff, this innovative platform offers stable and clear real-time control.

One of the main advantages of this project is its ability to enhance the quality of operations through the automation of routine tasks. This not only eases the workload for cinema staff but also increases the overall efficiency and precision of processes. Moreover, this project aids in creating a more refined experience for cinema customers, as the time and resources taken for management are used more intelligently and effectively.

In conclusion, this digital management solution offers us the opportunity to increase efficiency and control in all spheres of the cinema. From organizing daily tasks to achieving long-term strategic goals, this innovative platform brings about a significant change in the way we operate. This is not just technology, but a trustworthy partner that aids in the growth and success of your film business.

References

[1] ZHANG, H. L. (2006)-Design and implementation of management system of EA-PGR

based on ASP.Net.

[2] Zhao, G. &. (2012).-Design and Application of Information Management System Based on ASP.Net

[3] Gemino, A. &. (2009)- Use case diagrams in support of use case modeling: Deriving understanding from the picture. Journal of Database Management (JDM)

[4] Al-Ghrairi, A. H. (2021)- An Application of Web-based E-Healthcare Management System Using ASP.Net

[5] Adeoti-Adekeye, W. B. (1997)- The importance of management information systems.

[6] Hu, Y. P. (2003). Design Technology of Three-tier Architecture on Web Application Based on .NET. Computer Engineering.

[7] Dabbagh, M. &. (2014)- An approach for integrating the prioritization of functional and nonfunctional requirements. The Scientific World Journal.

[8] Aristocon (2021, July 09)- Why MIS Is Important For Businesses? https://aristotleconsultancy.com/mis-important-businesses/

[9] IBM Cloud. (2021, November 17)- What is three-tier architecture. https://www.ibm.com/cloud/learn/three-tier-architecture

[10] Techfunnel. (2021, May 31)-How Management Information Systems Can Help Businesses to Grow. https://www.techfunnel.com/information-technology/management-information-system/

[11] LAYOUTindex. (2019, April 9)- The importance of a cinema management system. https://blog.layoutindex.com/the-importance-of-centralizing-your-operations-in-a-cinema

[12] Manali gujarathi (2022, 22 March)- Configure Services and Configure method in ASP.NET https://www.geeksforgeeks.org/explain-configureservices-and-configure-method-in-asp-net/

[13] Ibrahim Šuta (2022) -ASP.NET Core ConfigureServices vs Configure https://codingblast.com/asp-net-core-configureservices-vs-configure

APPENDICES

Project\Web\Startup.cs

```
using System;
using Web.Models;
```

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Infrastructure.Identity;
using ApplicationCore.Entities;
using ApplicationCore.Interfaces;
using Infrastructure.Services;
using Infrastructure.Repositories;
using Microsoft.AspNetCore.Mvc.Razor;
using System.Globalization;
using Microsoft.AspNetCore.Localization;
using System.Collections.Generic;
using Microsoft.Extensions.Options;
using AutoMapper;
using Microsoft.Extensions.Localization;
using Web.Helpers;
using Rotativa.AspNetCore;

namespace Web {
    public class Startup {
        public Startup(IConfiguration configuration) {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to
add services to the container.
        public void ConfigureServices(IServiceCollection services)
    {
        services.Configure < RequestLocalizationOptions > (options => {
            var en = new CultureInfo("en-US");
            en.NumberFormat.NumberDecimalSeparator = ".";
            en.DateTimeFormat.ShortDatePattern = "dd/MM/yyyy";
            en.DateTimeFormat.LongTimePattern = "dd/MM/yyyy";
            en.DateTimeFormat.ShortTimePattern = "HH:mm";
            en.DateTimeFormat.LongTimePattern = "HH:mm";
            var al = new CultureInfo("sq-AL");
            al.DateTimeFormat.ShortDatePattern = "dd.MM.yyyy";
            al.DateTimeFormat.LongTimePattern = "dd.MM.yyyy";
            al.DateTimeFormat.ShortTimePattern = "HH:mm";
            al.DateTimeFormat.LongTimePattern = "HH:mm";
            al.NumberFormat.NumberDecimalSeparator = ".";

            var supportedCultures = new []
            {
                en,
                    al
            };

            options.DefaultRequestCulture = new RequestCulture(en, en);
            options.SupportedCultures = supportedCultures;
            options.SupportedUICultures = supportedCultures;
        });

        services.Configure < CookiePolicyOptions > (options => {
            // This lambda determines whether user consent for non-
essential cookies is needed for a given request.
            options.CheckConsentNeeded = context => false; // was true
            options.MinimumSameSitePolicy = SameSiteMode.None;
```

```
        });

        services.Configure < CookieTempDataProviderOptions > (options
=> {
            options.Cookie.IsEssential = true;
        });

        services.AddSession(options => {
            options.Cookie.IsEssential = true;
        });

        services.AddAuthentication()
            .AddFacebook(facebookOptions => {
                facebookOptions.AppId                                   =
Configuration["Authentication:Facebook:AppId"];
                facebookOptions.AppSecret                               =
Configuration["Authentication:Facebook:AppSecret"];
                facebookOptions.Scope.Add("public_profile");
                facebookOptions.Scope.Add("email");
                //facebookOptions.ClaimActions.MapJsonKey("firstName",
"first_name");
                //facebookOptions.ClaimActions.MapJsonKey("lastName",
"last_name");
                //facebookOptions.ClaimActions.MapJsonKey("birthday",
"birthday");
                //facebookOptions.ClaimActions.MapJsonKey("gender",
"gender");
            })
            .AddGoogle(googleOptions => {
                googleOptions.ClientId                                  =
Configuration["Authentication:Google:ClientId"];
                googleOptions.ClientSecret                              =
Configuration["Authentication:Google:ClientSecret"];
            });

        services.AddDbContext < ApplicationDbContext > (options =>

options.UseSqlServer(Configuration.GetConnectionString("DefaultConnecti
on")));

        services.AddDbContext < ApplicationDBContext > (options =>

options.UseSqlServer(Configuration.GetConnectionString("DefaultConnecti
on")));

        services.AddIdentity < ApplicationUser, ApplicationRole > ()
            .AddEntityFrameworkStores < ApplicationDbContext > ()
                .AddDefaultTokenProviders();

        services.Configure < IdentityOptions > (options => {
            // Password settings.
            options.Password.RequireDigit = true;
            options.Password.RequireLowercase = true;
            options.Password.RequireNonAlphanumeric = true;
            options.Password.RequireUppercase = true;
            options.Password.RequiredLength = 6;
            options.Password.RequiredUniqueChars = 1;

            // Lockout settings.
            options.Lockout.DefaultLockoutTimeSpan                      =
TimeSpan.FromMinutes(5);
            options.Lockout.MaxFailedAccessAttempts = 5;
            options.Lockout.AllowedForNewUsers = true;

            // User settings.
            options.User.AllowedUserNameCharacters =
```

```
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-._@+";
            options.User.RequireUniqueEmail = true;
        });

        services.AddAuthorization(option => {
            option.AddPolicy("SuperAdmin",          policy          =>
policy.RequireRole("Super Admin"));
            option.AddPolicy("Admin",             policy            =>
policy.RequireRole("Admin", "Super Admin"));
            option.AddPolicy("Manager",            policy           =>
policy.RequireRole("Admin", "Super Admin", "Manager"));
            option.AddPolicy("Receptionist",         policy         =>
policy.RequireRole("Admin", "Super Admin", "Receptionist"));
            option.AddPolicy("Client",            policy            =>
policy.RequireRole("Client"));
        });

        services.ConfigureApplicationCookie(options => {
            // Cookie settings
            options.Cookie.HttpOnly = true;
            options.ExpireTimeSpan = TimeSpan.FromMinutes(5);

            //options.LoginPath = "/Identity/Account/Login";
            //options.AccessDeniedPath                                =
"/Identity/Account/AccessDenied";
            options.SlidingExpiration = true;
        });


services.AddAutoMapper(AppDomain.CurrentDomain.GetAssemblies());

        // Add application services.
        services.AddTransient < ISelectListService, SelectListService >
();
        services.AddTransient < IEmailSender, EmailSender > ();
        services.AddTransient < IBookingRepository, BookingRepository >
();
        services.AddTransient < ICategoryRepository, CategoryRepository
> ();
        services.AddTransient < IEventRepository, EventRepository > ();
        services.AddTransient < IHallRepository, HallRepository > ();
        services.AddTransient  <  IMoviesRepository,  MovieRepository  >
();
        services.AddTransient < IScheduleRepository, ScheduleRepository
> ();
        services.AddTransient < ISeatRepository, SeatRepository > ();
        services.AddTransient < IErrorLogRepository, ErrorLogRepository
> ();
        services.AddTransient < IErrorLogService, ErrorLogService > ();
        services.AddTransient < IUserRepository, UserRepository > ();
        services.AddTransient < IRolesRepository, RolesRepository > ();
        services.AddTransient < IUserService, UserService > ();
        services.AddTransient < IFileHelper, FileHelper > ();
        services.AddTransient  <  IThumbnailService,  humbnailService  >
();
        services.AddTransient < ICompanyRepository, CompanyRepository >
();
        services.AddTransient < IVatRepository, VatRepository > ();
        services.AddTransient  <  ITicketRepository,  TicketRepository  >
();
        services.AddTransient < ISetupRepository, SetupRepository > ();
        services.AddTransient          <          IMovieCastRepository,
MovieCastRepository > ();
        services.AddTransient < ICastRepository, CastRepository > ();
        services.AddTransient < ICastTypeRepository, CastTypeRepository
```

```
> ();
        //services.AddScoped<INewsletterService, NewsletterService>();
        //services.AddTransient<IRepository, Repository>();
        services.AddControllersWithViews();

        services.AddLocalization(opts   =>   {   opts.ResourcesPath   =
"Resources"; });

        services.AddMvc()
            .AddViewLocalization(
                LanguageViewLocationExpanderFormat.Suffix,
                opts => { opts.ResourcesPath = "Resources"; })
            .AddDataAnnotationsLocalization();
    }

    // This method gets called by the runtime. Use this method to
configure the HTTP request pipeline.
    [Obsolete]
        public      void      Configure(IApplicationBuilder      app,
IWebHostEnvironment env)
    {
        if (env.IsDevelopment()) {
            app.UseDeveloperExceptionPage();
            app.UseDatabaseErrorPage();
        }
        else {
            app.UseExceptionHandler("/Home/Error");
            // The default HSTS value is 30 days. You may want to
change this for production scenarios, see https://aka.ms/aspnetcore-
hsts.
            app.UseHsts();
        }

        app.UseHttpsRedirection();
        app.UseStaticFiles();
        var options = app.ApplicationServices.GetService < IOptions <
RequestLocalizationOptions >> ();
        app.UseRequestLocalization(options.Value);
        app.UseCookiePolicy();

        app.UseRouting();

        app.UseAuthentication();
        app.UseAuthorization();
        app.UseSession(); // was added

        app.UseEndpoints(endpoints => {
            endpoints.MapControllerRoute(name:    "MyArea",    pattern:
"{area:exists}/{controller=Home}/{action=Index}/{id?}");
            endpoints.MapControllerRoute(name:    "default",    pattern:
"{area=Cinema}/{controller=Home}/{action=Index}/{id?}");
        });


RotativaConfiguration.Setup((Microsoft.AspNetCore.Hosting.IHostingEnvir
onment)env, "Rotativa");

    }
}
}
```

This is the application's configuration code. It consists of two main methods: ConfigureServices and Configure.

ConfigureServices:
This method is used to configure services that will be available throughout the application. In addition to the standard ASP.NET Core services, additional services for authentication, the database, and various

application services have been added.

Localization configuration has been set up for two languages: en-US and sq-AL.
Support for authentication with Facebook and Google has been added.
Authorization policies are configured.
Services for sending emails, services assisting in handling drop-down lists, etc. are registered.
Use of AutoMapper to facilitate mapping between models and application objects.

Configure:
This method is used to determine how HTTP requests will be processed by the application. This includes:
• Defining the development and production environments.
• Using authentication and authorization.
• Setting up the controller routes.
• Adding a middleware, a tool for generating PDFs with ASP.NET Core.

Project\Web\Helpers\FileHelper.cs

```
using ApplicationCore.Enums;
using ApplicationCore.Interfaces;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;

namespace Web.Helpers
{
    public class FileHelper : IFileHelper
    {
        private IWebHostEnvironment _env;
        private IThumbnailService _thumbnailService;

        public FileHelper(IWebHostEnvironment env, IThumbnailService
thumbnailService)
        {
            _env = env;
            _thumbnailService = thumbnailService;
        }

        public string GetProperFilePath(FileTypes type, Thumbnails
thumbnail, string path)
        {
            var properPath = "";
            try
            {
                var fileNameWithoutExtension =
Path.GetFileNameWithoutExtension(path);
                var fileExtension = Path.GetExtension(path);
```

```
                var pathWithoutFileName = Path.GetDirectoryName(path);
                var test = Path.GetFullPath(path);

                var                    newFileName                    =
$"{fileNameWithoutExtension}_{(int)thumbnail}{fileExtension}";

                var pathArray = path.Split('/');

                pathArray[pathArray.Length - 1] = newFileName;

                properPath = string.Join('/', pathArray);

                var absolutePath = pathWithoutFileName.Substring(1) +
"\\" + newFileName;
                absolutePath = _env.WebRootPath + absolutePath;
                if (!File.Exists(absolutePath))
                {
                    throw new Exception();
                }

            }
            catch (Exception)
            {
                properPath     =     $"~/images/default-images/product-
default_{(int)thumbnail}.jpg";
            }

            return properPath;

        }
        public string GetProperFilePath(FileTypes type, Thumbnails
thumbnail, string path, bool forLogo)
        {
            var properPath = "";
            try
            {
                var             fileNameWithoutExtension             =
Path.GetFileNameWithoutExtension(path);
                var fileExtension = Path.GetExtension(path);

                var pathWithoutFileName = Path.GetDirectoryName(path);
                var test = Path.GetFullPath(path);
```

```
                var                     newFileName                     =
$"{fileNameWithoutExtension}_{(int)thumbnail}{fileExtension}";

                var pathArray = path.Split('/');

                pathArray[pathArray.Length - 1] = newFileName;

                properPath = string.Join('/', pathArray);

                var absolutePath = pathWithoutFileName.Substring(1) +
"\\" + newFileName;
                absolutePath = _env.WebRootPath + absolutePath;
                if (!File.Exists(absolutePath))
                {
                    throw new Exception();
                }

            }
            catch (Exception)
            {
                properPath = null;
            }

            return properPath;

        }
        public string GetFavIconFilePath(string path)
        {
            string properPath;
            try
            {
                var             fileNameWithoutExtension             =
Path.GetFileNameWithoutExtension(path);
                var fileExtension = Path.GetExtension(path);

                var pathWithoutFileName = Path.GetDirectoryName(path);
                var test = Path.GetFullPath(path);
                var newFileName = Path.GetFileName(path);

                //var                   newFileName                   =
$"{fileNameWithoutExtension}_{(int)thumbnail}{fileExtension}";

                var pathArray = path.Split('/');
```

```
                pathArray[pathArray.Length - 1] = newFileName;

                properPath = string.Join('/', pathArray);

                var absolutePath = pathWithoutFileName.Substring(1) +
"\\" + newFileName;
                absolutePath = _env.WebRootPath + absolutePath;
                if (!File.Exists(absolutePath))
                {
                    throw new Exception();
                }

            }
            catch (Exception)
            {
                properPath = null;
            }

            return properPath;

        }
        public void SaveFile(FileTypes type, IFormFile file, string
folderName, string id, params int[] thumbnails)
        {
            var filePath = Path.Combine(_env.WebRootPath, "uploads",
folderName, id, type.ToString());
            if (!Directory.Exists(filePath))
            {
                Directory.CreateDirectory(filePath);
            }
            (new FileInfo(filePath)).Directory.Create();
            using (var fileStream = new
FileStream(Path.Combine(filePath, file.FileName), FileMode.Create))
            {
                file.CopyTo(fileStream);

                fileStream.Close();
            }
            var fileNameWithoutExtension =
Path.GetFileNameWithoutExtension(file.FileName);
            var fileExtenstion = Path.GetExtension(file.FileName);

            foreach (var item in thumbnails)
            {
```

```
                string    thumbnailPath    =    Path.Combine(filePath,
$"{fileNameWithoutExtension}_{item}{fileExtenstion}");
                _thumbnailService.GenerateThumbnail(item,
Path.Combine(filePath, file.FileName), thumbnailPath);
            }
        }

        public void SaveFavIcon(FileTypes type, IFormFile file, string
folderName, string id)
        {
            var  filePath  =  Path.Combine(_env.WebRootPath,  "uploads",
folderName, id, type.ToString());
            if (!Directory.Exists(filePath))
            {
                Directory.CreateDirectory(filePath);
            }
           (new FileInfo(filePath)).Directory.Create();
            using         (var         fileStream         =         new
FileStream(Path.Combine(filePath, file.FileName), FileMode.Create))
            {
                file.CopyTo(fileStream);

                fileStream.Close();
            }
        }
        public  void  SaveImage(FileTypes  type,  IFormFile  file,  string
folderName, string id)
        {
            var  filePath  =  Path.Combine(_env.WebRootPath,  "uploads",
folderName, id, type.ToString());
            if (!Directory.Exists(filePath))
            {
                Directory.CreateDirectory(filePath);
            }
            (new FileInfo(filePath)).Directory.Create();
            using         (var         fileStream         =         new
FileStream(Path.Combine(filePath, file.FileName), FileMode.Create))
            {
                file.CopyTo(fileStream);
                fileStream.Close();
            }

            var                fileNameWithoutExtension                =
Path.GetFileNameWithoutExtension(file.FileName);
```

```
            var fileExtenstion = Path.GetExtension(file.FileName);

        }

    }
}
```

This part of the code assists in managing files and aids in storing and retrieving various files.
The FileManager class is a helper for different file operations, including saving and retrieving files and assisting in generating different files, such as thumbnails (small images).

Project\Web\Helpers\IFileHelper.cs

```
using ApplicationCore.Enums;
using Microsoft.AspNetCore.Http;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Web.Helpers
{
    public interface IFileHelper
    {
        void SaveFile(FileTypes type, IFormFile file, string
folderName, string id, params int[] thumbnails);
        void SaveFavIcon(FileTypes type, IFormFile file, string
folderName, string id);
        string GetProperFilePath(FileTypes type, Thumbnails thumbnail,
string path);
        string GetProperFilePath(FileTypes type, Thumbnails thumbnail,
string path, bool forLogo);
        string GetFavIconFilePath(string path);
        void SaveImage(FileTypes type, IFormFile file, string
folderName, string id);
    }
}
```

This is an interface that specifies the helper operations for files, so it's a helper for FileHelper.cs.