Implementing a Movie Theater Ticket Sale Software

Pedro Cruz Agosto
Master of Engineering in Computer Engineering
Advisor: Alfredo Cruz-Triana, PhD
Polytechnic University of Puerto Rico
Graduate Project EXPO, February 2025

Abstract *— A software solution for movie theaters is developed to streamline operations, improve the experience of the user and the customers, and help in the process of making business decisions. The software incorporates logic that enables the creation of schedules, showing the resources available in each theater room and the times they are set to be occupied. The software helps operations by performing sales quickly. It also includes features for user management with role-based access control and theater configuration including room settings. The system keeps track of each ticket sold and prevents double booking of the same seat for the same show. The system also ensures the data integrity is preserved and tracks changes.*

Key Terms *— Digitalization, Movie Theater, Software Design, Software Development.*

INTRODUCTION

Movies are a form of entertainment that has existed for many years. People gather at movie theaters to watch them either alone or with friends or family. Many different movies of different genres are available at any time. The demand for each film changes constantly, and the movies that sell more tickets one week might not do so the next. Different factors can be chosen to match the preferences of the public [1] [2]. Scheduling correctly is important for ticket sales and mistakes cost a lot. It is possible the time and space reserved for a film could be used better.

A system that performs sales and allows the configuration of the rooms and shows in the theaters was developed. It supports business processes and decisions and makes operations more efficient by considering important factors like the user experience, the business value, and the data integrity and security [3] [4]. If the program looks useful and looks easy to use, people are more likely to use it. Research on user experience has found things affect what a user thinks of an application [5]. The terms and qualities identified vary. To provide a good experience, the software must be: Easy to learn and understand, Consistent, Visually Attractive, Satisfying. The business value can be an improvement in operations by being flexible, efficient, effective, and reliable [6] [7]. Missing one of these qualities can result in having to do manual operations. Data integrity and security is about making sure the information in the database is valid, complete, has not being altered, and has not been accessed or entered by people that should not see it.

Scheduling involves many variables, and a bad schedule can cause extended periods of time with no shows in a room or occupying a room for a show that sells far less than what it costs to run the theater. To overcome this issue, the function developed allows visualizing the current schedule of the day, and it continuously updates as changes are made.

METHODOLOGY

The application is aimed at the movie theater industry. It is composed of databases, microservices, and a client application. This system can digitize all the basic operations of a movie theater. The application allows configuring theaters and their movies. This information is then used as the master data for creating showing schedules. These schedules are the offering for the day for each movie theater and are used for selling tickets.

Design

The architecture design of the system is explained below. Figure 1 shows a High-level System Architecture Diagram. In this diagram, it is

shown that the overall architecture is client-server, and the server part is designed as microservices.

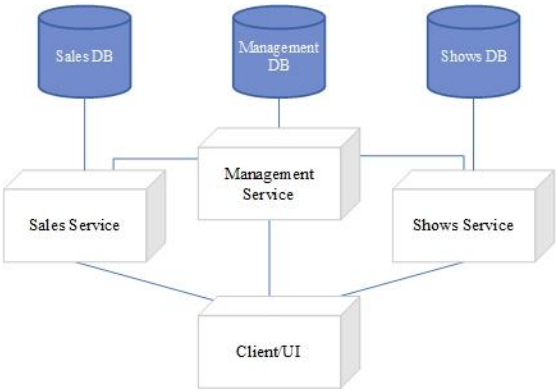

Figure 1
High-level System Architecture Diagram

Microservices are designed around domains, and three (3) domains are identified. These are Management, Shows, and Sales. Each service has its own dedicated database. The Shows and Sales services reference the Management service because it handles authentication. The client has a set of views for the users to gain access to the system's functionality and it interacts with the services.

Database Design

The system has three (3) SQL databases. The databases have master tables and transactional tables. System-versioning is enabled for all the transactional tables to track changes in the records. The system-versioned temporal tables are used in the application to show the changes a record had each time it was changed and the person who made that change. The tables feature soft deletion to keep information about changes.

1. Management Database - It stores users, roles, permissions, and configurations related to movie theaters, including rooms available and seats. The theater settings belong to one facility and multiple facilities can be configured. Information in this database supports operations that are based on the other databases. It contains master tables for permission types and permissions available.
2. Shows Database - It stores information related to movies, advertisements, and scheduled shows. It contains master tables for age ratings and features.
3. Sales Database - It stores details about orders, payments, and tickets sold. It contains master tables for payment methods.

Services Design

The system uses REST API services following the microservice architecture. Figure 2 shows the Microservice Diagram for one (1) service. The system has three (3) microservices, one per domain. These are Management Service, Shows Service, and Sales Service.

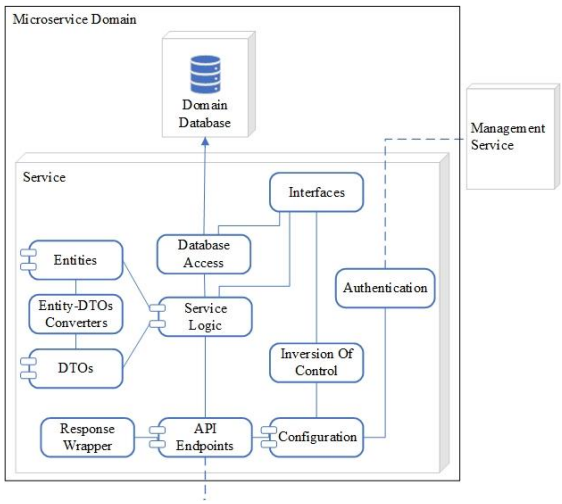

Figure 2
Microservice Diagram

The service receives HTTP requests through the API controller endpoints. Before the request is processed, it performs an authentication check. For the Sales and Shows service, the authentication logic will communicate with the Management Service to perform the check. This process is internal for the Management Service. The endpoints call the service logic and perform the requested operation. The service logic calls the database access classes depending on the operation. If the operation involves data, the data in the request received is deserialized and converted from a Data Transfer Object (DTO) to an entity model, and if the data is a response, it is converted from an entity model to a DTO. The service design is centered around the Inversion of Control principle.

This pattern allows the substitution of component dependencies without requiring the modification of the code in the components that call them. The only requirement is that the interface stays the same and it is included in the component that consumes it, rather than the one that implements it. This improves decoupling and maintainability. The configuration section also sets up the authentication logic. The Service responses are wrapped and then serialized in a JSON format.

Client Design

The client runs on the computer of the user. Figure 3 shows the Client-side Diagram, which presents the interaction between the code components. This explains more about the design for the client portion of the application.

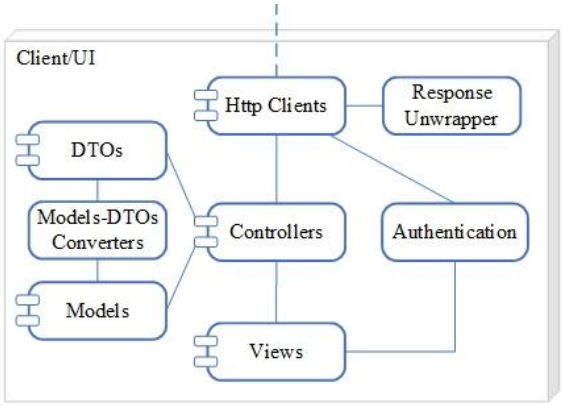

Figure 3
Client-side Diagram

The UI has lists, input form, and detail windows. The user interacts with those to give the application commands. These commands call the underlying business logic and perform the requested operation. Some operations include validations to ensure invalid data does not reach the database. If the validations show the data is invalid or incomplete, an error is shown to the user and the operation ends. If logic requires database information or is about writing valid data, it consumes the HTTP Client classes. The HTTP Client classes contain the methods needed to contact the services. These classes receive parameters and retrieve information in the form of wrapped DTOs. The Response Unwrapper extracts the DTOs. Controllers convert them to models so they can be used by the application.

Visual Design

The application needs to be easy to learn and understand. It is visually consistent across the system. Views of collections like users, theaters, and movies are presented as tables. Tables are an organized and concise form of showing information. These lists are presented in the main window and are accessed from the navigation bar. To add or change an element for a collection, a dedicated window is provided with the input fields. Data that references other data, like the movie of a show, uses dropdowns to limit the input to valid values. A read-only window is used for viewing.

Figure 4 presents a prototype of the visual design. In it, the navigation bar appears above the functionality content and shows that it has collapsible submenus. Clicking or hovering over the navigation bar with the mouse, touch pad or touchscreen expands the submenus and allows the user to access the desired function.

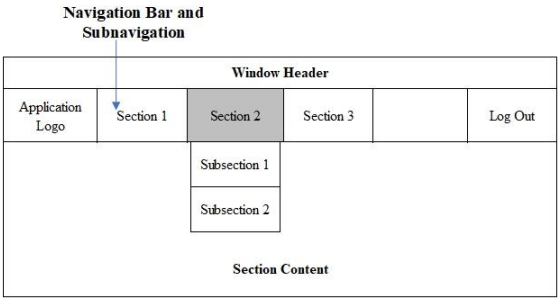

Figure 4
Visual Design

Development – Databases

The creation of the database involves creating a database server and logging into it as the admin user. Then, for each domain, a database is created. For each database, scripts are written and executed to create Tables, Constraints and Keys. The script also populates data for Master Tables. Then, another script is written and executed that creates users and roles for each of the service applications and assigns the minimum necessary permissions. These scripts are used for database deployments.

Services

The following steps are used to create the services. A solution repository is created for one service and the projects that compose it are added. In those projects, the database connection is configured, and Entity Framework is integrated. The database context, entity classes and data repositories needed by Entity Framework is added. Then, the helper classes for frequent operations and base classes for CRUD operations are implemented. The service's business logic, controllers, and their DTOs are implemented. The calls from the business logic to the repositories and conversions between DTOs and entities are added. The process is repeated to create the other two (2) services.

Client

The following steps are used to create the client. A solution repository is created and the projects that compose it are added. In those projects, the service connections are configured and the http clients with their DTOs are implemented. The helper classes for frequent operations are implemented. The graphical user interface layouts for the main window, lists, views, and form inputs are created and their corresponding events, controllers and model classes are implemented. Validations, calls to the http clients and conversions between DTOs and model classes are added in the controller classes.

Application – User Management

This component is used for managing access to the system through user identities. In the list view, the information is presented in the table. Above the table, the view has an add and a refresh button. The view also has a search bar to filter by the columns shown in the table. In the table, the users can interact with the entries to perform view, edit, delete, and view history actions. Icons were used instead of text, because they are more compact and because images can communicate meaning and capture user attention better than words. The add and edit functions allow the user to enter a new user or modify an existing one and assign permissions. The view function displays all the information stored for a user. The history function displays a list of changes made to the user. Figure 5 presents a screen capture of the User Management list view.

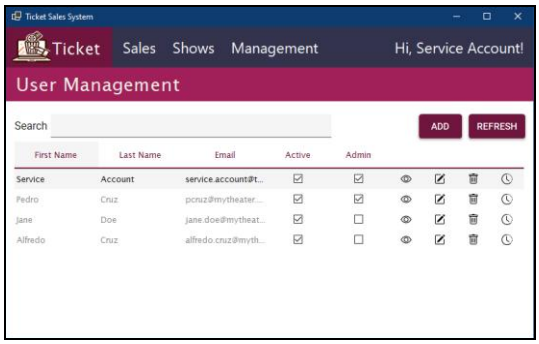

Figure 5
User Management User Interface

Role Management

This component is used for simplifying the management of access to functions with role-based access control. In the list view, the information is presented on the table. Above the table, the view has an add and a refresh button. The view also has a search bar to filter by the columns shown in the table. In the table, the users can interact with the entries to perform view, edit, delete, and view history actions. Icons were used instead of text, because they are more compact and because images can communicate meaning and capture user attention better than words. The add and edit functions allow the user to enter a new role or modify an existing one. The view function displays all the information stored for a role. The historical function displays a list of changes made to the role. Figure 6 presents a screen capture of the Role Management list view.

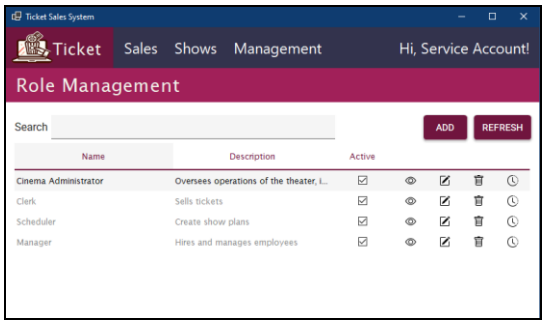

Figure 6
Role Management User Interface

Theater Management

This component is used for maintaining a registry of theaters. In the list view, the information is presented on the table. Above the table, the view has an add and a refresh button. The view also has a search bar to filter by the columns shown in the table. In the table, the users can interact with the entries to perform view, edit, delete, and view history actions. Icons were used instead of text, because they are more compact and because images can communicate meaning and capture user attention better than words. The add and edit functions allow the user to enter a new theater or modify an existing one. The view function displays all the information stored for a theater. The history function displays a list of changes made to the theater. Figure 7 presents a screen capture of the Theater Management list view.

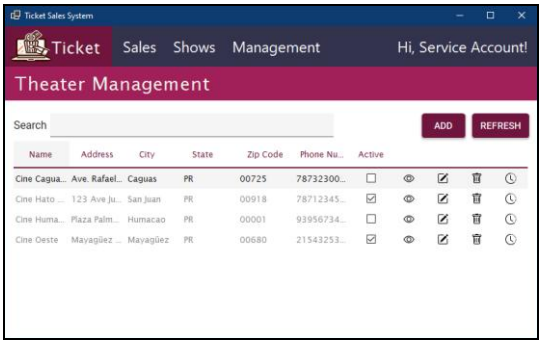

Figure 7
Theater Management User Interface

Room Management

This component is used for maintaining a registry of rooms, their configuration, and their features. In the list view, the information is presented on the table. Above the table, the view has an add and a refresh button. The view also has a search bar to filter by the columns shown in the table. In the table, the users can interact with the entries to perform view, edit, delete, and view history actions. Icons were used instead of text, because they are more compact and because images can communicate meaning and capture user attention better than words. The add and edit functions allow the user to enter a new room or modify an existing one. The view function displays all the information stored for a room. The history function displays a list of changes made to the room. Figure 8 presents a screen capture of the Room Management list view.

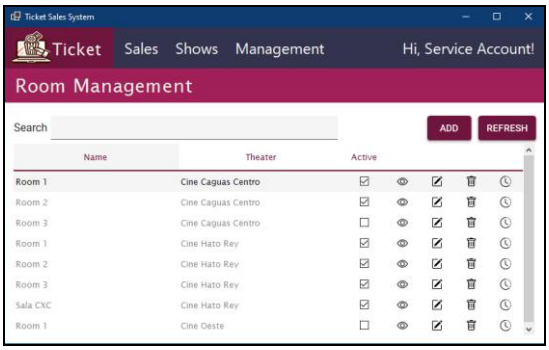

Figure 8
Room Management User Interface

Movies

This component is used for maintaining a registry of movies. In the list view, the information is presented on the table. Above the table, the view has an add and a refresh button. The view also has a search bar to filter by the columns shown in the table. In the table, the users can interact with the entries to perform view, edit, delete, and view history actions. Icons were used instead of text, because they are more compact and because images can communicate meaning and capture user attention better than words. The add and edit functions allow the user to enter a new movie or modify an existing one. The view function displays all the information stored for a movie. The history function displays a list of changes made to the movie. Figure 9 presents a screen capture of the Movies list view.

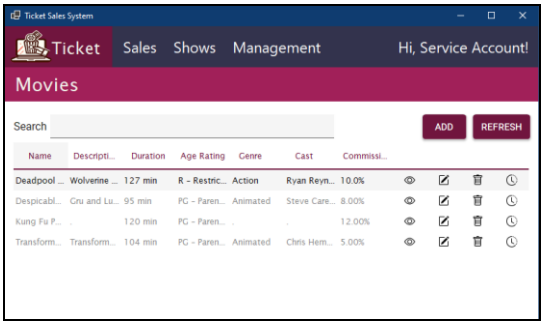

Figure 9
Movies User Interface

Advertisements

This component is used for maintaining a registry of advertisements. In the list view, the information is presented on the table. Above the table, the view has an add and a refresh button. The view also has a search bar to filter by the columns shown in the table. In the table, the users can interact with the entries to perform view, edit, delete, and view history actions. Icons were used instead of text, because they are more compact and because images can communicate meaning and capture user attention better than words. The add and edit functions allow the user to enter a new advertisement or modify an existing one. The view function displays all the information stored for an advertisement. The history function displays a list of changes made to the advertisement. Figure 10 presents a screen capture of the Advertisements list view.

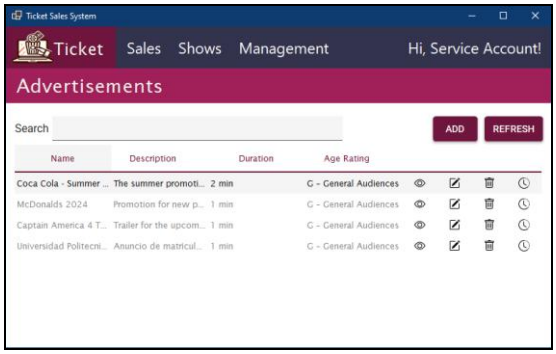

Figure 10
Advertisements User Interface

Shows

This component is used for scheduling the offerings of the theater. In the main view, the information is presented on a timeline limited to a single day and a single theater. The timeline is very wide, so the user may have to scroll. The information presented in the timeline includes the movie name, language configuration, time of showing, and room. The controls above the timeline allow the user to specify the theater and the date for which they want to see the schedule. Above the timeline, the view has an add and a refresh button. Users can interact with the timeline to edit information. The add and edit functions allow the user to enter a new show or modify an existing one. They require users to provide a movie, a room, features for the showing, language configuration, advertisements to play before the movie, and date and time for the show. Depending on the information change, dependent information like availability and duration is recalculated and displayed. Figure 11 presents a screen capture of the Shows main view.

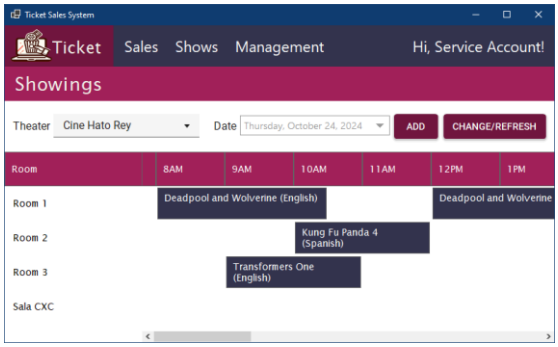

Figure 11
Shows User Interface

Sales

This component is used to display the showing offerings and allow users to purchase tickets. The shows are grouped by movie and language configuration. It is limited to the shows that: are in the selected theater, are scheduled for the current day, and have not ended. Each group is presented as an entry that contains the poster of the movie, basic movie information, and a table containing the showing options of that group. The table contains the times of the showings, and the features assigned. In each table, the users can interact with the listed showing entries to buy tickets. Above the offerings, the user can change the theater and date to see a different offering and a button to cancel a sale and issue a refund. Figure 12 presents a screen capture of the Sales main view.

The ticket purchase window will ask the user for the number of tickets and seats preferred. The left side of the window contains the checkboxes that represent the seats in the theater. The seats already sold are grayed out. The right side contains the picture box that shows the movie poster, the

name of the movie, the showing time, and the features included. Below the movie information, there are 2 textboxes indicating the number of adult and child tickets. Then, it will ask for a payment method and payment amount. After the transaction is completed, it prints the tickets.

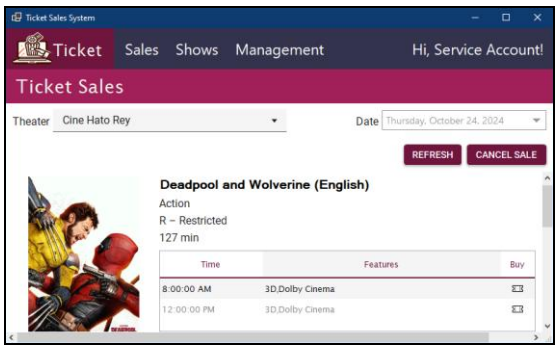

Figure 12
Sales User Interface

CONCLUSION

This paper presented the design and development of a software solution for movie theaters. It focused on the challenges faced daily on the operations of a movie theater, and the financial risks associated with it. It identified the factors that make users want to use an application and that make it have value for a business. These factors served as a basis for designing an application that can help overcome those challenges. The application helps increase efficiency and effectiveness of operations. The application can help optimize the use of each theater room by showing the existing schedule in a timeline and assist room selection for movies by listing their features on the show scheduling window. This helps improve revenue by enabling show diversity and better use of the facilities. The application is also user-friendly. Users can use the program with ease and navigate through the different functionalities quickly. Mistakes and difficulties in any aspect of the operation can cost a lot to the movie theater. It can be a lack of show diversity, slow service, inconvenient showing times, etc. A happy customer will return, but a dissatisfied customer will not visit again.

FUTURE WORK

The system created in this project can perform all the basic operations of a movie theater. These operations generate data that can be used to create added value through:

- Sales reports and dashboard that organize the data available and present it in a way that can be understood. This information has greater business values in industries with high costs, like theaters, to ensure enough profits [8].
- An intelligent schedule assistant. Better schedules result in fewer empty seats and more revenue [9]. When planning for the day, the feature can recommend possible configurations based on specified criteria for each show. The operational data used for the assistant can also be combined with historical public data sets.

REFERENCES

[1] J. Eliashberg, Q. Hegie, J. Ho, D. Huisman, S. J. Miller, S. Swami, C. B. Weinberg & B. Wierenga, "Demand-driven scheduling of movies in a multiplex," in *International Journal of Research in Marketing*, pp. 75-88, 2009.

[2] L. Bo & X. Xinghui, "A Grey Correlation Analysis Approach to Analyze the Demand Factors for Movie Theatre Attendance," in *Applied Mechanics and Materials*, pp. 5181-5184, 2014.

[3] S. Saeed, A. Shaikh, M. A. Memon, M. A. Nizamani, F. A. Abbasi & S. M. R. Naqvi, "Evaluating the Quality of Point of Sale (POS) Software," in *University of Sindh Journal of Information and Communication Technology*, pp. 69-75, 2019.

[4] A. A. Zoltners, P. Sinha, D. Sahay, A. Shastri & S. E. Lorimer, "Practical insights for sales force digitalization success," in *Journal of Personal Selling and Sales Management*, pp. 87-102, 2021.

[5] J. Yablonski, *Laws of UX: Using Psychology to Design Better Products & Services*, O'Reilly Media, Inc., 2020.

[6] M. A. Kabir & B. Han, "An Improved Usability Evaluation Model for Point-of-Sale Systems," in *International Journal of Smart Home*, pp. 269-282, 2016.

[7] Y. Ramos & A. Ojeda, "Point-of-Sales Systems in Food and Beverage Industry: Efficient Technology and Its User Acceptance," in *Journal of Information Sciences and Computing Technologies*, pp. 582-591, 2017.

[8] R. Bennett, “Ticket Sales Forecasting Methods and Performance of UK Theatre Companies,” in *International Journal of Arts Management*, pp. 36-49, 2002.

[9] E. Weiden. (2017, June 2). *A Review of the Challenges of Ticketing and Operations in Various Branches* [Online]. Available: https://www.linkedin.com/pulse/review-challenges-ticketing-operations-various-branches-eli-weiden.