

INTERNATIONAL BURCH UNIVERSITY
FACULTY OF ENGINEERING AND NATURAL SCIENCES
ELECTRICAL AND ELECTRONICS ENGINEERING
INFORMATION TECHNOLOGY



IT 309 SOFTWARE ENGINEERING

PROJECT DOCUMENTATION

Pozorista Application

Prepared by:
Nermin Maslo
Hajrudin Vejzovic
Sadullah-Ado Tahirović

Proposed to:
Nermina Durmić, Assist. Prof. Dr.
Naida Fatic, Teaching Assistant

Table of Contents

1. Introduction	2
1.1. About the Project.....	2
1.2. High-level Plan	2
1.3. Project Requirements.....	7
1.4. UML diagrams	32
2. Project Structure	37
2.1. Technologies	37
2.2. Architectural Pattern.....	38
2.3. Database Entities	39
2.4. Design Patterns	39
2.5. Project Functionalities and Screenshots	41
2.6. Tests.....	46
3. Conclusion.....	46

1. Introduction

This document is for our software project called "Pozorišta" or "Theatre." In today's fast-paced digital age, ensuring a seamless and engaging experience for theatre enthusiasts is paramount. The Theatre app is a comprehensive web application designed to revolutionize the way users discover, book, and enjoy theatrical performances. This project focuses on developing a user-friendly platform that not only simplifies the ticket purchasing process but also enhances the overall theatre-going experience.

GitHub: <https://github.com/neraization/Theater>

1.1. About the Project

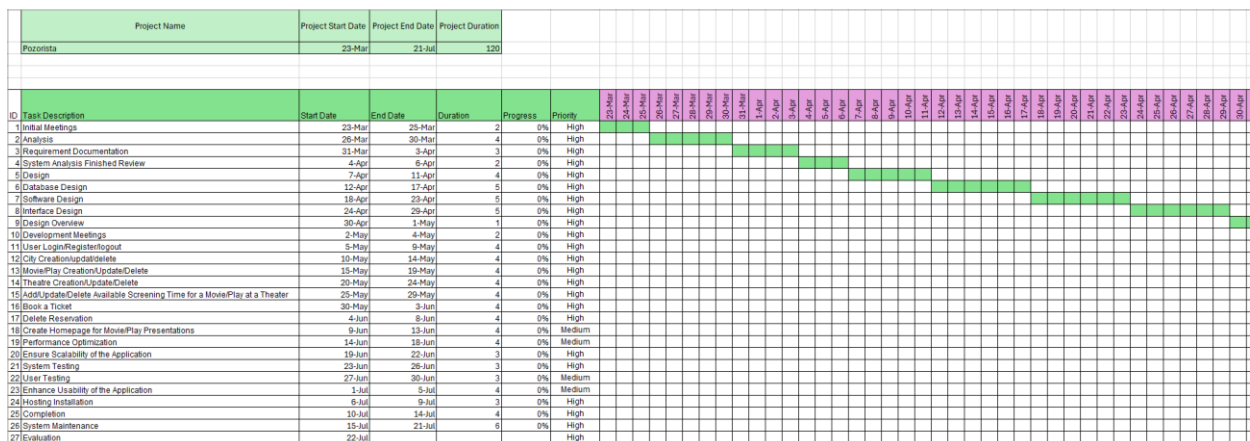
The project is a website where you can find, book, and manage theatre shows in your city. You can sign up, log in, and do things like search for theatres and movies, make reservations, and edit theatre info. You can also order tickets. Admins can post theatres, cities, shows, prices, and seat numbers. The site is made to be easy to use, with features like validation and cookie management for security. For example, if an email or password is incorrect, or if the user does not exist, a popup will show an error message. There are also popups for cookie info and when cookie info is deleted. The project is connected to a database to store data, reservation and each show or theatre has its own ID. You can see the project online on the link

<https://theater-se.onrender.com/>

1.2. High-level Plan

Plan-Driven Development

Gantt chart



Project Name		Project Start Date	Project End Date	Project Duration																																													
Pozinta		23-Mar	21-Jul	126																																													
ID	Task Description	Start Date	End Date	Duration	Progress	Priority	9-Jun	10-Jun	11-Jun	12-Jun	13-Jun	14-Jun	15-Jun	16-Jun	17-Jun	18-Jun	19-Jun	20-Jun	21-Jun	22-Jun	23-Jun	24-Jun	25-Jun	26-Jun	27-Jun	28-Jun	29-Jun	30-Jun	1-Jul	2-Jul	3-Jul	4-Jul	5-Jul	6-Jul	7-Jul	8-Jul	9-Jul	10-Jul	11-Jul	12-Jul	13-Jul	14-Jul	15-Jul	16-Jul					
1	Initial Meetings	23-Mar	25-Mar	2	0%	High																																											
2	Analysis	26-Mar	30-Mar	4	0%	High																																											
3	Requirement Documentation	31-Mar	3-Apr	3	0%	High																																											
4	System Analysis Finished Review	4-Apr	6-Apr	2	0%	High																																											
5	Design	7-Apr	11-Apr	4	0%	High																																											
6	Database Design	12-Apr	17-Apr	5	0%	High																																											
7	Software Design	18-Apr	23-Apr	5	0%	High																																											
8	Interface Design	24-Apr	29-Apr	5	0%	High																																											
9	Design Overview	30-Apr	1-May	1	0%	High																																											
10	Development Meetings	2-May	4-May	2	0%	High																																											
11	User Login/Register/Logout	5-May	9-May	4	0%	High																																											
12	UI Creation/Update/Delete	10-May	14-May	4	0%	High																																											
13	Backend/Server Creation/Update/Delete	15-May	19-May	4	0%	High																																											
14	Theatre Creation/Update/Delete	20-May	24-May	4	0%	High																																											
15	Add/Update/Delete Available Screening Time for a Movie/Play at a Theater	25-May	29-May	4	0%	High																																											
16	Book a Ticket	30-May	3-Jun	4	0%	High																																											
17	Delete Reservation	4-Jun	8-Jun	4	0%	High																																											
18	Create Homepage for Movie/Play Presentations	9-Jun	13-Jun	4	0%	Medium																																											
19	Performance Optimization	14-Jun	18-Jun	4	0%	Medium																																											
20	Ensure Scalability of the Application	19-Jun	22-Jun	3	0%	High																																											
21	System Testing	23-Jun	26-Jun	3	0%	High																																											
22	User Testing	27-Jun	30-Jun	3	0%	Medium																																											
23	Enhance Usability of the Application	1-Jul	5-Jul	4	0%	Medium																																											

3

Risk analysis

1. User Story unclearness:

- a. Risk: Unclear user stories may lead to misunderstandings and misaligned expectations.
- b. Solution: Conduct regular refinement sessions with individuals involved in the project to clarify user stories, elaborate on acceptance criteria, and ensure shared understanding among team members.

2. User Authentication Vulnerabilities

- a. Risk: Weaknesses in user authentication mechanisms may result in unauthorized access or data breaches
- b. Solution: Strengthen the authentication system by implementing measures to handle invalid login attempts effectively, such as clear error messaging for incorrect passwords. Additionally, ensure the security of the database through proper configuration and access controls. Regularly review and update security measures to address potential vulnerabilities and protect against unauthorized access.

3. Data Validation Issues:

- a. Risk: Lack of proper data validation leading to security vulnerabilities such as injection attacks or data corruption.
- b. Solution: Apply strict input validation and sanitation techniques to prevent injection attacks and ensure data integrity. Implement client-side and server-side validation checks for all user inputs.

4. Cross-Browser Compatibility:

- a. Risk: Incompatibility issues with different web browsers leading to inconsistent user experience.
- b. Solution: Perform comprehensive testing across various web browsers and devices to identify and address compatibility issues. Utilize responsive design principles and CSS frameworks to ensure consistent user experience across different platforms.

5. Performance Bottlenecks:

- a. Risk: Performance issues impacting system responsiveness and scalability.
- b. Solution: Conduct performance testing to identify bottlenecks and optimize code, database queries, and server configurations. Implement caching mechanisms to improve response times and scalability.

6. Database Scalability:

- a. Risk: Inability of the database to handle increasing data volumes and user load.
- b. Solution: Design databases with scalability in mind, employing techniques such as sharding, replication, and vertical/horizontal partitioning. Regularly monitor database performance and scale resources as needed to accommodate growing data volumes.

7. Insecure Data Storage:

- a. Risk: Weaknesses in data storage methods may result in unauthorized access or data breaches.
- b. Solution: Ensure the use of secure storage practices. Implement access controls, maintain audit

logs, and conduct routine security assessments to identify and address vulnerabilities in data storage systems.

8. Inadequate Error Handling:

- a. Risk: Insufficient error handling may cause system instability or unexpected behavior.
- b. Solution: Introduce robust error handling mechanisms across the application to gracefully manage exceptions, record errors for troubleshooting, and offer informative error messages to users. Perform comprehensive testing to detect and resolve potential error scenarios.

9. Inadequate Backup and Recovery:

- a. Risk: Lack of proper backup and recovery procedures leading to data loss or system downtime.
- b. Solution: Implement automated backup procedures with regular schedules, offsite storage, and versioning. Test backup and recovery processes regularly to ensure data integrity and minimize downtime in case of system failures or data loss.

10. Regulatory Compliance:

- a. Risk: Failure to comply with legal or regulatory requirements resulting in legal penalties or reputational damage.
- b. Solution: Keep abreast of pertinent regulations and compliance mandates within the project's domain. Set up compliance checkpoints, conduct regular audits, and engage legal counsel to ensure adherence to relevant laws and standards.

11. User Interface Usability:

- a. Risk: Poor user interface design leading to usability issues and user dissatisfaction.
- b. Solution: Conduct usability testing with target users to identify usability issues and gather feedback for interface improvements. Follow established design principles, conduct heuristic evaluations, and iterate on designs based on user feedback.

12. Resource Constraints:

- a. Risk: Limited resources (time, budget, or manpower) hindering project progress or quality.
- b. Solution: Prioritize tasks based on business value and resource availability. Consider outsourcing non-critical tasks, hiring additional resources, or reallocating tasks among team members to balance workload and meet project deadlines.

13. Scope Creep:

- a. Risk: Uncontrolled expansion of project scope leading to timeline and budget overruns.
- b. Solution: Define project scope clearly from the outset and obtain stakeholder buy-in. Establish change control processes to evaluate and prioritize new requirements, ensuring that changes align with project objectives and constraints.

14. Communication Breakdowns:

- a. Risk: Ineffective communication among team members leading to misunderstandings or delays.
- b. Solution: Foster open communication channels within the team through regular meetings, status updates, and collaborative tools. Address conflicts promptly, clarify roles and responsibilities, and encourage feedback to improve team dynamics.

15. Change Management:

- a. Risk: Badly handled changes causing problems with project scope, timing, or quality.
- b. Solution: Use a clear process for managing changes to see how they affect the project's scope, timing, and resources. Put important changes first, let everyone involved help decide, and make sure everyone knows about the changes.

16. Training and Support:

- a. Risk: Insufficient training or support resulting in user frustration or inefficiency in system usage.
- b. Solution: Offer thorough training to all team members, especially those with less experience, to ensure proficient use of the system. Develop user manuals and knowledge bases, and provide ongoing assistance to address inquiries and issues after deployment.

17. Testing Environment Limitations:

- a. Risk: Differences between the testing and production environments may result in inaccurate testing outcomes.
- b. Solution: Ensure the testing environment closely resembles the production setup regarding configurations, data, and infrastructure. Utilize automated testing tools, virtualization, and cloud-based testing environments to enhance test coverage and accuracy.

18. Cost Optimization Based on Region and Demonstration:

- a. Risk: Varied costs across regions may impact overall project expenses.
- b. Solution: Evaluate regional costs and adjust spending strategies to optimize the project budget. Demonstrate potential savings and benefits through cost analysis for different regions, and implement customized cost plans for each region to achieve optimal results.

19. Member Quits:

- a. Risk: If a team member leaves the project, it could cause potential disruptions and delays.
- b. Solution: Document each team member's responsibilities and areas of expertise to ease task reassignment if someone leaves. Cross-train remaining team members (each others) on essential tasks and maintain comprehensive documentation, including fair separation tasks of the departing member.

1.3. Project Requirements

Functional and Non-Functional

User Story 1

Title: User Registration	Priority: High	Estimate:
User story As a user I want to be able to register on the application because I want to be able to use the application.		
Acceptance criteria <ul style="list-style-type: none">• When accessing the application for the first time, the user is presented with a registration screen.• The registration screen includes fields for the user to enter their email address, create a password.• The password field includes requirements for minimum length and complexity to ensure security.• The email address provided by the user must be in a valid format.• Error messages are displayed if any registration fields are incomplete or if the email address entered is already in use.• Once registered, the user can log in to the application using their email address and password for subsequent access.		

User Story 2

Title: User Login	Priority: High	Estimate:
User story As a registered user I want to be able to login into the application because I want to be able to use the application.		
Acceptance criteria <ul style="list-style-type: none">• In order to login into the application user has to be registered, its information has to be in the database.• Login page will have email and password inputs and a login button one below another.• Users are required to enter both inputs in order to login.• When the user clicks on the login button the system will check if the user's email address exists in the database. If it does it will check if the password for that email address is the same as the inputted password.• If both email and password are correct the user will be redirected to the Home page.• If the email address or password don't match one in the database the alert message will be displayed and the user will not be able to login.		

User Story 3

Title: User Logout	Priority: High	Estimate:
User story As a registered user, I want to be able to logout from my account so that I can securely end my session and protect my privacy.		
Acceptance criteria <ul style="list-style-type: none">• Develop a RESTful endpoint specifically for logout functionality, accessible via a URL such as "/api/logout".• Ensure that the logout endpoint is protected and can only be accessed by authenticated users.• Validate the user's authentication token or session credentials before processing the logout request.• Upon receiving a valid logout request, immediately invalidate the user's session token or clear any session-related data stored in the backend.• Return an HTTP status code 200 (OK) upon successful logout.• If the logout attempt fails due to an invalid session or other technical issues, return an appropriate error status code (e.g., 401 Unauthorized, 500 Internal Server Error) along with a meaningful error message.		

User Story 4

Title: City Creation	Priority: High	Estimate:
User story As an administrator of the application, I want to be able to add new cities to the system, so that users can search for movie showtimes and make reservations in additional locations.		
Acceptance criteria <ul style="list-style-type: none">• The city creation form should include fields for entering the city name• The city name field should not allow duplicates to prevent the creation of multiple cities with the same name.• Upon submission of the city creation form, the system should validate the entered information to ensure accuracy and completeness.• Once the city is successfully added to the system, it should be immediately available for users to select when searching for movie showtimes or making reservations• The new city should appear in the application's list of available locations		

User Story 5

Title: City Update	Priority: High	Estimate:
User story As an administrator of the application, I want to be able to update existing cities from the system, so that I am sure that the list of available locations remains accurate and up-to-date.		
Acceptance criteria <ul style="list-style-type: none">• Administrators should be able to select a city from the list of existing cities to update.• Upon submission of the city update form, the system should validate the entered information to ensure accuracy and completeness.• Once the city is successfully updated in the system, the changes should be reflected immediately for users searching for movie showtimes or making reservations.• Changes to the city details should be logged for audit purposes, documenting the actions taken by administrators and ensuring accountability.		

User Story 6

Title: Delete City	Priority: High	Estimate:
User story As an administrator of the application, I want to be able to delete existing cities from the system, so that I am sure that the list of available locations remains accurate and up-to-date.		
Acceptance criteria <ul style="list-style-type: none">• Administrators should be able to select a city from the list of existing cities to delete.• Deleted cities should no longer appear in the application's list of available locations, ensuring that users cannot select them for searching or reservation purposes.• Changes to the list of cities, including deletions, should be logged for audit purposes, documenting the actions taken by administrators and ensuring accountability.		

User Story 7

Title: Create a Play/Movie	Priority: High	Estimate:
User story As an administrator of the application, I want to be able to add a new movie/play to the database, so that I am sure that users have access to up-to-date movie/play listings.		
Acceptance criteria <ul style="list-style-type: none">• The movie/play creation form should include fields for entering the movie/play name• The movie/play name field should not allow duplicates to prevent the creation of multiple records with the same name.• Upon submission of the movie/play creation form, the system should validate the entered information to ensure accuracy and completeness.• Once the movie/play is successfully added to the system, it should be immediately available for users to select when searching for movie showtimes or making reservations• The new movie/play should appear in the application's list of available movies/plays		

User Story 8

Title: Movie/Play Update	Priority: High	Estimate:
User story As an administrator of the application, I want to be able to update existing movies/plays from the system, so that I am sure that the list of available movies/plays remains accurate and up-to-date.		
Acceptance criteria <ul style="list-style-type: none">• Administrators should be able to select a movie/play from the list of existing movies/plays to update.• Upon submission of the movie/play update form, the system should validate the entered information to ensure accuracy and completeness.• Once the movie/play is successfully updated in the system, the changes should be reflected immediately for users searching for movie showtimes or making reservations.• Changes to the movie/play details should be logged for audit purposes, documenting the actions taken by administrators and ensuring accountability.		

User Story 9

Title: Delete movie/play	Priority: High	Estimate:
User story As an administrator of the application, I want to be able to delete existing movies/plays from the system, so that I am sure that the list of available movies/plays remains accurate and up-to-date.		
Acceptance criteria <ul style="list-style-type: none">• Administrators should be able to select a movie/play from the list of existing movies/plays to delete.• Deleted movie/play should no longer appear in the application's list of available movies/plays, ensuring that users cannot select them for searching or reservation purposes.• Changes to the list of movies/plays, including deletions, should be logged for audit purposes, documenting the actions taken by administrators and ensuring accountability.		

User Story 10

Title: Create New Theatre	Priority: High	Estimate:
User story As an administrator of the application, I want to be able to add a new theatre listing to the system, so that I am be able to provide users with additional options for movie screenings in different cities.		
Acceptance criteria <ul style="list-style-type: none">• Administrators should be able to enter the name of the theatre, ensuring it is unique within the system.• The administrator should specify the number of seats available in the theatre.• The administrator should set the price for tickets at the theatre.• Administrators should select the city where the theatre is located from a pre-existing list of cities within the application.• Once the theatre is successfully added to the system, it should be immediately available for users to view and search.• Changes to the theatre listings, including additions, should be logged for audit purposes, documenting the actions taken by administrators and ensuring accountability.		

User Story 11

Title: Update Existing Theatre Details	Priority: High	Estimate:
User story As an administrator of the application, I want to be able to update the details of an existing theatre in the system, so that I am sure that the information remains accurate and up-to-date.		
Acceptance criteria <ul style="list-style-type: none">• Administrators should be able to select the theatre they want to update from a list of existing theatres.• The interface should allow administrators to modify the name of the theatre, ensuring it remains unique within the system.• Administrators should be able to update the number of seats available in the theatre.• Administrators should be able to change the price for tickets at the theatre.• Administrators should have the option to change the city where the theatre is located from a pre-existing list of cities within the application.• Once the theatre details are successfully updated in the system, the changes should be reflected immediately for users to view and search.• Changes to the theatre details should be logged for audit purposes, documenting the actions taken by administrators and ensuring accountability.		

User Story 12

Title: Delete Existing Theatre	Priority: High	Estimate:
User story As an administrator of the application, I want to be able to delete an existing theatre from the system, so that I am sure that outdated or irrelevant listings are removed.		
Acceptance criteria <ul style="list-style-type: none">• Administrators should be able to select the theatre they want to delete from a list of existing theatres.• The system should remove the selected theatre from the database.• The deleted theatre should no longer appear in the list of available theatres for users to view or search.• Once deleted, any associated showtimes or bookings for the theatre should also be removed from the system.• Changes made to the theatre listings, including deletions, should be logged for audit purposes, documenting the actions taken by administrators and ensuring accountability.		

User Story 13

Title: Add Available Screening Time for a Movie/Play at a Theater	Priority: High	Estimate:
User story As an administrator of the application, I want to be able to add available screening times at a particular theater, so that I am sure that users have options to attend screenings at convenient times.		
Acceptance criteria <ul style="list-style-type: none">• Administrators should choose the specific theater where the screenings will be held from a list of available theaters.• The interface should allow administrators to specify the time for each available screening at the selected theater.• Once the available screening times are successfully added in the system, they should be immediately available for users to view and book tickets.• Changes to the available screening times should be logged for audit purposes, documenting the actions taken by administrators and ensuring accountability.		

User Story 14

Title: Update Available Screening Time for a Movie/Play at a Theater	Priority: High	Estimate:
User story As an administrator of the application, I want to be able to update the available screening times at a particular theater, so that I am sure that users have access to accurate and up-to-date screening options.		
Acceptance criteria <ul style="list-style-type: none">• Administrators should choose the specific theater where the screenings are scheduled to take place from a list of available theaters.• The interface should display the existing available screening times for the selected theater, allowing administrators to make modifications• Administrators should be able to edit the time for existing screening times, ensuring flexibility to adjust schedules as needed.• Once the available screening times are successfully updated in the system, they should be immediately available for users to view and book tickets.• Changes to the available screening times should be logged for audit purposes, documenting the actions taken by administrators and ensuring accountability.		

User Story 15

Title: Delete Available Screening Time for a Movie/Play at a Theater	Priority: High	Estimate:
User story As an administrator of the application, I want to be able to delete available screening times at a particular theater, so that I am sure that outdated or unnecessary screening options are removed.		
Acceptance criteria <ul style="list-style-type: none">• Administrators should choose the specific screening time at a particular theater where the screenings are scheduled.• The interface should display the existing available screening times for the selected theater, allowing administrators to select the screening times they wish to delete.• The system should remove the selected screening times from the database.• Once deleted, the removed screening times should no longer appear in the list of available options for users to view or book tickets.• Changes made to the available screening times, including deletions, should be logged for audit purposes, documenting the actions taken by administrators and ensuring accountability.		

User Story 16

Title: Create Showing for a Movie at a Theater	Priority: High	Estimate:
User story As an administrator of the application, I want to be able to create a showing for a movie/play at a specific theater, specifying the screening time, release date, and end date of the film screening, so that I am sure that users have accurate information about the movie's availability.		
Acceptance criteria <ul style="list-style-type: none">• Administrators should select the movie/play for which they want to create a showing.• Administrators should choose the specific theater where the showing will take place from a list of available theaters.• Administrators should specify the time for the showing at the selected theater.• Administrators should enter the release date and end date of the film screening, ensuring users have information about the duration of the movie's availability at the theater.• Once the showing is successfully created in the system, it should be immediately available for users to view and book tickets.• Changes to the showing details should be logged for audit purposes, documenting the actions taken by administrators and ensuring accountability.		

User Story 17

Title: Book and Pay for Movie/Play Tickets	Priority: High	Estimate:
User story As a user of the application, I want to be able to browse available movie/play showings, select seats, and securely pay for my tickets, so that I am sure that I have a smooth and convenient booking process.		
Acceptance criteria <ul style="list-style-type: none">• As a user, I should be able to browse the list of available movies/plays and showings, filtered by location, date, and movie/play title.• Upon selecting a movie/play showing, I should be presented with a seating chart for the theater, showing available and booked seats.• I should be able to select one or multiple seats for the desired showing from the seating chart.• The system should validate that the selected seats are available and not already booked by other users.• Once seats are selected, I should be prompted to proceed to the payment process.• I should be able to review the selected seats, total ticket price, and payment details before confirming the booking.• Upon successful payment, I should receive a confirmation of my booking via email or within the application.• The booked seats should be marked as reserved in the system to prevent double bookings• After completing the booking, I should have the option to view and manage my reservations within the application.		

User Story 18

Title: Delete Reservation	Priority: High	Estimate:
User story As a user of the application, I want to be able to delete my reservation in case I no longer plan to attend the movie/play, so that I am sure that the seats become available for other users and I can manage my bookings effectively.		
Acceptance criteria <ul style="list-style-type: none">• As a user, I should be able to access a list of my reservations within the application.• Each reservation entry should display relevant details such as the number of reservation and booked seats.• For each reservation, there should be an option or button to delete the reservation.• The system should promptly remove the reservation from the database and mark the previously booked seats as available.• The deleted reservation should no longer appear in the list of my reservations within the application.		

User Story 19

Title: Create Homepage for Movie/Play Presentations and Booking	Priority: High	Estimate:
User story As a user of the application, I want to be presented with a user-friendly homepage where I can easily browse available movies/plays, view upcoming showings, and proceed to book tickets, so that I am sure that I have a seamless a		
Acceptance criteria <ul style="list-style-type: none">• Upon accessing the application, I should be directed to the homepage, prominently featuring a visually appealing layout.• The homepage should display a carousel or grid showcasing featured movies/plays or current promotions to capture my attention.• Each movie/play listing should include relevant details such as the movie/play title, poster image, brief description, release date, and rating.• For each movie/play, I should be able to view available showings, including dates, times, and theaters, directly from the homepage.• The homepage should include clear calls-to-action (CTAs) or buttons to proceed to booking tickets for selected showings.• The booking process should be initiated seamlessly from the homepage, guiding me through selecting seats, entering payment details, and completing the reservation.• The homepage should be mobile-responsive, providing an optimal viewing experience across different devices and screen sizes.• The design of the homepage should prioritize usability and intuitive navigation, making it easy for me to find relevant information and complete actions efficiently.• The homepage layout and content should be dynamic, allowing for updates and adjustments to reflect changes in available movies, promotions, or user preferences over time.		

User Story 20

Title: Performance Optimization	Priority: Medium	Estimate:
User story As a user of the application, I expect the system to perform efficiently and respond quickly to my interactions, even during peak usage periods or when handling large volumes of data, because I want from application to be stable and optimized.		
Acceptance criteria <ul style="list-style-type: none">• The application should load within a reasonable timeframe (e.g., under 3 seconds)• Response times for common actions such as searching for movies, booking tickets should be consistently fast, with minimal latency.• The application's codebase should be optimized for efficiency, with attention to reducing unnecessary database queries, minimizing resource usage, and implementing caching mechanisms where appropriate.		

User Story 21

Title: Ensure Scalability of the Application	Priority: Medium	Estimate:
User story As a system architect, I want to ensure that the application is designed and implemented with scalability in mind, allowing it to handle increased loads and accommodate growing user demands over time, because I want from my application to be scalable.		
Acceptance criteria <ul style="list-style-type: none">• Conduct thorough analysis of the application's architecture and infrastructure to identify potential scalability bottlenecks and areas for improvement.• Implement horizontal scalability by designing the application to support distributed computing and the ability to add or remove resources dynamically based on demand• Implement caching mechanisms at various levels of the application stack to reduce database and server load, improving overall responsiveness and scalability.• Monitor application performance and scalability metrics in real-time using monitoring and alerting tools, allowing proactive identification and resolution of scalability issues.		

User Story 22

Title: Enhance Usability of the Application	Priority: Medium	Estimate:
User story As a user experience designer, I want to improve the usability of the application to ensure that users can easily navigate, understand, and interact with its features, so that I am sure that enhancement will lead to a more enjoyable and efficient user experience.		
Acceptance criteria <ul style="list-style-type: none">• Conduct usability testing with representative users to identify pain points, challenges, and areas for improvement in the current user interface and workflow.• Implement intuitive and consistent navigation patterns throughout the application, ensuring that users can easily find and access desired features and content.• Optimize the layout and design of user interface elements to prioritize important information, minimize clutter, and maintain visual hierarchy.• Ensure responsive design principles are applied to make the application accessible and functional across a variety of devices and screen sizes.		

User Story 23

Title: Unit Testing for Theater Creation POST Method	Priority: Medium	Estimate:
User story As a developer, I want to create unit tests for the POST method used to create a theater in the application's API, so that I am sure that the endpoint functions correctly and handles various scenarios effectively.		
Acceptance criteria <ul style="list-style-type: none">• Verify that the theater creation endpoint returns a 201 status code when provided with valid input data.• Ensure that the response body contains the details of the newly created theater.• Validate that the created theater is correctly persisted in the database.		

User Story 24

Title: Unit Testing for Theater PUT Method	Priority: Medium	Estimate:
User story As a developer, I want to create unit tests for the update method used to modify theater details in the application's API, so that I am sure that the endpoint behaves as expected and handles various scenarios accurately.		
Acceptance criteria <ul style="list-style-type: none">• Verify that the theater update endpoint returns a 200 OK status code when provided with valid input data.• Ensure that the response body contains the updated details of the theater.• Validate that the changes made to the theater are correctly reflected in the database.		

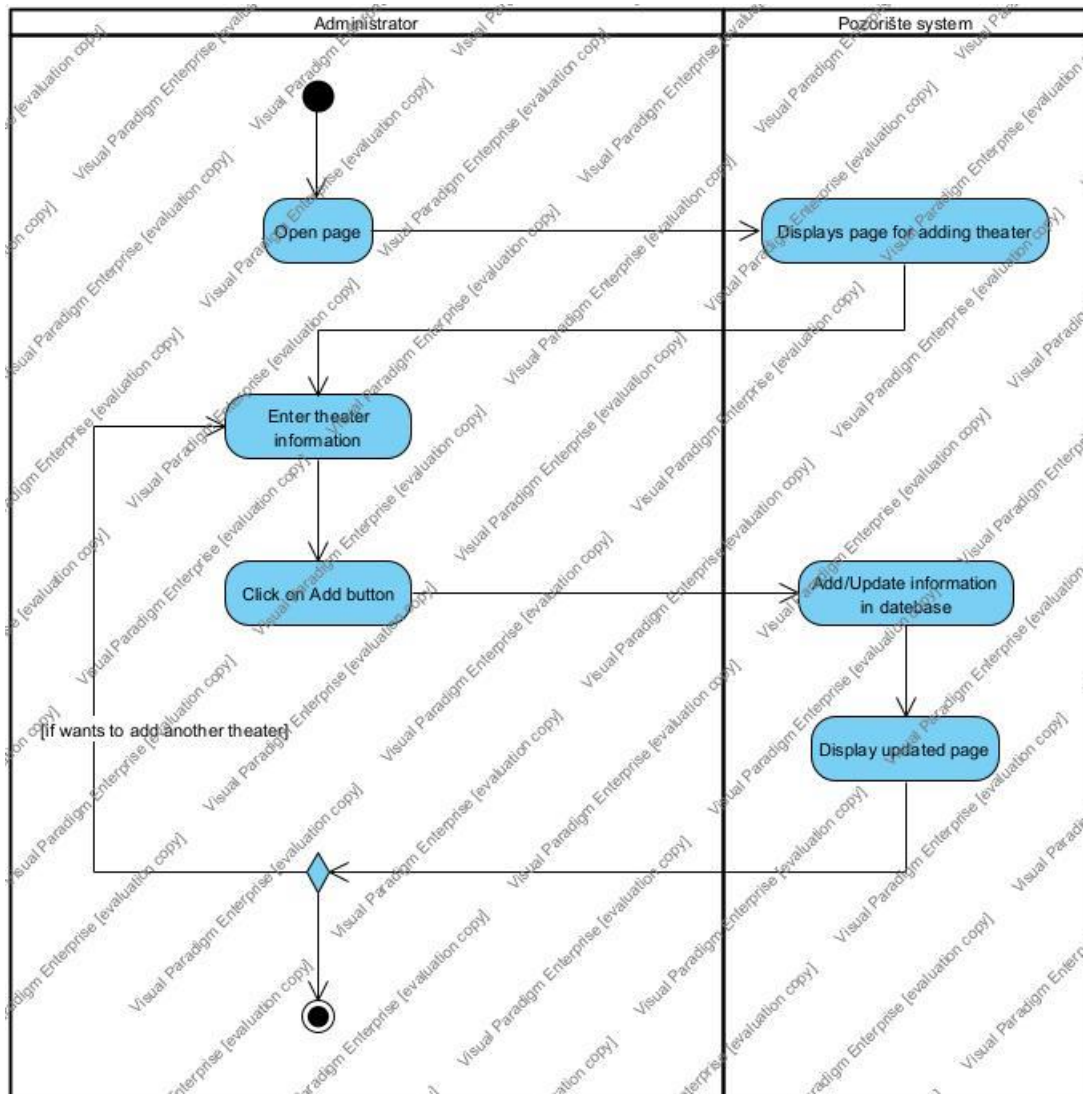
User Story 25

Title: Unit Testing for Theater Delete Method	Priority: Medium	Estimate:
User story As a developer, I want to create unit tests for the deletion method used to remove theaters from the application's API, so that I am sure that the endpoint behaves correctly and handles various scenarios accurately.		
Acceptance criteria <ul style="list-style-type: none">• Verify that the theater deletion endpoint returns a 204 No Content status code when provided with a valid theater ID.• Ensure that the specified theater is successfully removed from the database.• Validate that the response body is empty, as no content is expected to be returned upon successful deletion.		

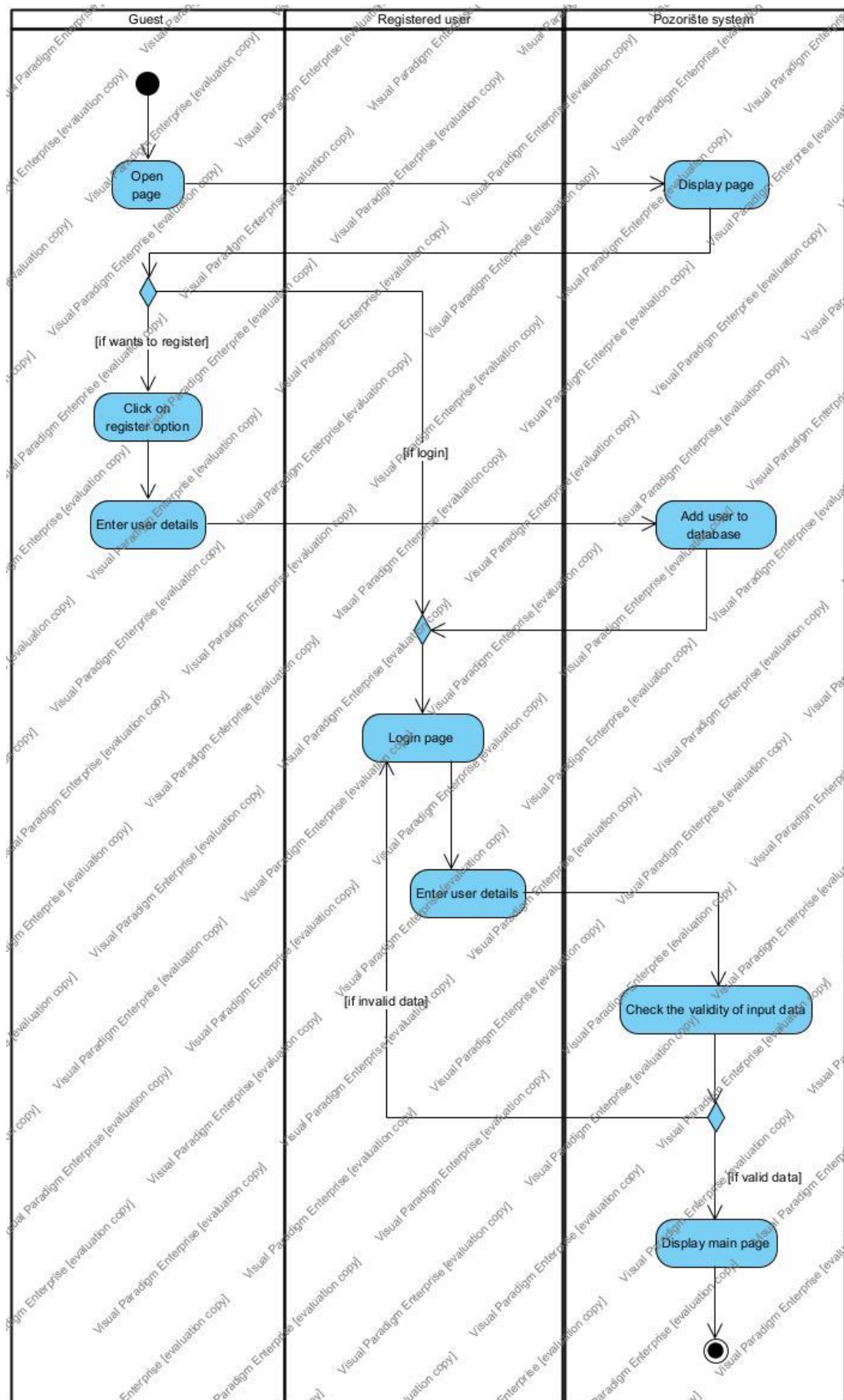
1.4. UML diagrams

Activity Diagrams

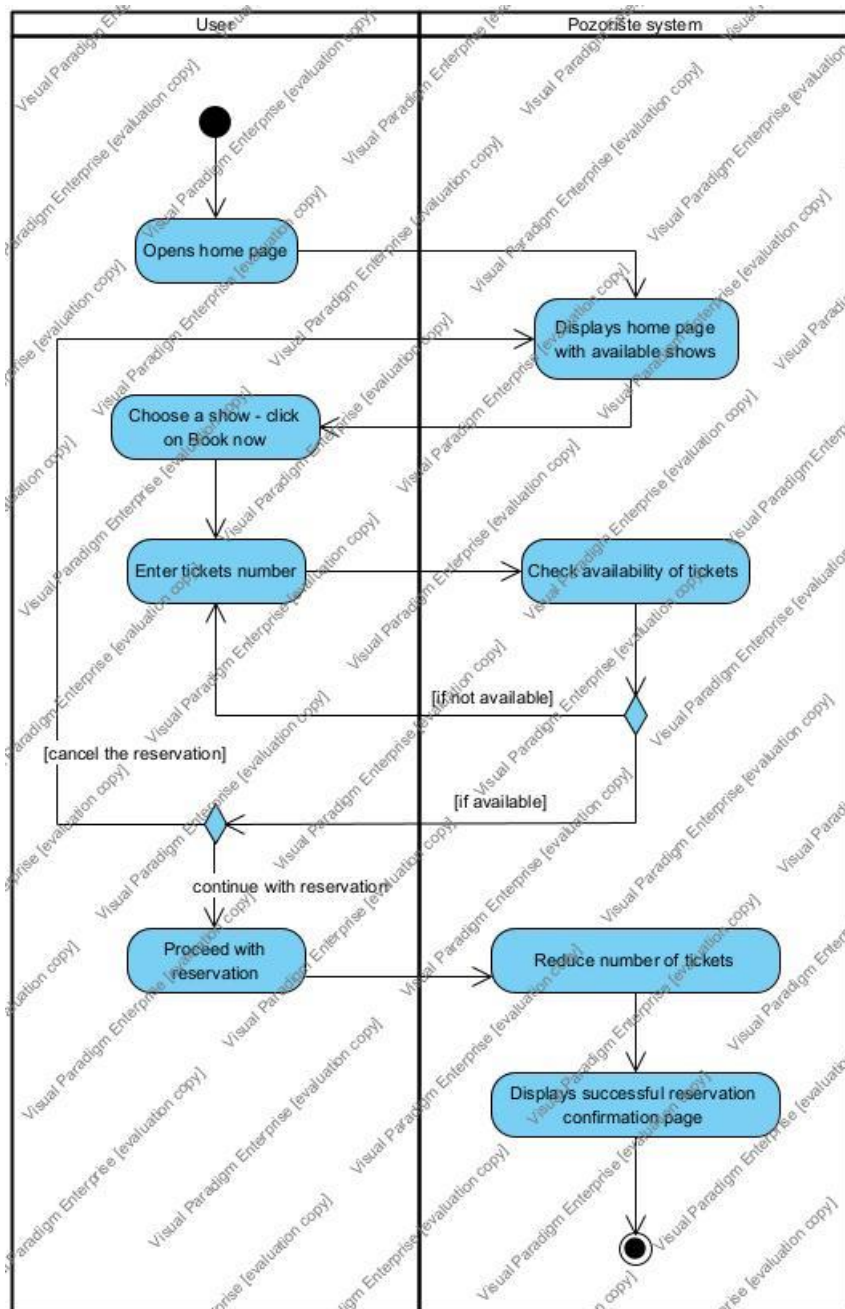
addTheater Activity Diagram



login/register Activity Diagram

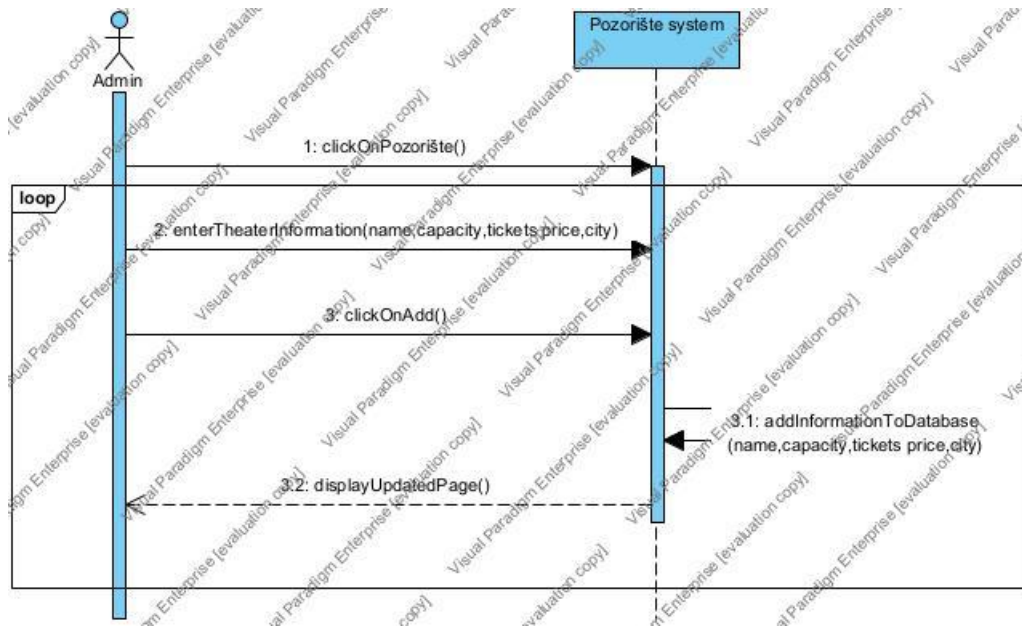


reserveShow Activity Diagram

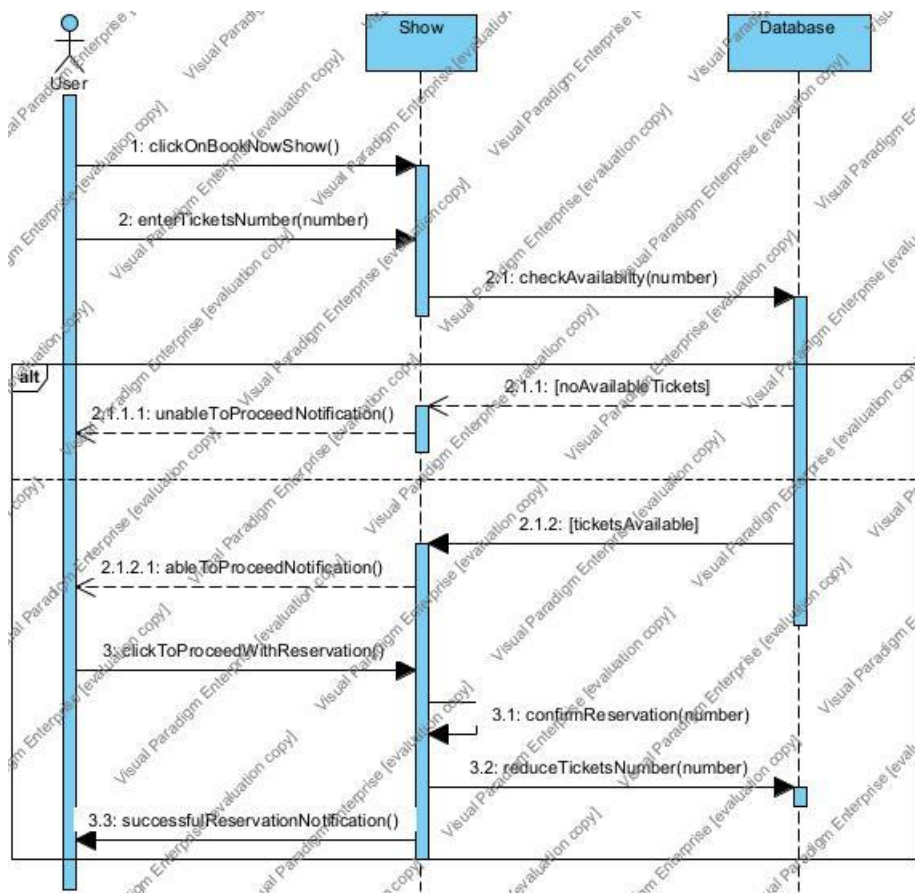


Sequence diagrams

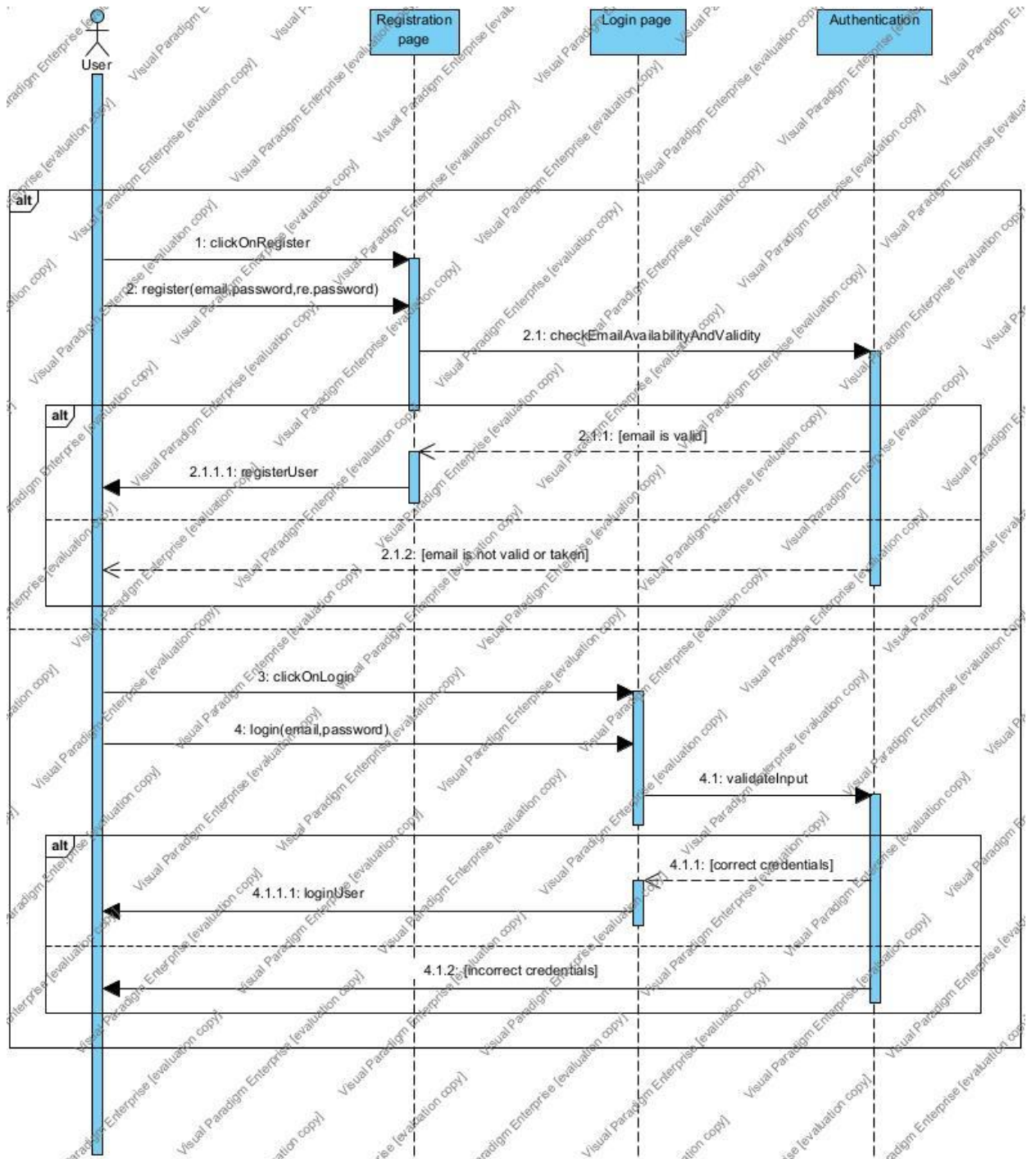
addTheater Sequence Diagram



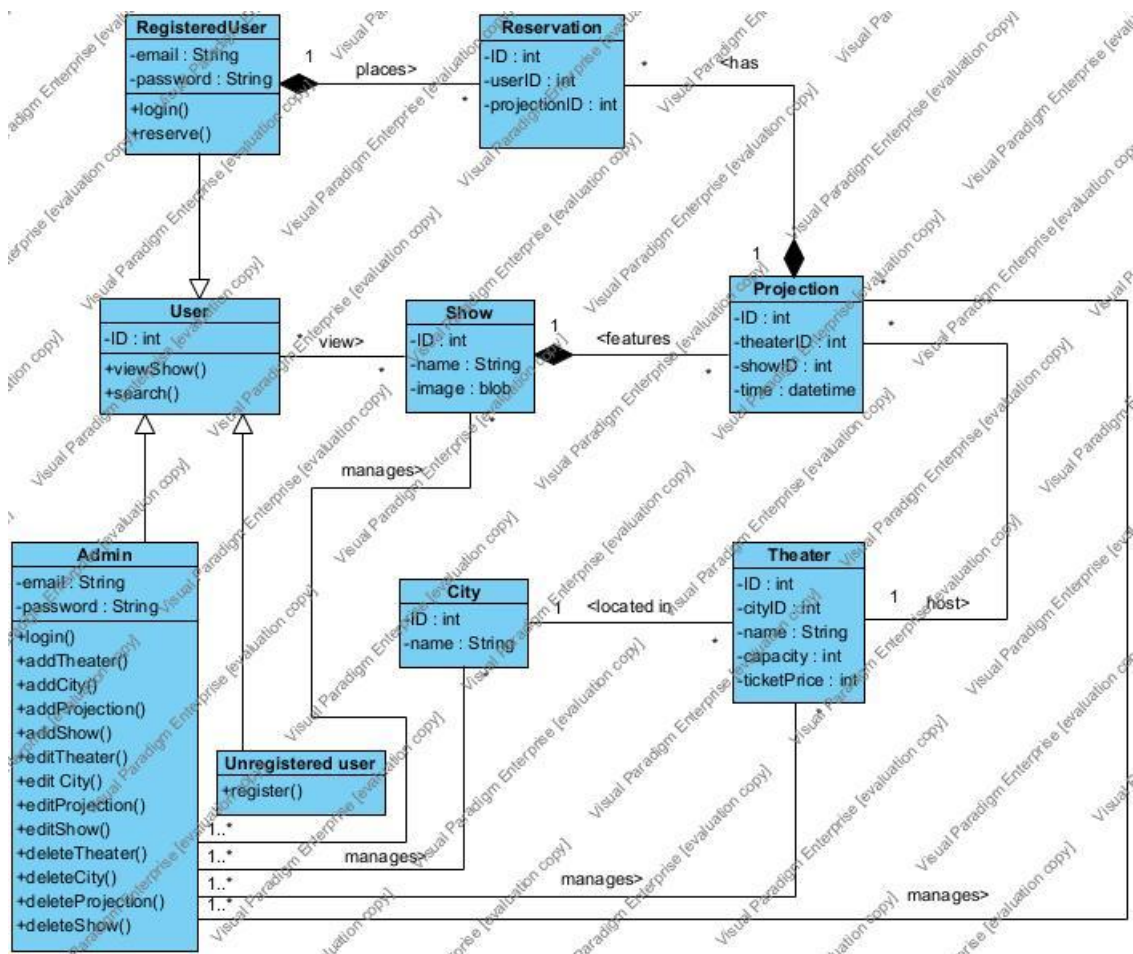
reserveShow Sequence Diagram



login/register Sequence Diagram



Class Diagram



2. Project Structure

2.1. Technologies

Backend:

- **Node.js:** Utilized for building a scalable and efficient server-side application, handling API requests, and managing business logic.

Frontend:

- **Angular:** Employed for creating a dynamic and responsive user interface, leveraging its powerful two-way data binding and component-based architecture.

Database:

- **MongoDB:** Used as a NoSQL database for flexible and efficient data storage, enabling high performance and easy scalability.

Logging and Additional Services:

- **Firebase:** Integrated for comprehensive logging, real-time database features, and user authentication, providing robust backend services and analytics.

Other Tools and Technologies:

- **Express.js:** Used alongside Node.js for routing and middleware functionalities.
- **Mongoose:** An Object Data Modeling (ODM) library for MongoDB, facilitating schema design and data validation.
- **Angular Material:** For building a consistent and aesthetically pleasing user interface with pre-built UI components.
- **Git:** For version control and collaboration.
- **Render.com:** Deployment platforms for hosting the web application, ensuring reliability and scalability.

This combination of technologies ensures a robust, efficient, and user-friendly web application for booking theatre tickets.

2.2. Architectural Pattern

Model-View-Controller (MVC):

The Theatre web application follows the Model-View-Controller (MVC) architectural pattern. This pattern is employed to separate the application into three interconnected components:

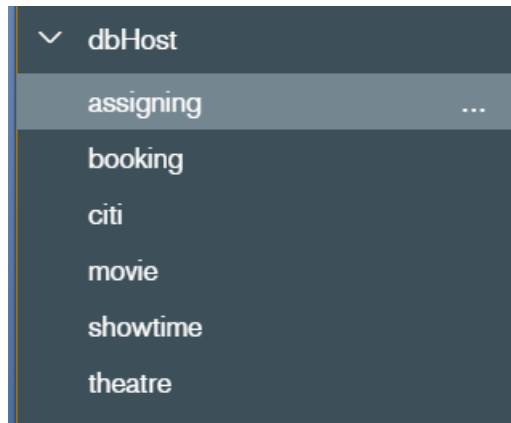
1. **Model:** Manages the data and business logic. It interacts directly with the MongoDB database to handle data operations.
2. **View:** Represents the user interface. Implemented using Angular, it displays data to the user and collects user inputs.
3. **Controller:** Acts as an intermediary between the Model and View. Built with Node.js, it processes incoming requests, manipulates data using the Model, and updates the View.

Why MVC?

- **Separation of Concerns:** MVC divides the application into distinct components, making it easier to manage, maintain, and scale.
- **Reusability and Testability:** Components can be reused and tested independently.
- **Parallel Development:** Different team members can work on the Model, View, and Controller simultaneously.
- **Flexibility and Scalability:** New features can be added with minimal impact on existing components.

2.3. Database Entities

Database: dbHost (Mongo DB)



1. **assigning** - Manages the assignment of movies to theatres and showtimes.
2. **booking** - Stores information about ticket bookings, seat numbers, and order id.
3. **citi** - Contains details of cities where theatres are located, including names and any regional information.
4. **movie** - Holds information about movies such as titles, genres, durations, and descriptions etc.
5. **showtime** - Schedules of movie showtimes, including dates, times, and related theatre details.
6. **theatre** - Details of theatres, including names, locations, seating capacities, and facilities.

2.4. Design Patterns

- **Middleware pattern:** used in the backend, in the file *app/assign-crud.js* on line 136-140, also we used it on in another backend files like: *movie-crud.js*, *city-crud.js*, *bookings-crud.js*, *theatre-crud.js*.

The Middleware Pattern was chosen in this case because it provides a flexible and modular way to handle HTTP requests and responses, including error handling, in a web application.

Why Middleware Pattern:

1. **Separation of Concerns**
2. **Modularity**
3. **Chain of Responsibility**

How It Helped in This Case:

1. **Centralized Error Handling:** The middleware function we provided centralizes the handling of 404 "Not Found" errors. Instead of checking for the existence of resources in multiple places, this middleware catches any request that didn't match a route and generates a 404 error.
 2. **Simplification:** By using middleware for error handling, the main route-handling logic remains clean and focused on its primary purpose, while the error-handling logic is delegated to the middleware.
 3. **Flexibility:** This approach makes it easy to modify how 404 errors are handled without changing the core routing logic. If we later decide to log these errors or send a different response, we can simply update this middleware function.
- **Observer Pattern:** used in the frontend, in the file *public/js/controllers/ MovieCtrl.js*, also we used it on lot of places on frontend side like: *MainCtrl.js*, *AssignCtrl.js* etc.

Example in the Code:

In *MovieCtrl.js* line 15:

- When the movie data is fetched from an API, it is assigned to `$scope.moviList`. Due to the Observer Pattern, AngularJS automatically updates the view to display the new list of movies.
- This automatic update mechanism reduces the need for boilerplate code and allows us to build reactive and interactive user interfaces more efficiently. The Observer Pattern is thus a fundamental aspect of AngularJS that helps us creating dynamic, real-time web applications with ease.

Why the Observer Pattern:

1. **Two-Way Data Binding:** AngularJS's implementation of the Observer Pattern facilitates automatic synchronization between the model (data) and the view (UI). This means that any changes in the model are immediately reflected in the view and vice versa, providing a seamless user experience.
2. **Dynamic UI Updates:** The Observer Pattern allows the application to dynamically update the UI in response to changes in the underlying data model. This is particularly useful in modern web applications where real-time data display is crucial.

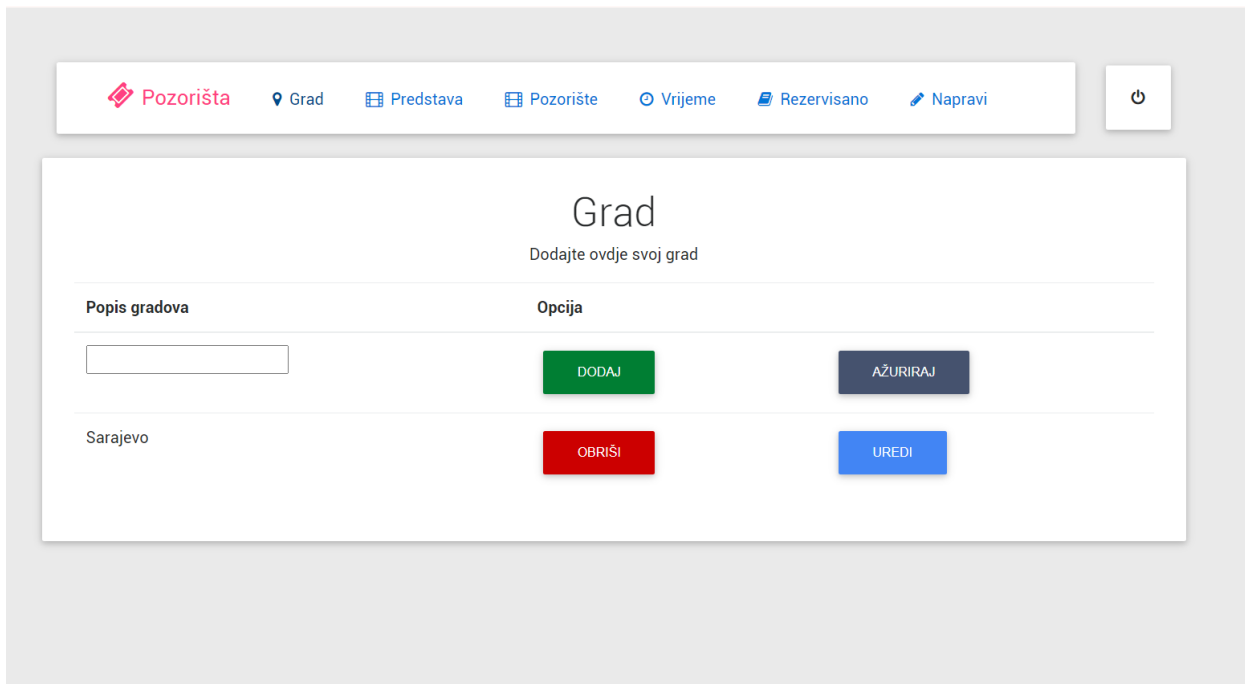
How It Helped in This Case:

1. **Simplified Code:** By using the Observer Pattern, we avoid manually updating the DOM. AngularJS takes care of reflecting the changes in the model to the view, which simplifies the code and reduces the likelihood of errors.
2. **Improved User Experience:** The automatic synchronization ensures that the UI is always in sync with the current state of the data. For instance, when new movie data is fetched and assigned to `$scope.moviList` in *MovieCtrl.js*, the list of movies displayed in the view is updated immediately without any additional code.

3. **Maintainability:** With the Observer Pattern, we can focus on updating the model, knowing that the view will automatically update. This separation of concerns makes the application easier to maintain and extend. Changes to the data handling logic do not require changes to the view update logic, and vice versa.
4. **Consistency:** Using \$scope for binding ensures that data and UI components remain consistent throughout the lifecycle of the application. Any changes made in one part of the application are consistently propagated to all relevant parts.

2.5. Project Functionalities and Screenshots

1. City - City management module utilizes CRUD operations to efficiently handle city data. The application allows administrators to Create new cities, essential for adding venues where theaters are located. Through the Read function, users can easily access information about existing cities, such as their names. Updating city details is streamlined, ensuring accuracy in location information and reflecting any changes in city attributes. Moreover, administrators can Delete cities when necessary, maintaining the database's relevance and reflecting real-world changes in theater locations. This CRUD functionality ensures that our application remains responsive to dynamic urban environments and facilitates seamless management of city data crucial to the overall user experience.



Popis gradova	Opcija
<input type="text"/>	<button>DODAJ</button>
Sarajevo	<button>OBRISI</button>
	<button>AŽURIRAJ</button>
	<button>UREDI</button>

2.Showtime - Showtime management module employs CRUD operations to effectively handle show scheduling and details. The Create functionality allows administrators to add new showtimes, after adding a specific showtime based on the name, we pull data via API call from the IMDB site database and based on that we will display all the necessary data about the showtime. Through Read operations, users can easily browse available showtimes, viewing comprehensive information including seating availability and ticket prices. Updating showtimes ensures accuracy in scheduling and reflects any changes to show details or venue adjustments. Additionally, administrators can utilize the Delete function to remove outdated or canceled showtimes, ensuring the platform remains current and responsive to real-time scheduling needs.

[Grad](#)
[Predstava](#)
[Pozorište](#)
[Vrijeme](#)
[Rezervisano](#)
[Napravi](#)

Kreiraj Predstavu

predstava ID	Opcija	
<input type="text"/>	<button>DODAJ</button>	<button>AŽURIRAJ</button>
Vikings	<button>OBRIŠI</button>	<button>UREDI</button>

3.Theatre - Theatre management module utilizes CRUD operations to efficiently handle theater details and routes. Administrators can employ the Create function to add new theaters, specifying essential details such as the theater's name, seating capacity, ticket prices, and the city where the theater is located. The Read operation enables users to access comprehensive information about theaters, including their seating arrangements and current ticket prices. Updating theater routes allows administrators to modify theater details as needed, ensuring accurate information for users and reflecting any changes in seating availability or pricing. Moreover, administrators can utilize the Delete function to remove theaters from the system, maintaining data relevance and accommodating changes in venue availability. This CRUD functionality ensures that our application provides a user-friendly interface for discovering, managing, and updating theater routes, enhancing the overall booking experience for our users.

[Grad](#)
[Predstava](#)
[Pozorište](#)
[Vrijeme](#)
[Rezervisano](#)
[Napravi](#)

Kreirajte pozorišne zapise

Upravlajte svojim pozorištem!

Naziv pozorišta	Broj sjedišta	Cijena ulaznica	Opcija	
<input type="text" value="Naziv pozorišta"/>	<input type="text" value="Broj sjedišta"/>	<input type="text" value="Cijena ulaznica"/>	<input type="text" value=""/>	<input type="button" value="DODAJ"/> <input type="button" value="AŽURIRAJ"/>
Kamerni Teatar 55	70	25	Sarajevo	<input type="button" value="OBRIŠI"/> <input type="button" value="UREDI"/>

4.Time - Time management module employs CRUD operations to effectively handle show timings and their associated theaters. The Create functionality allows administrators to add new show timings, specifying details such as the time of the showing and the specific theater where the show will be held. Through Read operations, users can easily browse available show timings, viewing comprehensive information that includes the theater name and the exact time of each showing. Updating show timings ensures accuracy in scheduling and reflects any changes to show schedules or theater assignments. Additionally, administrators can utilize the Delete function to remove outdated or canceled show timings, ensuring the platform remains current and responsive to real-time scheduling needs.

The screenshot shows a web application interface for managing show times. At the top, there is a navigation bar with icons and labels for 'Pozorišta', 'Grad', 'Predstava', 'Pozorište', 'Vrijeme', 'Rezervisano', and 'Napravi'. The main heading is 'Vrijeme prikazivanja' with a subtitle 'Možete dodati vremena prikazivanja za sva pozorišta na ovoj stranici'. Below this is a form with three columns: 'Vrijeme prikaza', 'Odaberite pozorište', and 'Opcija'. The 'Vrijeme prikaza' column has a text input field. The 'Odaberite pozorište' column has a dropdown menu. The 'Opcija' column contains two buttons: 'DODAJ' (green) and 'AŽURIRAJ' (dark blue). Below the form, there is a table with two rows. The first row shows '21:00' in the 'Vrijeme prikaza' column and 'Kamerni Teatar 55' in the 'Odaberite pozorište' column. The second row shows '21:00' in the 'Vrijeme prikaza' column and 'Kamerni Teatar 55' in the 'Odaberite pozorište' column. The 'Opcija' column for the second row contains two buttons: 'REMOVE' (red) and 'EDIT' (blue).

Vrijeme prikaza	Odaberite pozorište	Opcija
<input type="text"/>	<input type="text"/>	<button>DODAJ</button> <button>AŽURIRAJ</button>
21:00	Kamerni Teatar 55	<button>REMOVE</button> <button>EDIT</button>

5.Reservations - Reservations route utilizes CRUD operations to manage bookings for specific showtimes. The Read function allows users to view reservations associated with a particular showtime, displaying details such as the reservation ID and the number of seats booked. This functionality provides transparency regarding seat availability and allows users to review their booking details. Administrators can use the Delete operation to cancel reservations if needed, ensuring flexibility and accommodating changes in booking requirements.


The screenshot shows a web application interface for managing reserved shows. At the top, there is a navigation bar with icons and labels for 'Pozorišta', 'Grad', 'Predstava', 'Pozorište', 'Vrijeme', 'Rezervisano', and 'Napravi'. The main heading is 'Rezervisane predstave' with a subtitle 'Sve rezervacije na ovoj stranici'. Below this is a search bar labeled 'Traži narudžbu'. Below the search bar is a table with three columns: 'ID narudžbe', 'Broj sjedala', and 'Opcija'. The table has one row with the values '13627' in the 'ID narudžbe' column and '2' in the 'Broj sjedala' column. The 'Opcija' column contains a button labeled 'OBRIŠI' (red).


ID narudžbe	Broj sjedala	Opcija
13627	2	<button>OBRIŠI</button>


6.Create a new showtime - Showings route leverages comprehensive CRUD operations to manage and schedule showtimes effectively. The Create feature enables administrators to initiate new showings by specifying essential details such as the city, theatre venue, start time, end time, show date, and end date. This functionality ensures flexibility in planning and accommodating various scheduling needs within the application.


Users can utilize the Read function to access existing showings, viewing pertinent information like city, theatre, showtime, and dates, allowing them to stay informed about upcoming performances. The Update capability empowers administrators to modify details of existing showings, ensuring accuracy and reflecting any changes in scheduling or venue assignments.


Furthermore, the Delete operation offers the flexibility to remove outdated or canceled showings from the system, maintaining data relevance and accommodating adjustments in the show schedule over time. This robust CRUD functionality ensures that our application provides a seamless user experience for managing and scheduling theatre showings, catering to both administrators and users seeking up-to-date information and flexible booking options.


 Pozorišta


 Grad


 Predstava

 Pozorište

 Vrijeme

 Rezervisano

 Napravi



Sada možete kreirati predstave u gradu kojem želite!

Odaberite grad

Odaberite pozorište

Odaberite vrijeme prikazivanja

Odaberi predstavu

Datum izlaska

mm/dd/yyyy

Datum završetka

mm/dd/yyyy

DODAJ

AŽURIRAJ

Grad	Pozorište	Vrijeme prikaza	Predstava	Datum prikazivanja	Datum završetka	Opcija
Sarajevo	Kamerni Teatar 55	21:00	Vikings	6/13/2024	6/20/2024	<div>OBRIŠI</div> <div>UREDI</div>

7.Home page - In our Theatre Ticket Booking application, the homepage serves as a central hub where users can easily access information about currently playing showtimes, search for specific showtimes, and proceed to book tickets seamlessly. The homepage prominently displays showtimes that are currently playing, providing users with instant visibility into ongoing performances and their respective details such as title, theatre, and showtime.

Additionally, the search functionality allows users to find specific showtimes by entering criteria such as show title, city, or date, enhancing convenience and enabling users to quickly locate desired performances.

For booking tickets, the homepage offers intuitive pathways where users can click through to view more details about a particular showtime and proceed to reserve seats. This streamlined process ensures that users can easily navigate from discovering showtimes to completing their ticket bookings efficiently.

Overall, the homepage of our application serves as a user-friendly interface that caters to both casual browsers looking for current showtimes and committed users seeking to book tickets, providing a seamless experience from exploration to reservation.



2.6. Tests

Location and Type of Tests

- **tests/apiUnitTest.spec.js:** This file contains Jest tests using Supertest to test the API endpoints defined in `app/movie-crud.js`. These tests validate the behavior of endpoints such as fetching movies, adding movies, updating movies, and deleting movies.
- **Types of Tests:** Primarily unit tests using Jest and Supertest.

Test Details

- **Fetch All Movies:** Tests the `GET /getMovie` endpoint to ensure it returns the expected list of movies.
- **Fetch Movie by ID:** Tests the `GET /getMovie/:id` endpoint to verify that it retrieves a specific movie based on the provided ID.
- **Add a New Movie:** Tests the `POST /addMovie` endpoint to confirm that a new movie can be successfully added to the database.
- **Delete a Movie:** Tests the `DELETE /deleteMovie/:id` endpoint to check if a movie can be deleted by its ID.
- **Update a Movie:** Tests the `PUT /updateMovie/:id` endpoint to ensure that a movie can be updated with new information.

3. Conclusion

In conclusion, the "Pozorišta" project is a good theatre booking site. We followed the project requirements and added many useful features. You can easily book tickets, and admins can manage theatres, cities, shows, prices, and seat numbers. We are happy with the project, especially the easy-to-use interface. But there are things we can improve, like adding more theatres and shows, and better search options. One hard part was making sure the data validation and security worked well, but we solved it with testing. The database connection helps store all the data, IDs, shows and theatres... We plan to make the site better and add new features as this project grows into reality