Software Architecture Document

A. Nasim Adabi
ID : 40079444
E-mail: nasim.adabi@gmail.com

B. Mahshad Saghaleini
ID: 40058409
E-mail: mahshad.saghaleini@gmail.com

C. Nandini Bandlamudi
ID: 40105415
E-mail: bandlamudi.nandini@gmail.com

D. Venkata Pavan Kumar Reddy Ravi
ID: 40083392
E-mail: pavan.03121996@gmail.com

E. Hina Masood Ahmed
ID: 40076287
E-mail: m.hinathahseen@gmail.com

F. Swetha Chenna
ID: 40092019
E-mail: swethachenna2018@gmail.com

G. Kiranmayie
ID: 40092284
E-mail: 2809kiran@gmail.com

H. Sahana Shankar
ID: 40092026
E-mail: sahana15shankar@gmail.com

Abstract:

The system is a web-based application where one can buy tickets with just one click go. An internet user can buy tickets. Also, in the system, if the customer wishes to cancel his tickets he will be given a confirmation details regarding his cancellations. As the customer buys tickets online through his credit card. All the customers have to register and become a member before buying tickets so that he faces no problem while accessing the website.

Introduction

In this document you can find the detailed description for architecture of our Online Movie Ticket Booking system.

Purpose

This document is created to facilitate comprehension of our software which is online movie ticket purchasing system. Different views that have been proposed in this document can assist various groups in the company to have a better understanding of the system and its structure in a short time. Logical View in this document presents the functionality of this software and it can be proposed to the end user to make sure whether it fulfills their requirements or not. Development view in this document demonstrates static organization of this software. It's useful for developers to attain a better understanding of the system structure.Process view in this document displays behaviour of the system at run time and how system maintain reliability during the ticket purchasing process. Physical view in this document describes the hardware configuration required for this software to be implemented with the highest availability and it can be most useful for people at the company who are providing hardware infrastructure for the system.

Scope

This document describes how users can buy movie tickets online. This would include payment and seat selection processes as well. This program only supports one cinema with different salon and movie numbers. It doesn't support multiple cinemas.

Definitions, acronyms, and abbreviations

UML: Unified Modeling Language

SAD: Software Architecture Document

Architectural representation

In order to describe our application's architecture, we have used 4+1 view which contains: Logical View, Process View, Development View, Deployment View and User Stories. We also have displayed our data model to store users' data.

Logical view:

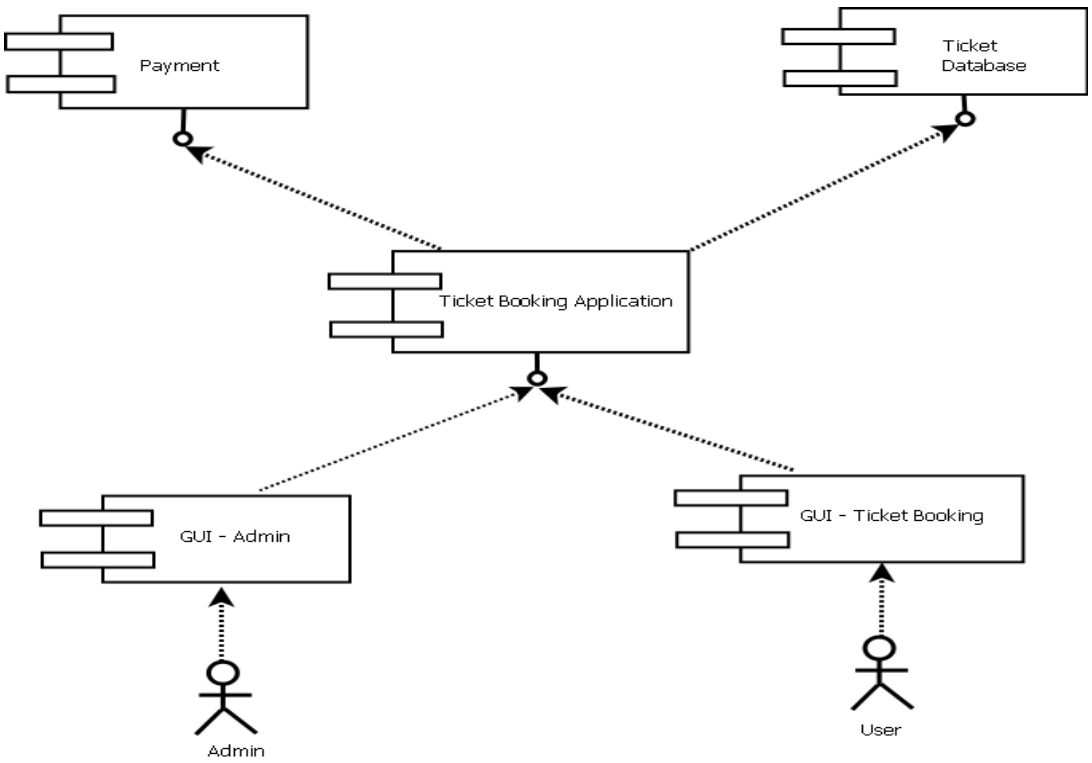

Figure 1. Component Diagram

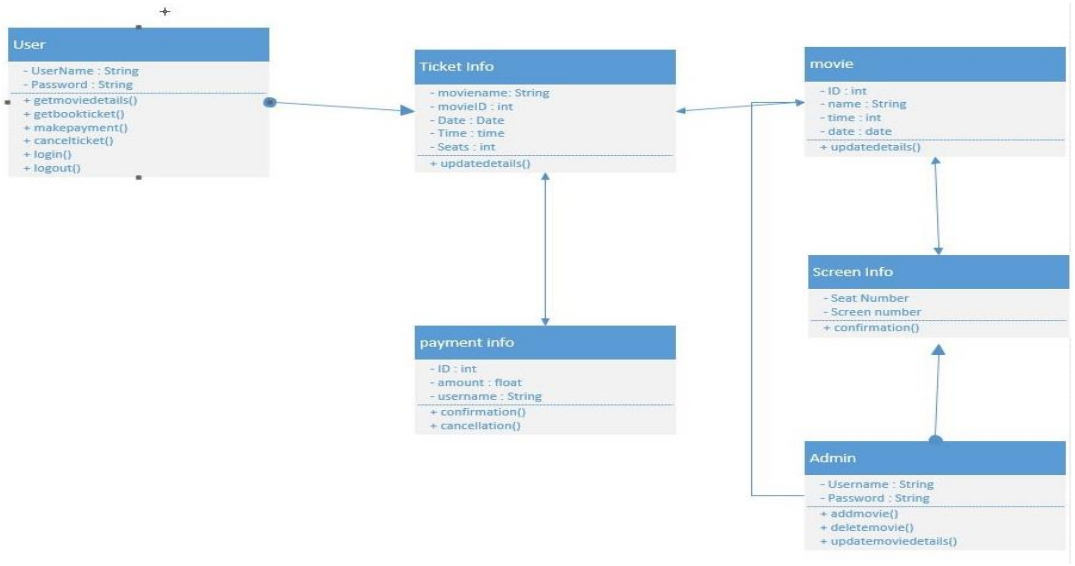

Figure 2. Class diagram

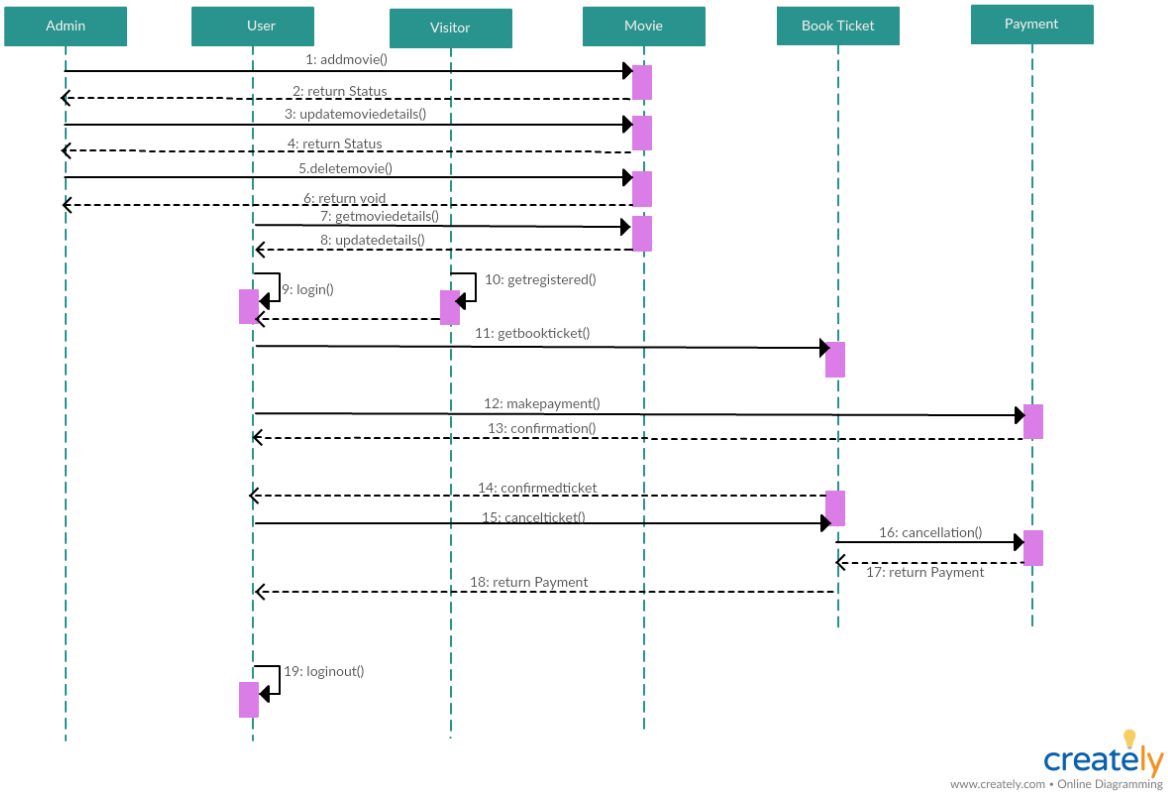

Figure 3. Sequence Diagram

Development view:

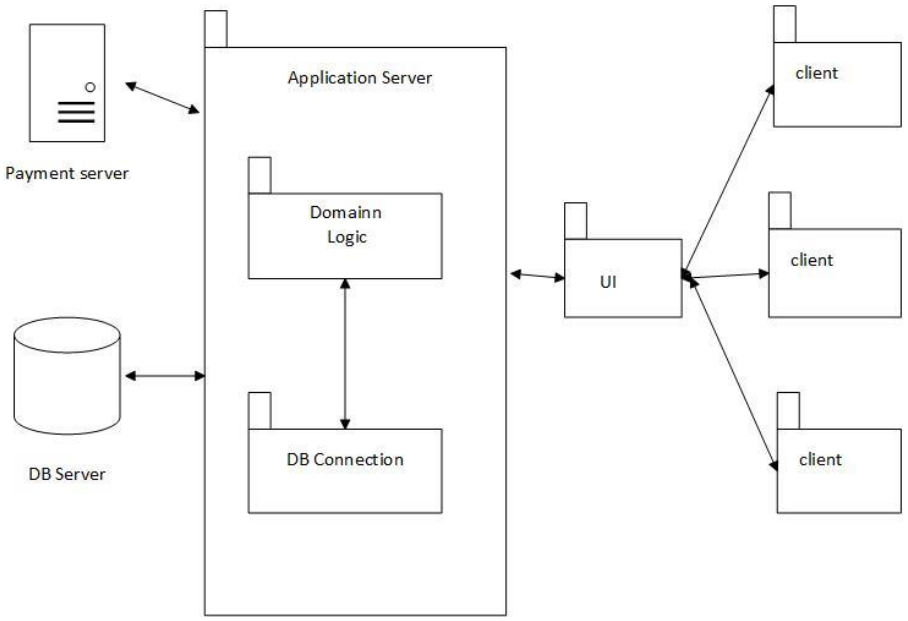

Figure 4. Package Diagram

Process view:

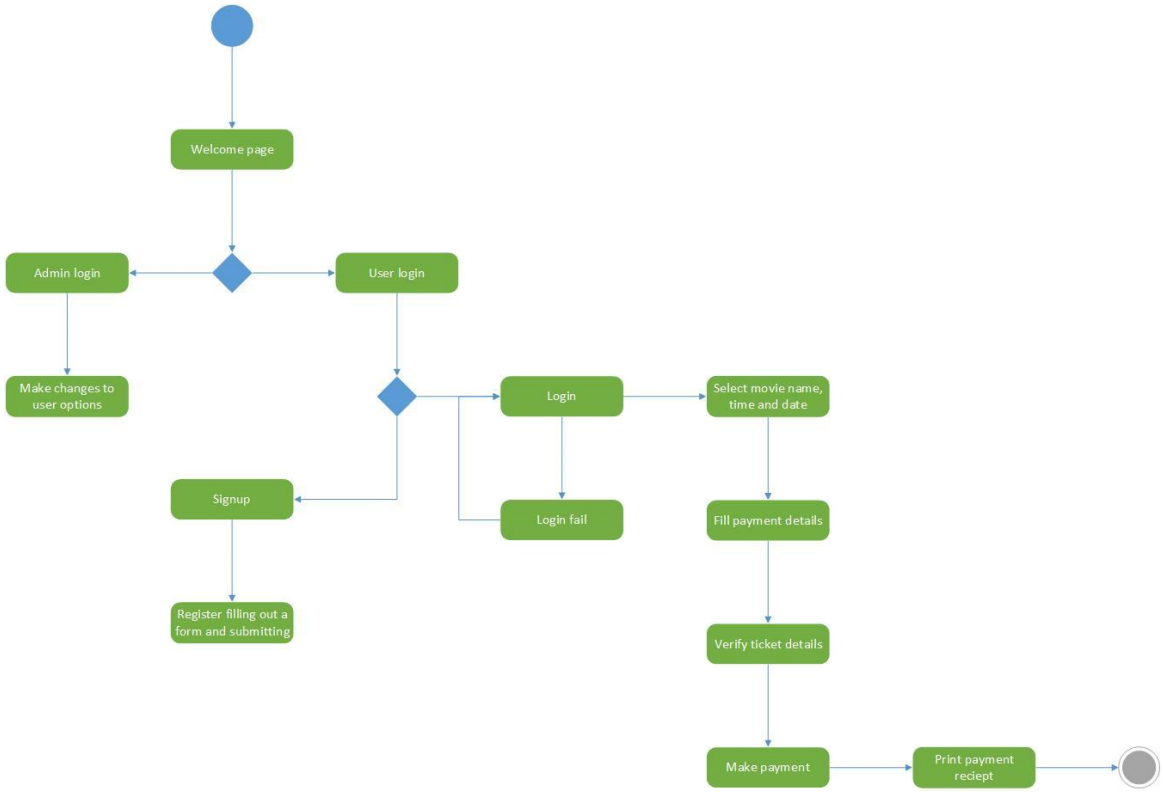

Figure 5. Activity Diagram

Physical view:

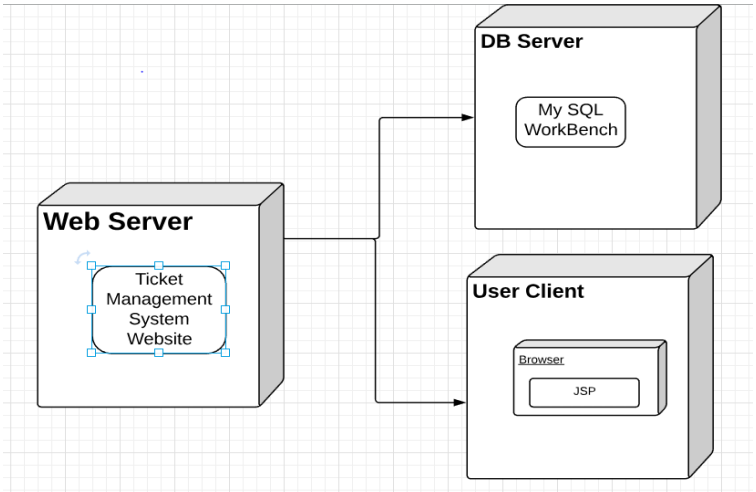

Figure 6. Deployment Diagram

Use case view:

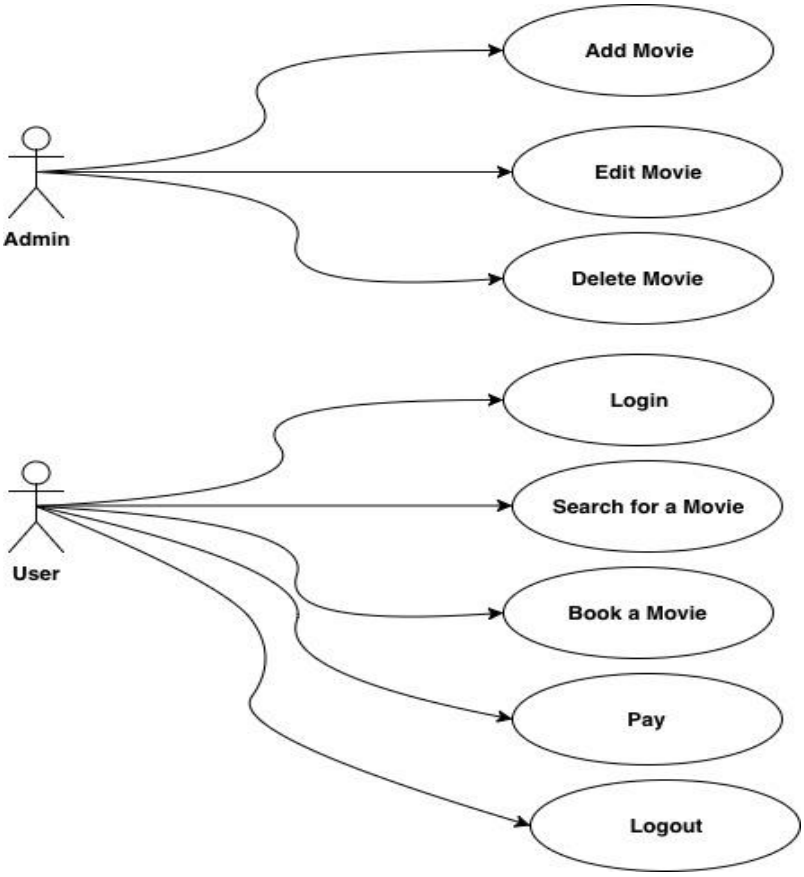

Figure 7. Use Case

Data view:

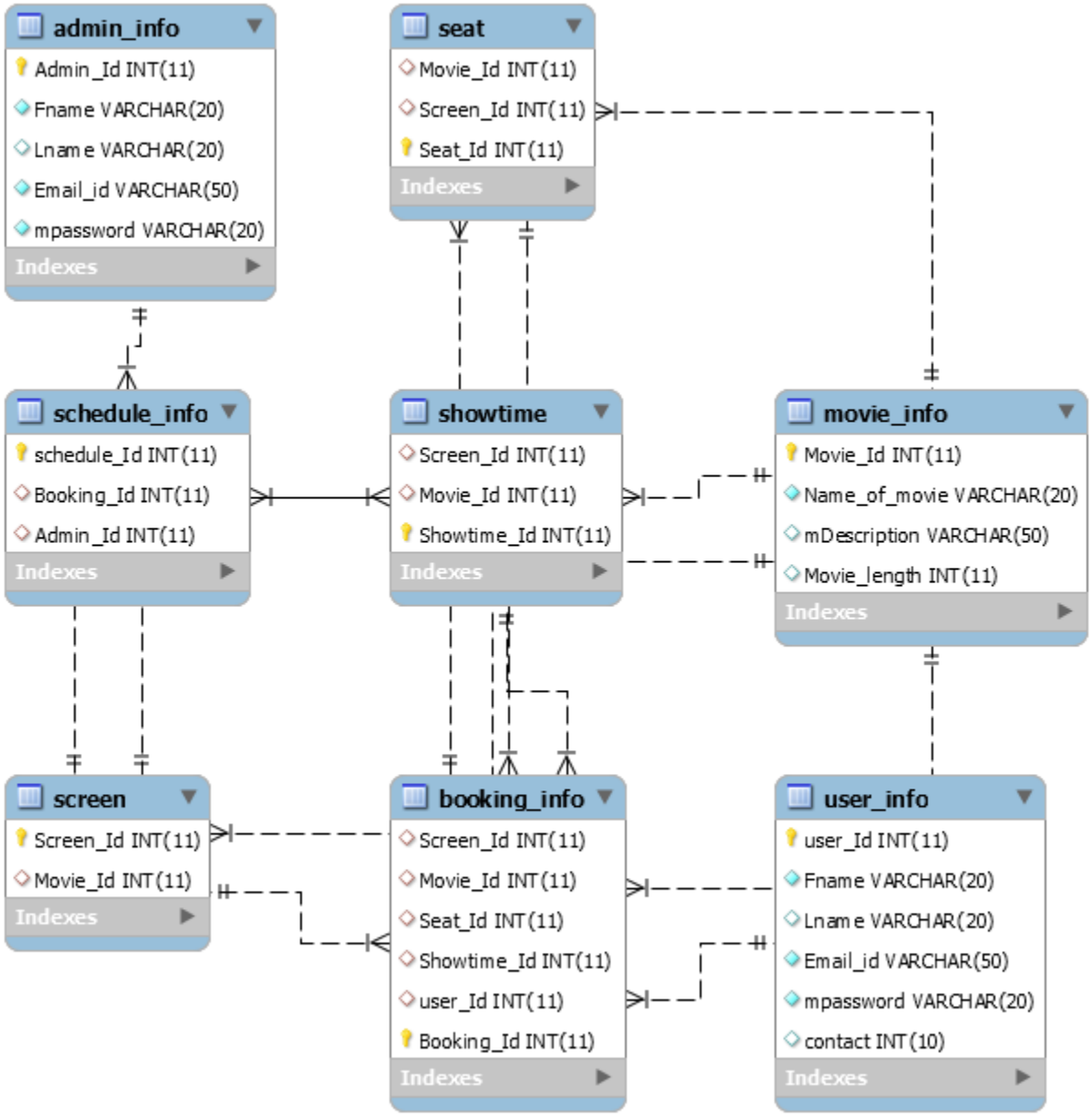

Figure 8. Database diagram

Architectural requirements:

The main goal of this architecture is to facilitate access of multiple users to the online ticket booking system. In this way, many clients would be able to book their ticket in a short time with only sending a request to the server.

We used this architecture because it is fast for multiple access and in case of user traffic, we would be able to handle it in server side. There is no complicated business logic required for this system, so we used the simple and fast architecture to handle the system.

Functional Requirements:

Refer to Use Cases or Use Case scenarios which are relevant with respect to the software architecture. The Use Cases referred to should contain central functionality, many architectural elements or specific delicate parts of the architecture.

The overview below refers to architecturally relevant Use Cases from the Use Case Model (see references).

Source	**Name**	**Architectural relevance**	**Addressed in:**
Admin Add and Remove and make Changes to Movies	Admin module	Admin has the access to add any new user and edit details of movies and add them.	Section 5 usecase view
User Registration User See the ticket availability	User module		Section 5 usecase view

Non Functional Requirements:

Source	**Name**	**Architectural relevance**	**Addressed in:**
SRS	Performance.	This is important to client-server Architecture because single server will be accessed by multiple clients. The response from	Section 14

SRS .	Reliability	the server should be fast and serve multiple users simultaneously. The users should be able to rely on the system. The synchronization problem will be addressed here as multiple users access the system at the same time.	Section 14

1. Use case view (Scenarios)

The scenarios describe sequences of interactions between objects, and between processes. They are used to identify architectural elements and to illustrate and validate the architecture design. They also serve as a starting point for tests of an architecture prototype.

There are two kinds of access that has been provided for the system.

If user is “Admin”, she/he would be able to “Add”, “Edit” and “Delete” a movie from movie lists that user later on can book a ticket for it. However, if the user is a normal client with no extra access, then he/she would only be able to book a ticket. So, in order to book a ticket, client would be able to see the list of available movies, available seats, date & time of showing the movie. After selection of criteria, total price of tickets would be shown to the user. If she/he accept to continues, then they would be led to the payment page. After the payment, tracking number and confirmation info would be shown to the user.

2. Logical view

The functionality of the system depends on the type of user. In client side, we check if the user is admin user, they can have an access to Movie add/edit/delete functionalities. However, if the user is normal client, they would have access to the booking system only and they won’t be able to edit or make any changes on the movie section. So, on client side user can select from movie types, ticket count, show date & time in client side and send the request to the server to book the ticket. Server side would book the ticket for the user and lead her/him to the payment section. Payment section is a component in an external server and we only have access to it by a service. If payment was successfully, user will be directed to the booking successfully done page in the end with

a tracking number in client side and history of booking would be sent to server to keep it as a history in the database.

Subsystems

Describe the decomposition of the system in subsystems and show their relation.

Architecturally significant design packages

Describe packages of individual subsystems that are architecturally significant. For each package includes a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

Use case realizations

In this section you have to illustrate how use cases are translated into *UML interaction diagrams*. Give examples of the way in which the Use Case Specifications are technically translated into Use Case Realizations, for example, by providing a sequence-diagram. Explain how the tiers communicate and clarify how the components or objects used realize the functionality.

3. Development (Implementation) view

As the architecture that has been selected for this system is client-server, at server side we have database, database connection and logic part of the system. In the client side, we only have UI which is .jsp pages and javascript validations. There is also a payment external server that application server needs to reach it by a service.

Reuse of components and frameworks

The only third-party component that is reused is Payment Component in order to pay for the ticket online.

4. Process view

As it is illustrated in activity diagram, first page is welcome page which user should decide to go through the process of login as a user or admin. If user chose to login as admin, after authentication, they would have access to make changes on the movie options available for users. However, if user chose to login as a normal user, the first thing that would be asked on the process is whether user is guest or they already have registered. If she/he

is a guest, they should go through registration process, if not, they should log in by their email and password. If login fails, there would be only message showing up for user and process ends here. However if login id done successfully, user would be entre to movie booking process. Booking process, consist of two other processes which are movie selection and payment. So, first user is able to select the seat, then they can enter their payment details. Here, we have considered validation process as well. So, if payment data was valid, receipt of payment would be shown to the user.

The concurrency issue is handled by blocking other users access to the same seats until 10 minutes. If user doesn't make the payment, seat would be free for others to choose. And, if two users select the same seat at client page, the request of the one would be confirmed that arrived earlier at the server even a mili-second.

5. Deployment (Physical) view

There are three main parts in the system. Web Server part includes application component, DB Server which is allocated to store user's data and User Client that includes the web interface component to communicate with clients.

A physical server is required to have application logic and database server on it, and these two components actively are interacting with each other and application receives and assesses requests that are sent by clients from outside of our server.

6. Data view

In order to store the user data, we have used MySql relational database. We are storing data related to both types of users (admin/client), movies, seats, screens, show times for each movie, and booking history of users.

7. Quality

Quality goals that we have declared for the system are as following:

Performance Efficiency:

- Description : System's reaction when user demands increase
- Solution : J2EE application servers support several workload management techniques

Usability:

- Description : How easy the functionality and interface of system is for user

- Solution : Simple and easy web page interface designed for clients

Reliability:

- Description : When multiple users send request to the server at the same time
- Solution : J2EE application server supports load balancing through clusters

Security:

- Description : Security associated with online ticket payment
- Solution : SWIFT MT payment component will be reused

Maintainability:

- Description : System is easy and affordable to maintain
- Solution : All related functionalities have been encapsulated in the same component and all components are loosely coupled with each other

References:

1. https://www.process.st/checklist/online-movie-ticket-booking-system/
2. https://www.academia.edu/31536019/MOVIE_TICKET_BOOKING_MANAGEMENT_SYSTEM_PROJECT_REPORT.doc
3. https://www.techopedia.com/definition/438/clientserver-architecture
4. http://apachebooster.com/kb/what-is-client-server-architecture-and-what-are-its-types/
5. http://testingnotes.com/basic-characteristics-client-server-testing-architecture.html
6. https://docs.oracle.com/cd/E13203_01/tuxedo/tux80/atmi/intbas3.htm
7. https://cio-wiki.org/wiki/Client_Server_Architecture