**#Milestone 1: Project proposal**

**SOEN 6471**

**Team 4**

|  |  |
| --- | --- |
| **Student Name** | **Student ID** |
| **Kiranmayie Bethi** | **40092284** |
| **Nasim Adabi** | **40079444** |
| **Hina Thahseen** | **40076287** |
| **Swetha Chenna** | **40092019** |
| **Sahana Shankar** | **40092026** |
| **Mahshad Saghaleini** | **40058409** |
| **Nandini Bandlamudi** | **40105415** |
| **R V Pavan Kumar Reddy** | **40083392** |

# 

**Project Proposal:**

**Movie Ticket Booking System**

1. High-level requirements

**User Module**

**User Requirements**

* Registration

User registration is required to book the ticket. An unregistered user cannot book the ticket.

* Login

User should give valid login credentials.

* Decide the movie

The user can see list of movies with show date, show time and venue. User can select the movie from the movie list.

* See the ticket availability

After selecting the movie, the user can see the ticket availability. If ticket is available, user can book the ticket.

* Seat Selection

User can select the desired number of seats with his credentials and a confirmation message is shown on the system.

* Add and Remove Changes to Cart

User can add or remove the number of tickets.

* Ticket Cancellation

The user will be given an option of cancellation before the payment process.

* Payment

The user can pay for the ticket using credit card, The card is validated and a confirmation message is shown on the system.

* Generation of Ticket

After booking, the ticket is displayed on the system with the information of the booked movie with an unique ID.

**Admin Module**

**Admin Requirements**

* Login

Admin should give valid login credentials.

* Add and Remove Changes to Movies

Admin can add or remove the number of movies.

* Logout

Admin should be able to logout from the system.

**2. Why we have chosen Client-Server architecture for this system?**

* Client-server architecture is suitable for the systems that need to have only one access point which provides service that many clients can use that service.
* In Movie Ticket Selling system, there are many clients involved and one server is managing the availability of the tickets and the whole information related to the process.
* So, in this case we cannot use peer-to-peer architecture since clients must not be able to see each other’s information and do some functionalities like adding movies and customers shouldn’t be able to give services, only getting services are allowed.
* For a simple synchronisation application like Movie Ticket selling we don’t need to complicate using rest of architectural styles like MVC.
* The main reason to use client server is centralized computing because the server can check the updated information that is requested by users about ticket availability immediately and it can handle synchronization among clients.
* In addition, if there are many clients that want to access the system at the same time, the server can manage users’ access to the system in term of response time and space.
* In this architecture there will be a server which will host, deliver and manages all the necessary services and all of these will be consumed by the client.
* We can also accommodate multiple users in the same time using multithreading concept which will be a perfect match for this proposed system.
* As the system will be centralized any addition of new updates or services can be easily done.It is an advantage in terms of maintenance to allow more changes for a user friendly application.
* There will be a huge advantage over scalability as both horizontal and vertical scalability is possible.
* Centralized security is also an added advantage which maintains the safety of the customer information.
* Movie Ticket Selling System is ubiquitous application and accessed by different vendors with different platforms. The hardware platform and operating system of the client and server need not be same. Client and Server communicate through a well defined set of protocols and interfaces.
* Using Client-server model as the architecture model for our application gives us the opportunity to build the system regardless of the hardware platform or operating system software.
* This application is a **large enterprise application** which offers diverse functionalities. Servers can be partitioned by resources, functionalities and databases and replicated for increased fault tolerance and performance. Thus offering more power and flexibility to support more services.
* Considering all these points, we have chosen to develop this system using client-server architecture.

**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