**Process Flow Design:**

The idea of creating a cinema booking system can be made easy by splitting tasks to various design patterns like Structural, Behavioral, Creational. Tasks can be even classified and sorted based on Front end, back end, Database.

**Key requirements**:

1. A website/application with user-friendly interface for user communication with database:

* It should display movie information like List of movies, description of movie, show timings, ratings and reviews.
* Ticket information and booking facility: Showcasing ticket price, seating information, list of available and booked seats, ticket offers like group tickets, unlimited cards.
* Secure payment gateway linking to bank API and database with different payment options and sophisticated authentication methods for user’s data and privacy.
* Notification alert: Messaging and email notifications on booking details, OTP for account authentication, offers and alerts.

1. Database to store data and to process user queries: Tables with movie data, ticket price, user and payment data
2. A server to host the website
3. System should be properly maintained for updating /deletion of movie information.

**Consideration:**

**2.**

* Secure Authentication: User authentication and user account should be securely maintained for data integrity.
* Account Privacy: User account may possess user private details and should be securely handled.
* Server Flooding: Sometimes when the number of users using the system is almost to threshold, the server may slow down. Limiting the users to the system at any point is necessary.
* Maintenance issues: It is necessary to hold the website during maintenance and update otherwise user can see unreliable information.
* Verification: A need for proper user verification, validating user reviews.
* Secure Payments: A great need for secure payments system with various user authentication methods like OTP, Password and account backup and retrieval methods incase of account lockout.
* Tool and Database selection: Design and maintenance tools should be properly selected and employed.

**Problems:**

* Data Privacy: Unauthorized users may try access the account and this spoils data privacy. So, need to send alerts on login attempts.
* Slow server: Multiple people using the system at once can make the server respond sluggish. Need to restrict number of users at any moment.
* Payment problems: Sometimes payment may get failed for many reasons like problems with bank server.
* Login issues: Forgetting account credentials can be a problem. So, there is any need to pose some memorable questions to retrieve the details.
* Maintenance: If users are not notified on maintenance timings, they can see different data on website and this can be a big problem. Updating of system, maintenance should be done during non-working hours for user ease.

**Tools:**

**Wireframing tools:**

Mockflow, Axure

**UI and Prototype tools:**

Sketch, Adobe Xd, Zeplin, Figma

**Web Design debugging tools:**

Eclipse, Webflow

**Deployment/ maintenance tools:**

AWS CodeDeploy, Gitlab, Jenkins

**Backend and frameworks:**

Spring, Flask, Firebase

**Database:**

SQL / Oracle/ Mongo DB

**Process Flow:**

|  |
| --- |
| Gathering requirements |
| Selecting tools and techniques |
| Defining user stories |
| Designing user interface |
| Testing and integrating finished code |
| Developing backend and database testing should be |
| Testing finished code done along with |
| Testing the system by integrating front, back, UI every task |
| Correcting errors, buys and updating |
| Deploying the system |