<Group 01>

<Web Booking Movie>
Software Architecture Document

Version <1.0>

Revision History

Date	Version	Description	Author
09/12/2022	1.0	Define software architecture	Group 01

Table of Contents

Software Architecture Document

1. Introduction

The introduction of the Software Architecture Document of Web Booking Movie provides an overview of the entire Software Architecture Document. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the Software Architecture Document.

2. Architectural Goals and Constraints

Performance:

- Focus should be on front-end optimizations by
- Using fast server to handle traffic
- Providing cross-browser compatibility
- Using minification
- Reducing HTTP request

Usability:

- User should be at ease in using website without any specialized training
- User should be able to relate further action needed for an interaction
- Web application should not have any ambiguity regarding the consequences of an action

Accessibility:

- The system has many user support functions (especially disabled person)

Security:

- Protecting information from external attacks by mean of
- Authorization
- Authentication
- Encryption
- Session Management
- Exception Management

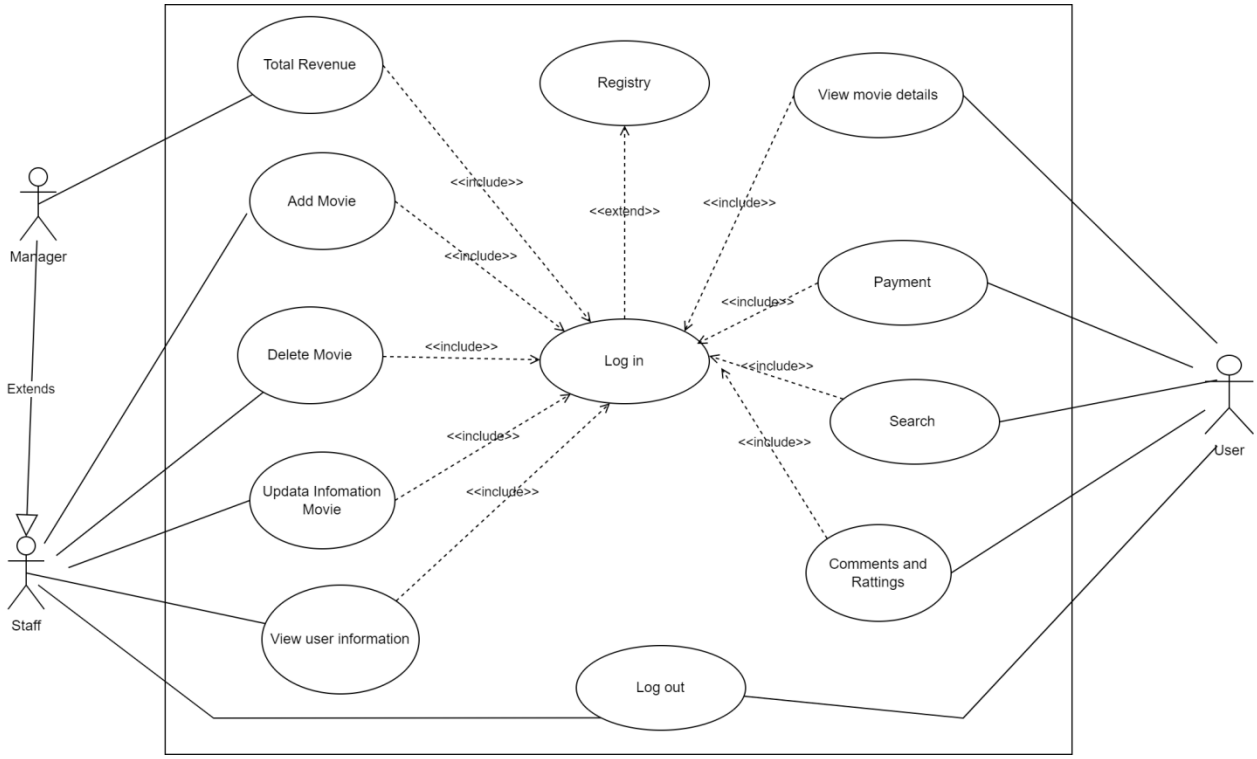
Maintainability

- It refers to capability of web system to maintain its performance over time.
- We can assure reliability by making website robust, recoverable and available under adversity.

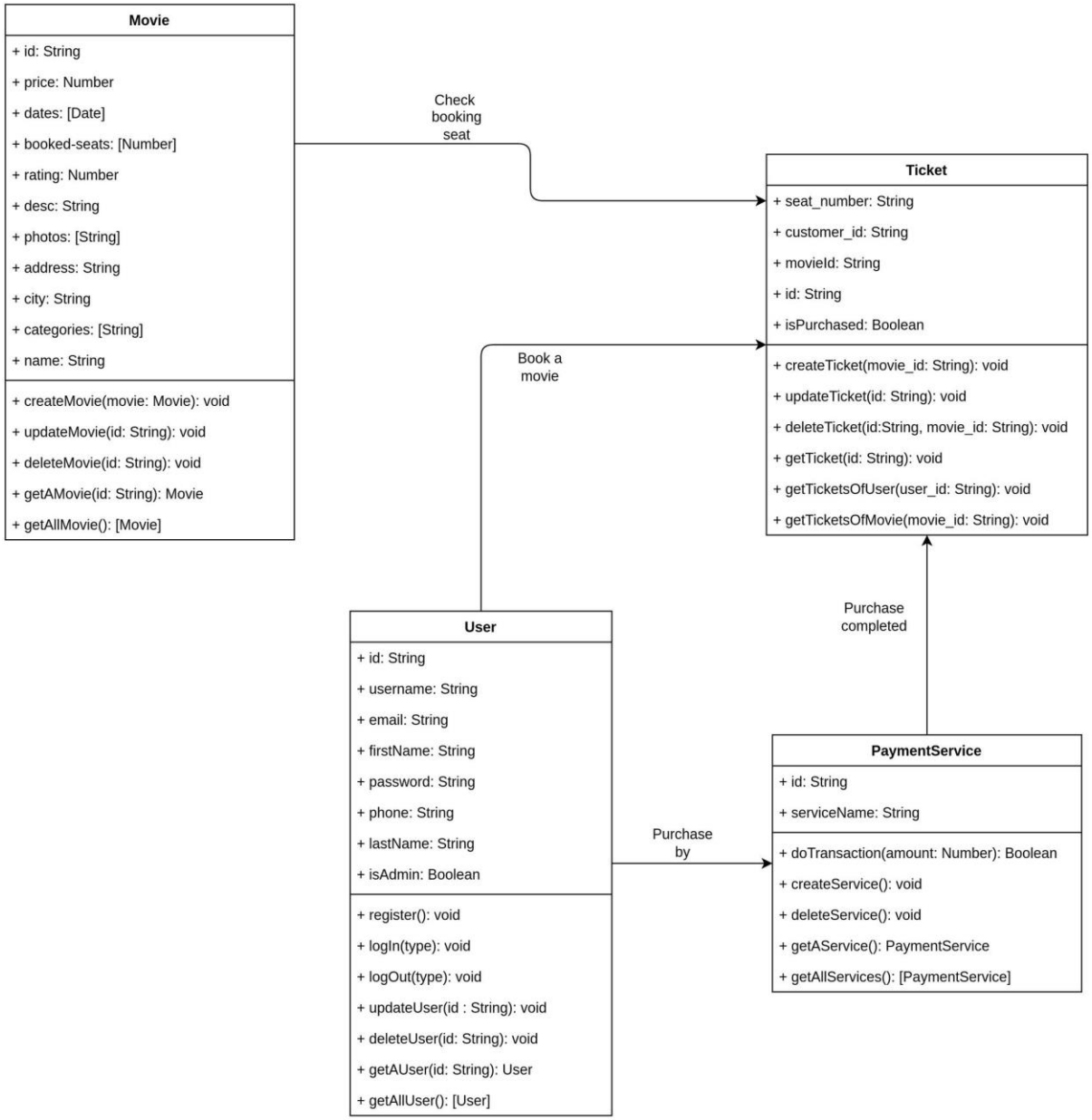
User Interface:

- Interface designs must accomplish any task with efficiency and effectiveness
- To reduce the weight of the users, interface designs must be lightweight and provides information in a more subtle, succinct manner.

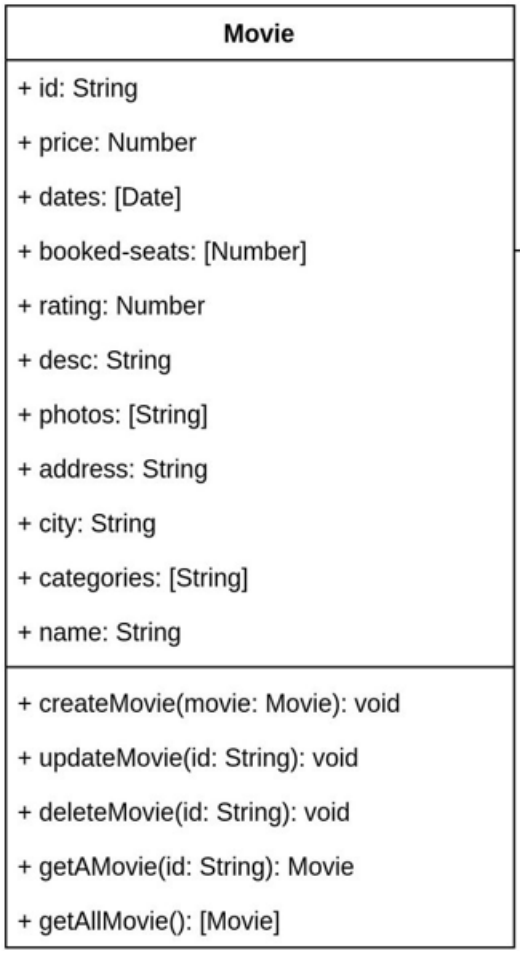
3. Use-Case Model



4. Logical View



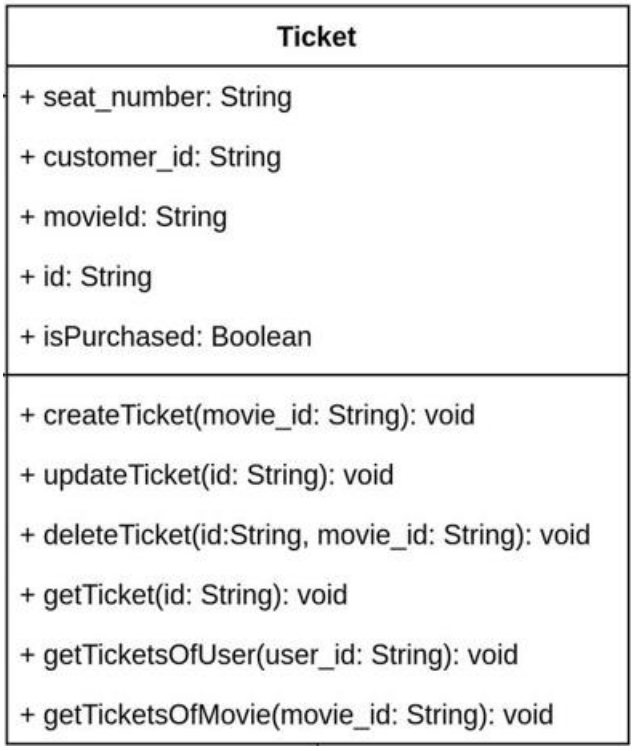
4.1 Component: Movie



Class name: Movie
Atributes:
- id: string → id of user.
- price: number → price of movie .
- date: [date] → premiere schedule.
- booking-seats: [number] → number of seat booked.
- rating: number → ratting for movie .
- desc: string→ describe about information movie .
- photos: [string] → poster of movie .
- address: string → movie theater address.
- city: string → city of movie theater address.
- categories: [string] → categorical movie such as action, discovery, ...
- name: string → name of movie.

Behaviours:
- createMovie(movie: Movie) : void → create a movie in the system.
- updateMovie(id: string): void → update more information of movie .
- deleteMovie(id: string): void → delete a movie from the system.
- getAMovie(id: string): Movie → get information on a movie in the system.
- getAllMovie(): [Movie] → get information on all movie in the system.

4.2 Component: Ticket



Class name: Ticket

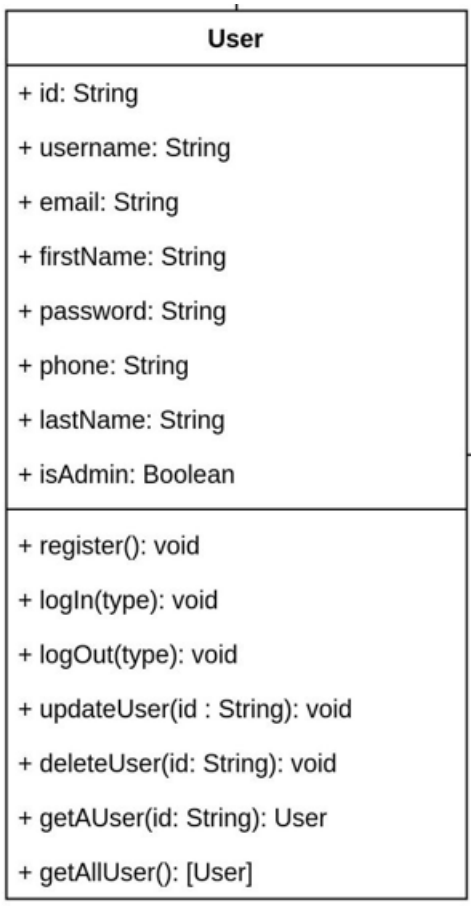
Atributes:

- Seat_number: string → number of seat booked.
- Customer_id: string → id of user when buying the ticket.
- movieId: string → id of movie.
- id: string → id of ticket .
- isPurchased: boolean → Check whether the purchase was successful or not.

Behaviours:

- createTicket(movieId: string) : void → create a ticket in the system.
- updateTicket(id: string): void → update more information of ticket.
- deleteTicket(id: string): void → delete ticket from the system .
- getTicket(id: string): void → get information on a ticket in the system.
- getTicketOfUser(user_id: string): void → get information of user in the system.
- getTicketOfMovie(movieId: string): void → get information of movie in the system.

4.3 Component: User



Class name: User

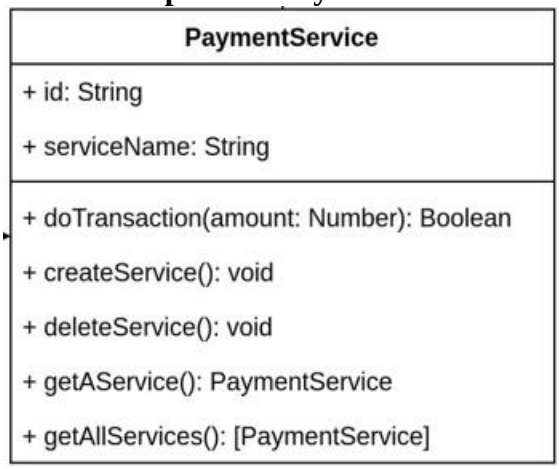
Atributes:

- id: string → id of user.
- usename: string → usename of user.
- email: string → email's user.
- firstName: string → firstname's user.
- password: string → password.
- phone: string → phone number .
- lastName: string → lastname's user.
- isAdmin: boolean → check if the user is admin or not.

Behaviours:

- register(): void → register a new account if the user has not yet.
- logIn(type): void → after user has account, user can login in the system.
- logOut(type): void → logout, use can logout when the user's account has login.
- updateUser(id: string): void → update more information of user in the system.
- deleteUser(id: string): void → delete user from the system.
- getAUser(id: string): void → get information of user.
- getAllUser(): [User] → get information of all user.

4.4 Component: PaymentService



Class name: PaymentService
Atributes:
- id: string → if of payment service.
- serviceName: string → name of service such as card, mono, vnpay,...

Behaviours:
- doTransaction(amount: number): boolean → check if the payment is successful or not.
- createService( ) : void → create a payment in the system.
- deleteService(): void → delete payment service from the system.
- getAService( ): PaymentService → get information of the payment in the system.
- getAllService(): 2[PaymentService] → get information of all payment service in the system.

5. Deployment

6. Implementation View