A background image of a movie theater interior. In the foreground, a white bowl filled with popcorn sits on a red cup. Behind it, a red cup with a white lid is visible. The background shows rows of red theater seats.

Movie ticket booking system

Yelena Martinez

Roadmap

Overview of product

Live demonstration

Functionality

User scenarios

Future plans



What is it? Who is it for?



The system is a user-friendly application for booking tickets to see a movie.



It can be used by anyone who has an interest in watching the latest movies.

Live video Demonstration

QuickTime Player File Edit View Window Help

MB MovieBookingSystem Version control BookingApp

Open Files ▾

- MovieBookingSystem ~/IdeaProjects/
- src
- com
- example
- moviebooking
 - Booking
 - BookingApp
 - Movie
 - Payment
 - Seat
 - Main

Main.java BookingApp.java Booking.java Seat.java Movie.java Payment.java

```
1 package com.example.moviebooking;
2
3 import javax.swing.*;
4 import java.awt.event.ActionEvent;
5 import java.awt.event.ActionListener;
6 import java.util.Random;
7
8 public class BookingApp {
9     private static Seat[] seats; // Array of Seat objects 8 usages
10    private static Random random = new Random(); // Random instance for seat availability 1 usage
11    private static Movie[] movies; // Array of Movie objects 2 usages
12    private static JCheckBox[] seatCheckboxes; // Declare seatCheckboxes at the class level 11 usages
13
14    public static void main(String[] args) {
15        // Initialize movies
16        movies = new Movie[]{
17            new Movie(id: 1, title: "Venom: The Last Dance", genre: "Sci-Fi", showtime: "18:00", rating: 8.2, summary: "Eddie Bro
18            new Movie(id: 2, title: "The Wild Robot", genre: "Animation", showtime: "20:00", rating: 8.5, summary: "Shipwrecked c
19            new Movie(id: 3, title: "We Live in Time", genre: "Romance", showtime: "22:00", rating: 7.4, summary: "Almut and Tobi
20        };
21
22    JFrame frame = new JFrame(title: "Movie Booking System");
```

Run BookingApp

/Users/yelemart/Library/Java/JavaVirtualMachines/openjdk-23.0.1/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=65223:/Applic

Processing payment via: PayPal

Process finished with exit code 0

MovieBookingSystem > src > com > example > moviebooking

3:22 LF UTF-8 4 spaces



Bookings [-]

- booking_id [serial]
- user_id [int]
- movie_id [int]
- selected_seats [varchar(255)]
- booking_date [timestamp]



Movies [-]

- movie_id [serial]
- title [varchar(255)]
- genre [varchar(100)]
- showtime [timestamp]
- duration [int]
- rating [decimal(2,1)]

SQL Demo

Disclaimer:

I was not able to find an SQL server that was free. So, I had to resort to using an online SQL editor to test out my ideal functions. It will not allow me to download without paying.



Payments [-]

- payment_id [serial]
- booking_id [int]
- amount [decimal(10,2)]
- payment_date [timestamp]
- payment_method [varchar(50)]
- status [varchar(20)]
- transaction_id [varchar(255)]



Seats [-]

- seat_id [serial]
- movie_id [int]
- seat_label [varchar(10)]
- available [boolean]

Searching movie by title

With `SELECT * FROM
Movies WHERE title LIKE
'%The Wild Robot%'`;

We get:

< Input

Run SQL

```
SELECT * FROM Movies WHERE title LIKE '%The Wild Robot%';x
```

Output

movie_id	title	genre	showtime	duration	rating
	The Wild Robot	Animation	20:00	175	8.3

Searching movie by genre

With `SELECT * FROM
Movies WHERE genre =
'Sci-Fi'`;

We get:

Input

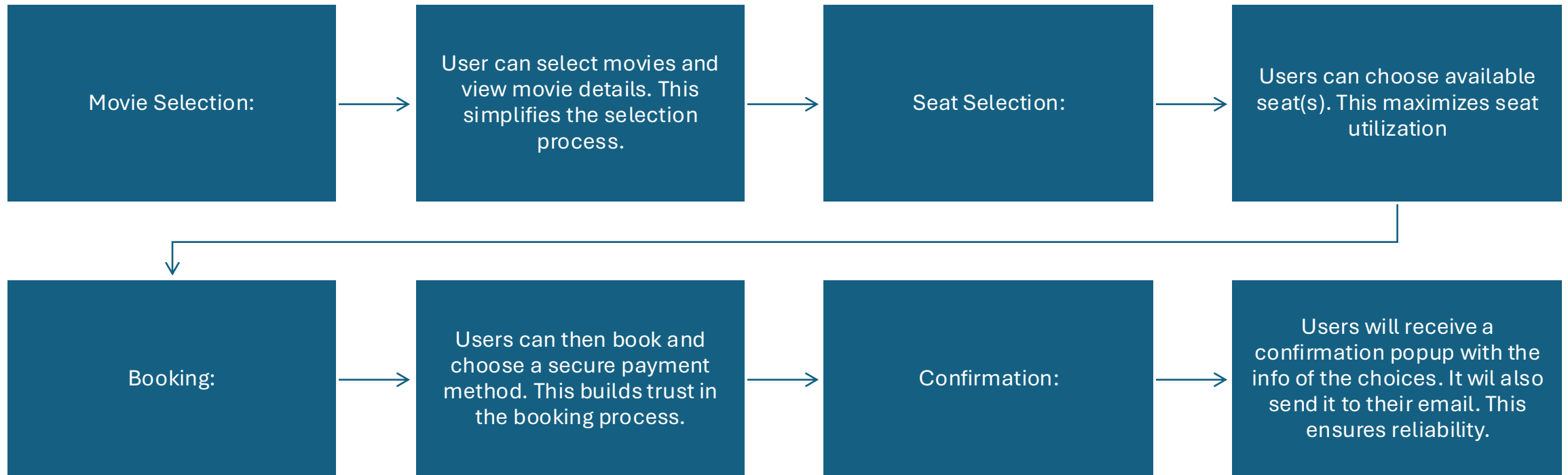
Run SQL

```
SELECT * FROM Movies WHERE genre = 'Sci-Fi';
```

Output

movie_id	title	genre	showtime	duration	rating
	Venom: the last dance	Sci-Fi	18:00	148	6.2

Functionality and business goals



User Scenario

Scenario: A user wants to book a ticket for “Venom: The last dance”.

Steps:

1. User opens the application
2. They select the movie from the dropdown
3. They choose available seat(s) and enters their email
4. Confirms booking
5. Choose secure payment option
6. Receives a confirmation message

Main aspects:

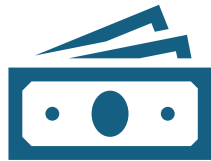
Easy navigation, intuitive interface !

```
mirror_mod = modifier_ob.  
Set mirror object to mirror  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly  
  
-- OPERATOR CLASSES --  
  
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```

Future Plans



A better layout



Enhanced payment
processing options



Enhanced interactive
seating



Enhanced structured
data storage.



Thank you!