
POP-FLIX

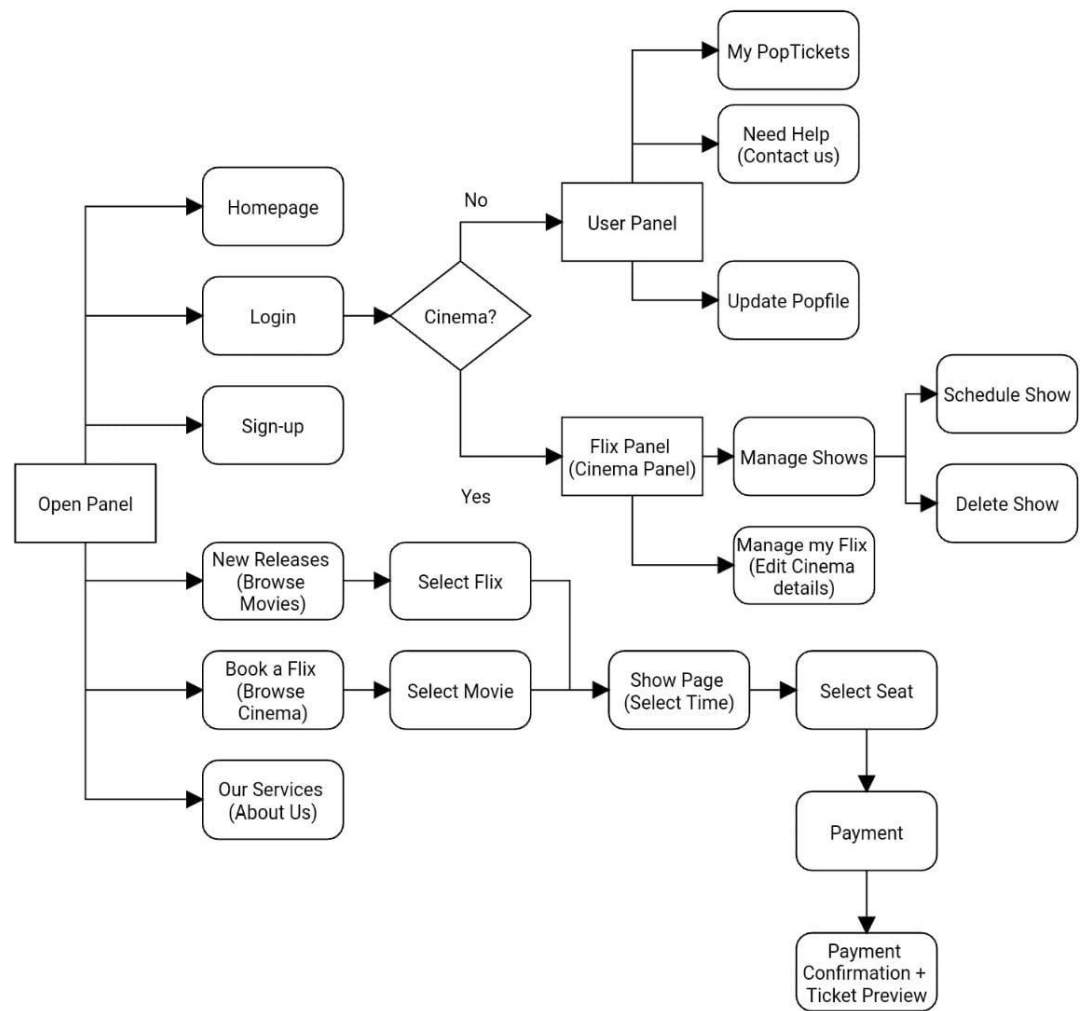
U19CS012 BHAGYA RANA
U19CS029 ANJALI SINGH
U19CS077 AASTHA PATEL
U19CS123 KRISHNA PATEL

FLOW OF THE PROJECT

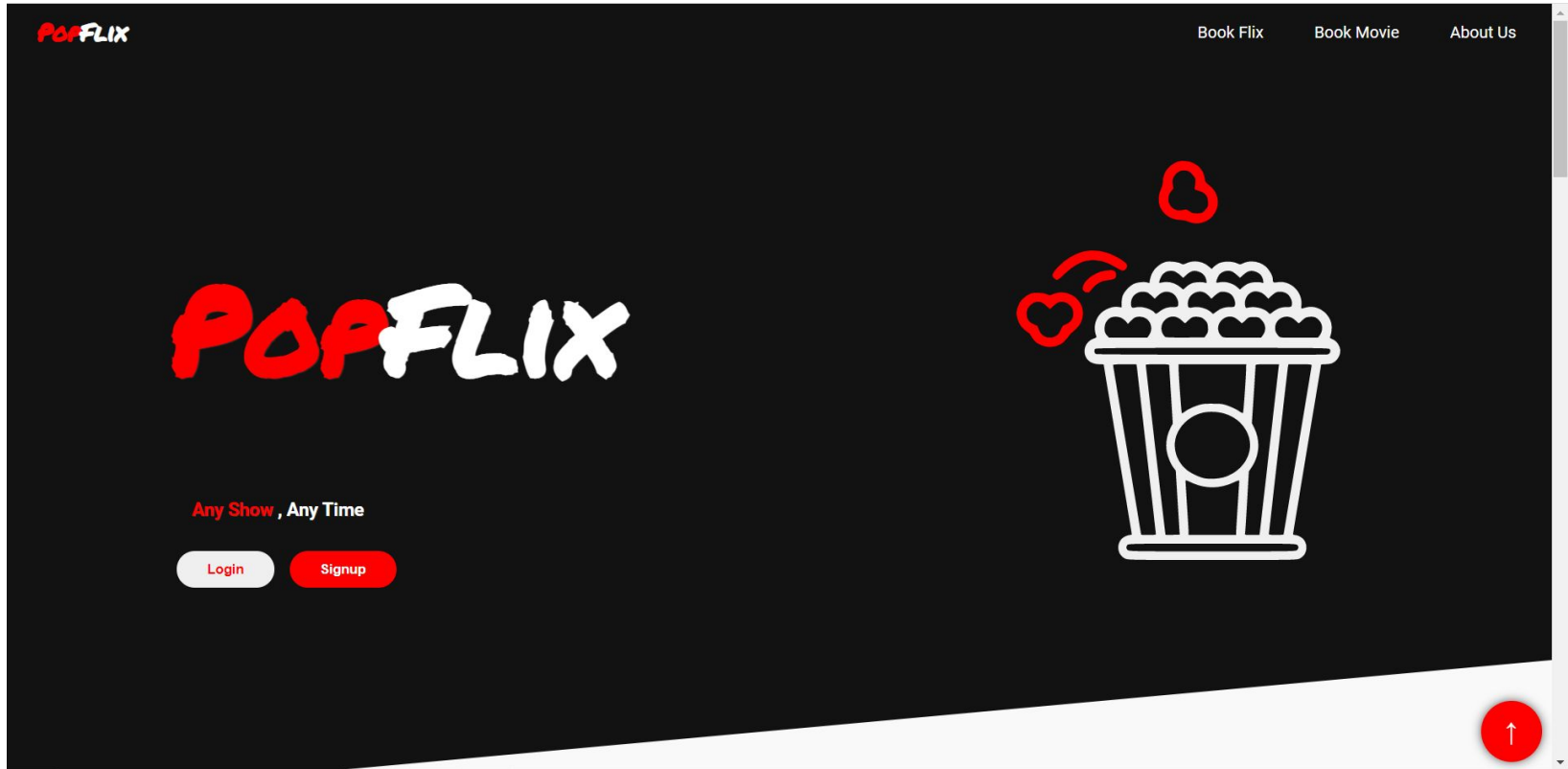
Customer

Theatre Owner [Add the New Movies in his Theatre, Show Timing & Fee]

Admin [Add Movies & Theatre]



LIVE DEMO



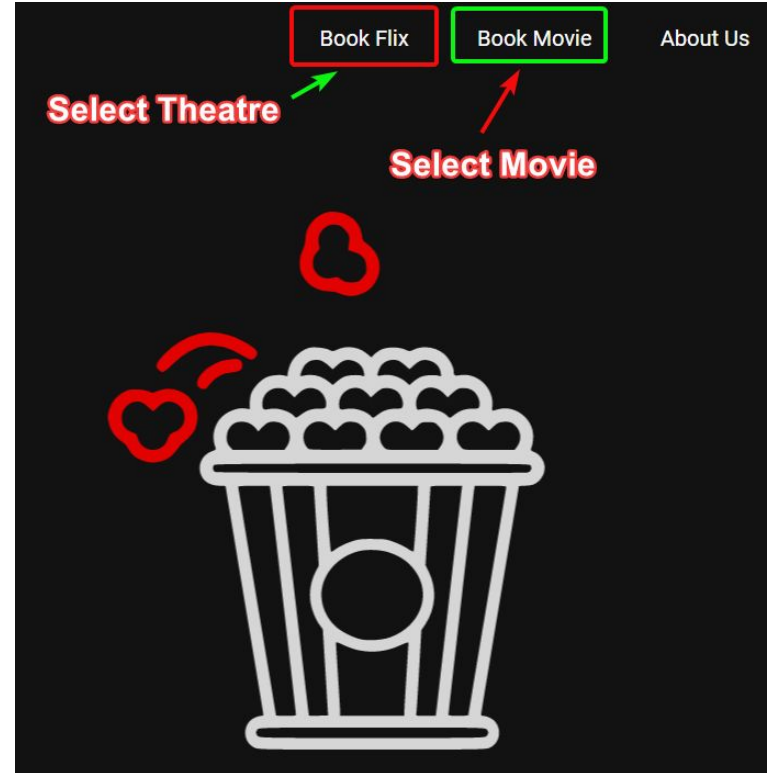
(A) CUSTOMER BOOKING TICKET

Customer can Either

Select the **Theatre** First and Then
Select the **Available Movies** at that
Theatre

OR

Select **Movie** First and Select the
Theatre where Movie is **Currently
Showing**



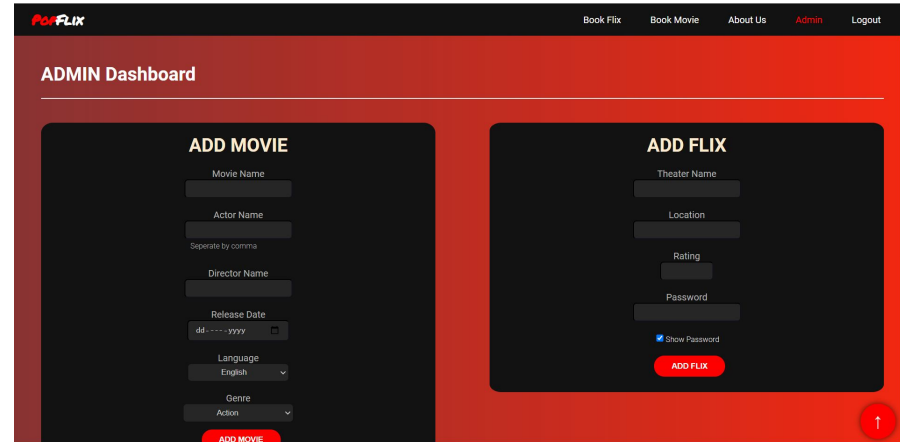
(B) ADMIN ADDING MOVIE AND THEATRE

Admin can **Add** the Newly Released **Movie** in Database, So that The **Theater Owners** can Add that Movie in their Theatres.

Eg: Fast and Furious 9

Admin can also Add a **New Theatre** in the **Database**.

Eg: PVR Cinema



The screenshot displays the ADMIN Dashboard for a system named 'PopFLIX'. The dashboard has a red header with navigation links: 'Book Flx', 'Book Movie', 'About Us', 'Admin' (highlighted), and 'Logout'. Below the header, the 'ADMIN Dashboard' title is centered. The main content area contains two dark-themed panels. The left panel, titled 'ADD MOVIE', includes input fields for 'Movie Name', 'Actor Name', 'Director Name', and 'Release Date' (with a date picker), a dropdown for 'Language' (set to 'English'), and a dropdown for 'Genre' (set to 'Action'). A red 'ADD MOVIE' button is at the bottom. The right panel, titled 'ADD FLIX', includes input fields for 'Theater Name', 'Location', 'Rating', and 'Password', with a 'Show Password' checkbox. A red 'ADD FLIX' button is at the bottom. A red circular icon with an upward arrow is visible in the bottom right corner of the dashboard area.

ADMIN VIEW

ADD MOVIE

Movie Name

Actor Name

Seperate by comma

Director Name

Release Date

dd - - - - - yyyy ☐

Language

English

Genre

Action

ADD MOVIE

ADD FLIX

Theater Name

Location

Rating

Password for Theatre Manager →

Password

☒ Show Password

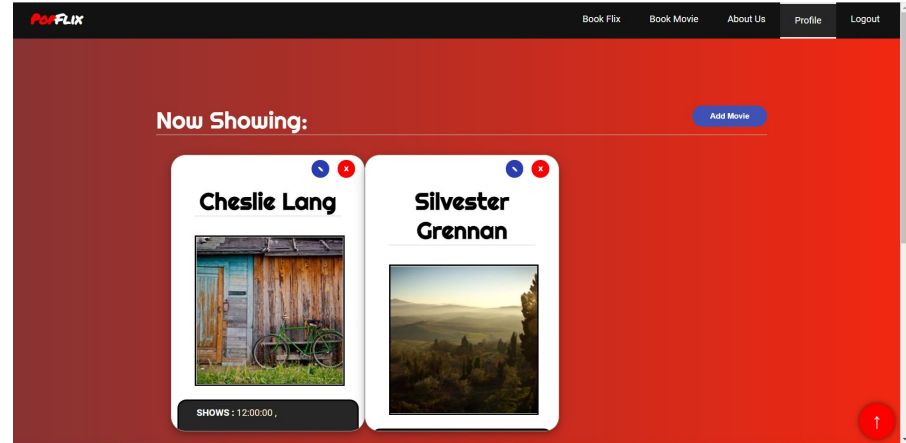
ADD FLIX

(C) THEATRE ADMIN ADDING SHOWS & MOVIES

Theatre Admin will add the Movie to his Theatre.

He will Add **Shows** [Like 10AM/2PM] in his Theatre along with Price of Ticket on Weekday and Weekend.

Eg: I want the F&F 9 to be shown in my Theatre in 3 Time Slots.



THEATRE MANAGER VIEW

POPFlix[Book Flix](#)[Book Movie](#)[About Us](#)[Profile](#)[Logout](#)

ADD Show

Movie Name

Select Movie

Set Show Time

No Slot Selected


Set Price

Enter Price

Set Weekend Price

Enter Price


ADD SHOW





SHOWS : 12:00:00 ,


THEATRE MANAGER

Theatre Manager can **Edit** and **Delete** the Show

Edit the Show → 

 Delete the Show \ 

Cheslie Lang




SHOWS : 12:00:00 ,

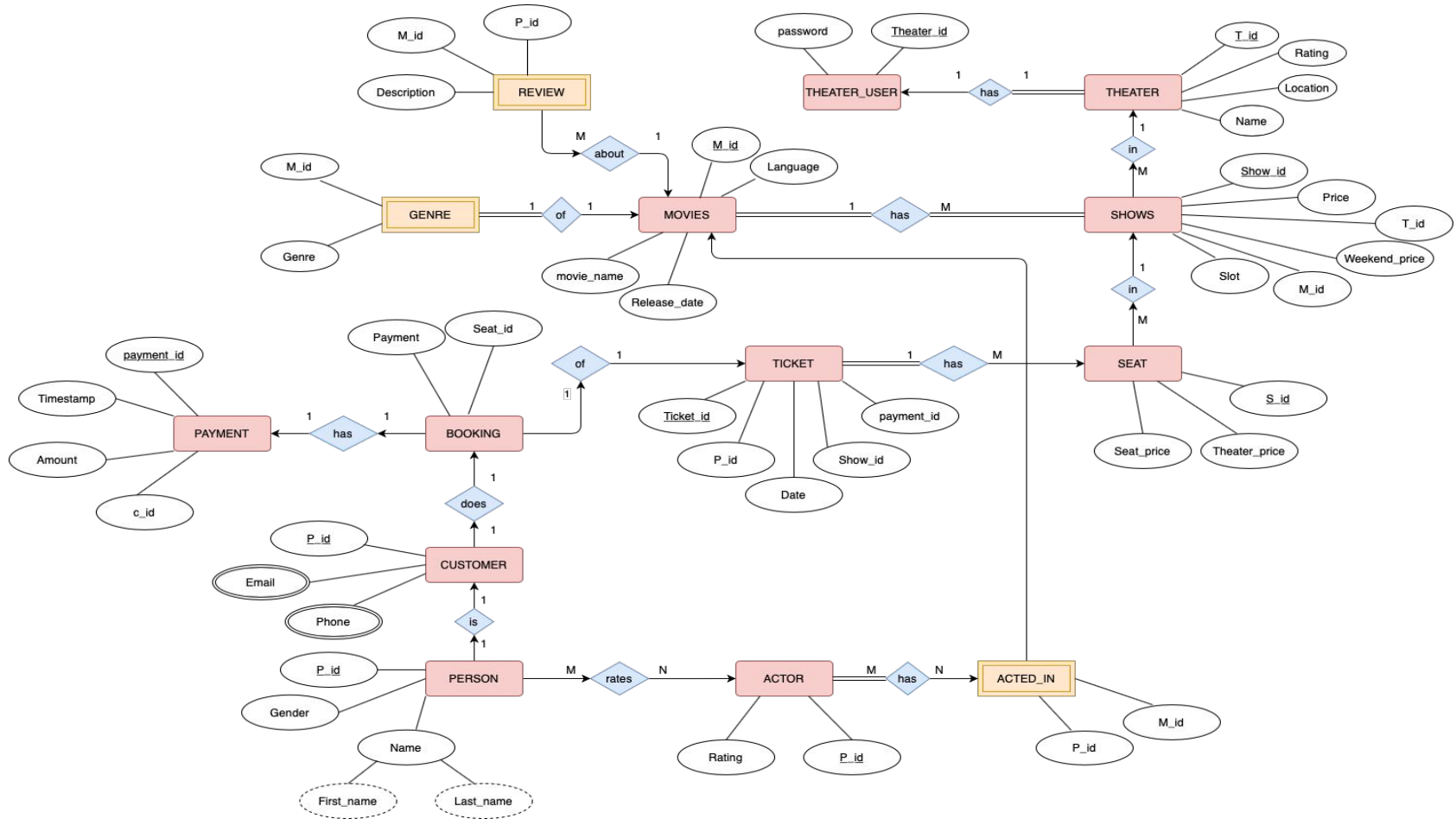
LANGUAGE : English

RELEASE DATE : Wed Sep 30
2020

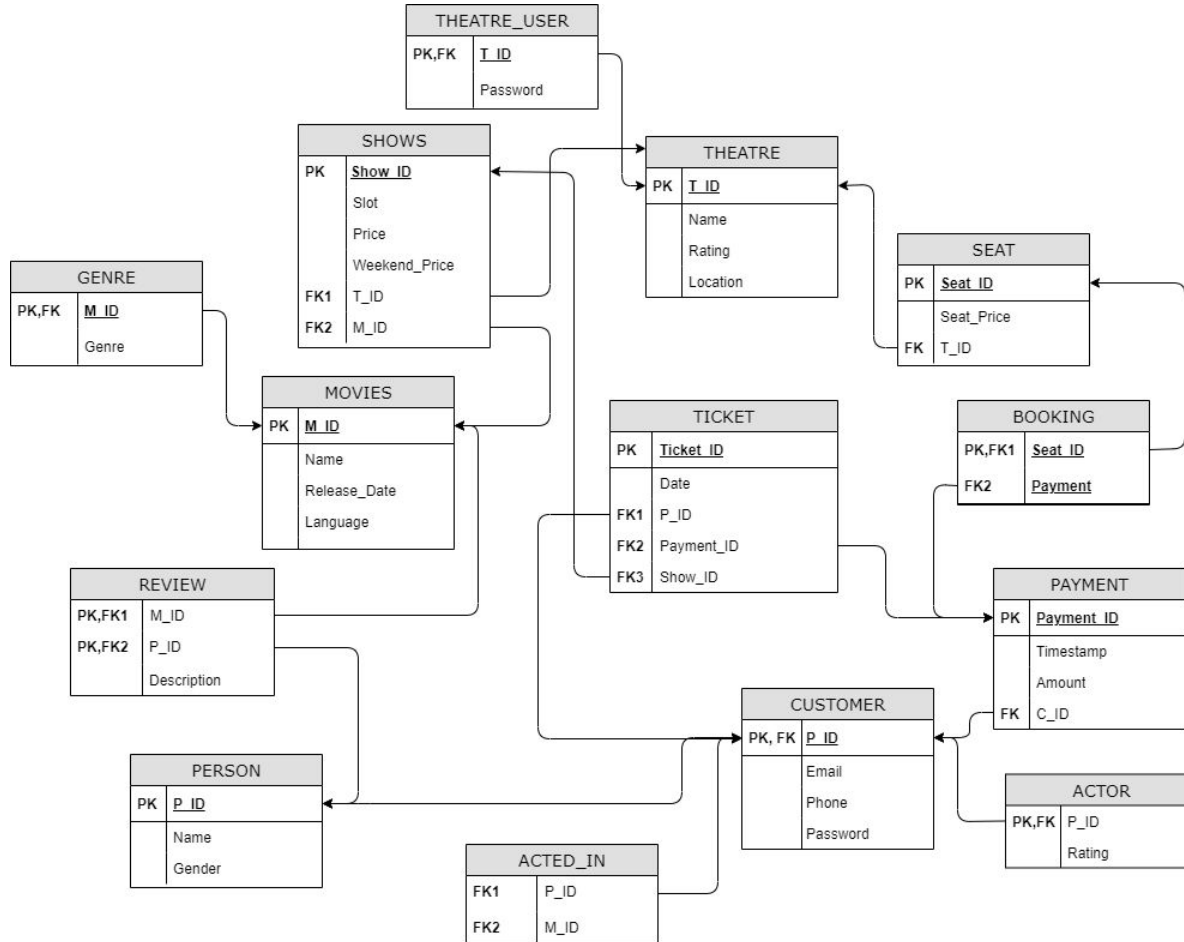
Silvester Grennan



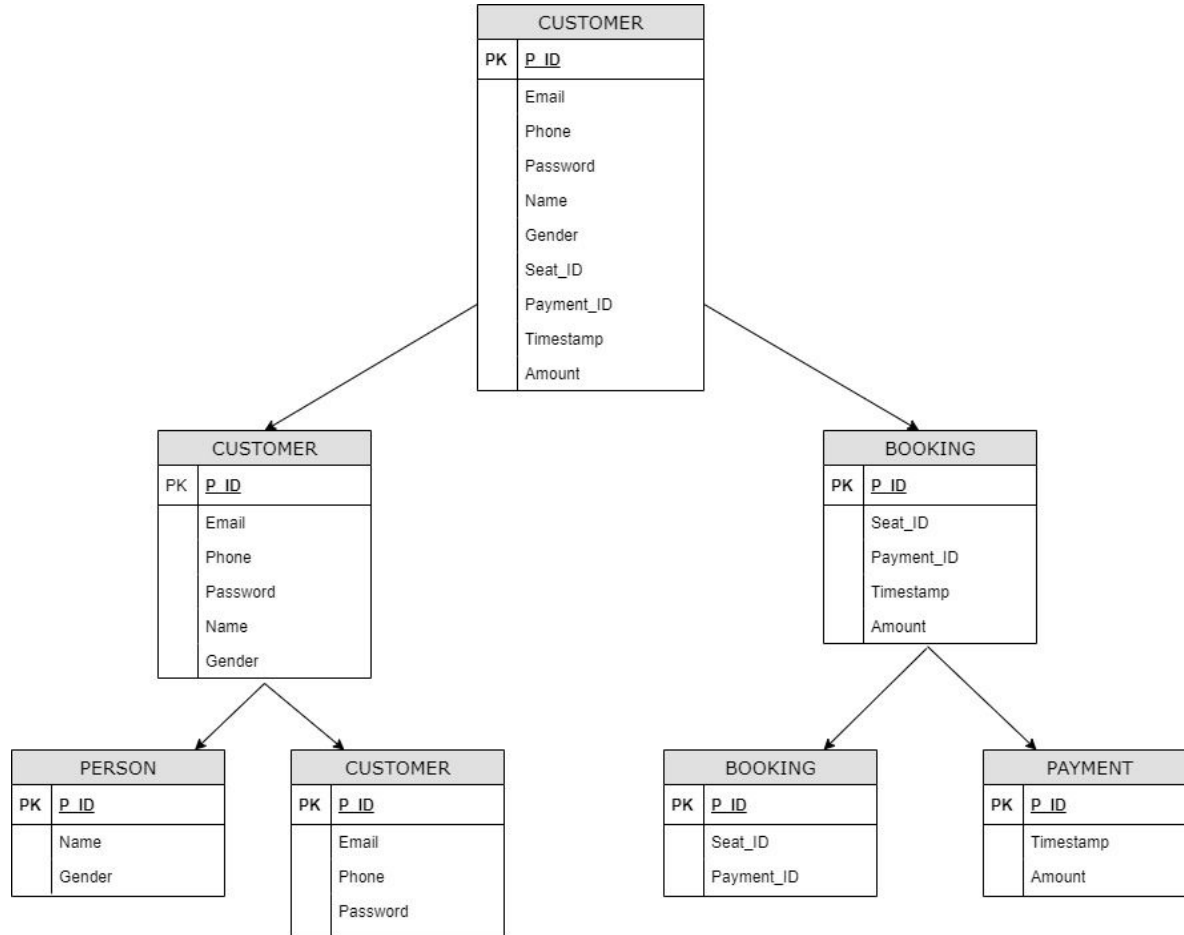
ENTITY RELATIONSHIP DIAGRAM (ER)



CONVERSION OF ER DIAGRAM TO RELATIONAL MODEL



NORMALIZATION OF TABLES



BASIC IMPLEMENTATION - FRONT END

- Can **Book** Movies
- Can **Search** For Movies
- Can also Add **Review**
- Can **Update Profile**
- **Media Query Support** (Changes Itself If Screen Resolution is Changed)

Bernarr Gower

Date: Thursday 1st October 2020

Actor :

Movie description: Movie description: Movie description: Movie descriptio
ea quis qui sunt consequat aute dolore officia nisi fugiat deserunt anim ar

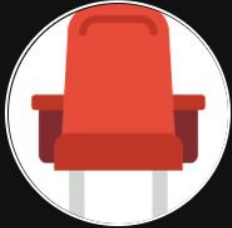
B BHAGYA RANA

This is Awesome Movie!

← **Review**

USER PROFILE AND TICKETS SHOWN (BOOK MOVIES)


MY PROFILE




BHAGYA RANA

Update Username and Phone


♂ Male

 9876976985

 bvr@gmail.com

Update

POP TICKETS



Alana - Eartha Preto

Mon Apr 26 2021

Amount : 146
Purchased on : Sun Apr 25 2021
Location : Londonderry
Seat No :11

SEARCH FEATURE OF PROJECT



This image shows a search interface for movies. It features three input fields: a text box for 'Enter Movie Name', a dropdown menu for 'Language', and another dropdown menu for 'Genre'. Each field is highlighted with a green box and a green arrow pointing to it from a label below. The labels are 'Search By Movie Name', 'Filter by Language', and 'Filter by Genre'. To the right of these fields is a red 'Search' button.

Enter Movie Name

Language

Genre

Search

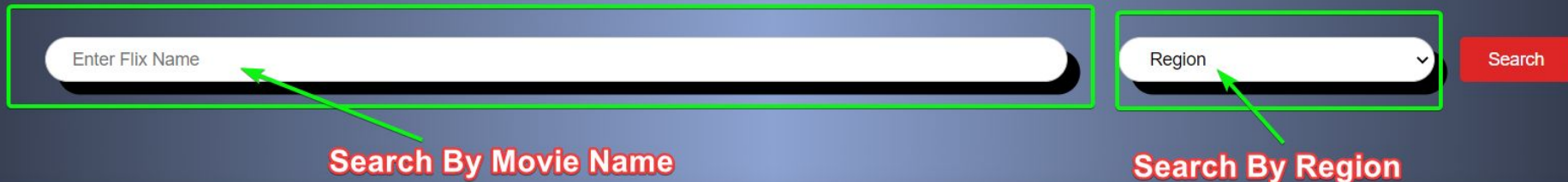
Search By Movie Name

Filter by Language

Filter by Genre

You can Search/Filter Movies Using:

- ❖ Movie Name, Language Filter, Genre Filter
- ❖ Theatre & Theatre Region Filter



This image shows a search interface for movies by name and region. It features two input fields: a text box for 'Enter Flix Name' and a dropdown menu for 'Region'. Each field is highlighted with a green box and a green arrow pointing to it from a label below. The labels are 'Search By Movie Name' and 'Search By Region'. To the right of these fields is a red 'Search' button.

Enter Flix Name

Region

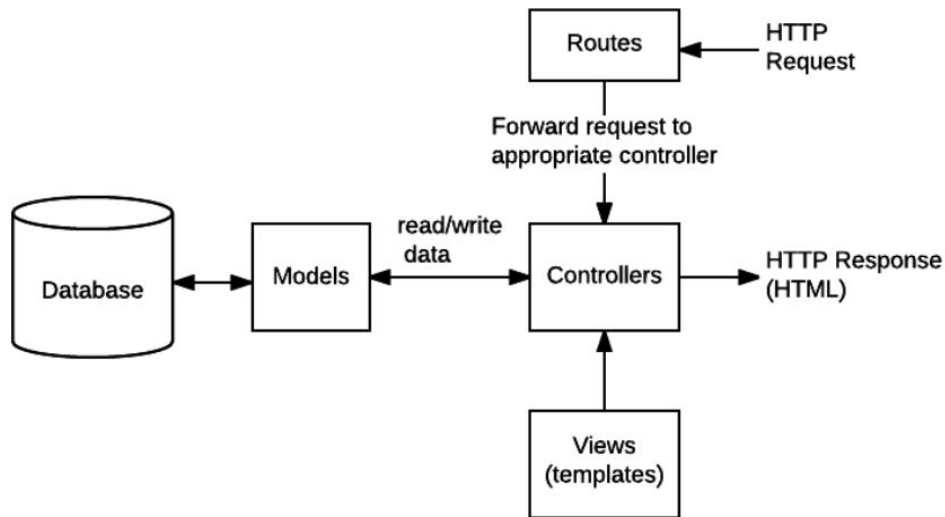
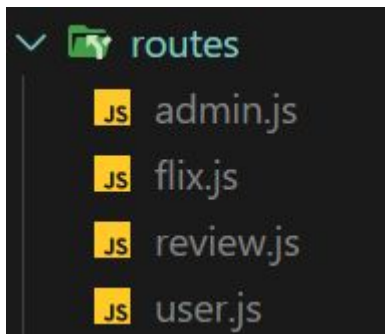
Search

Search By Movie Name

Search By Region

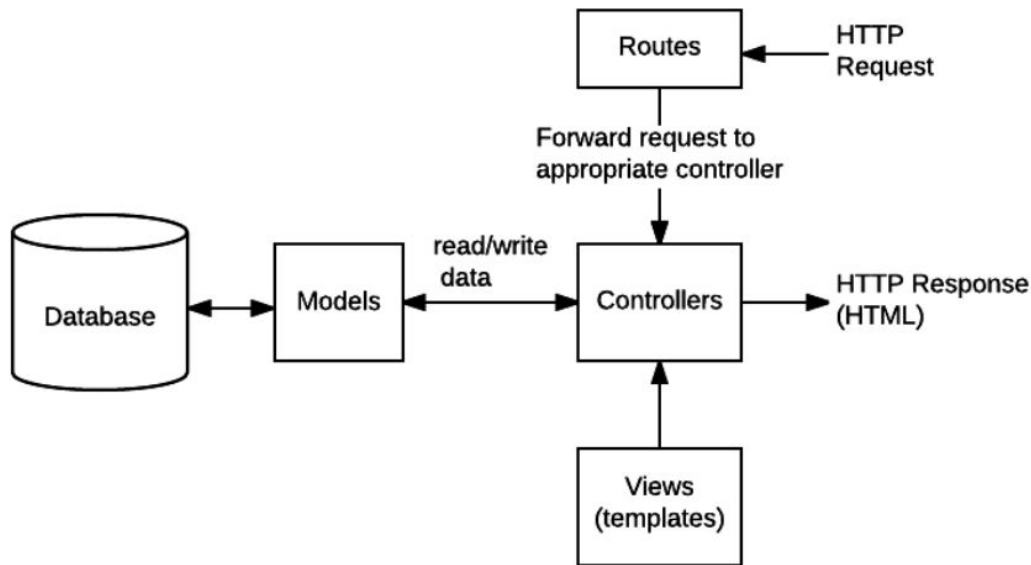
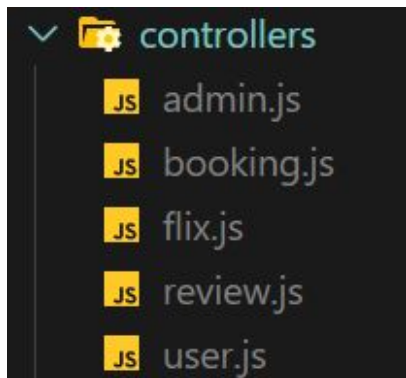
BASIC IMPLEMENTATION - BACK END

The **Routes** Folder forward the supported requests to the appropriate **controller functions**.



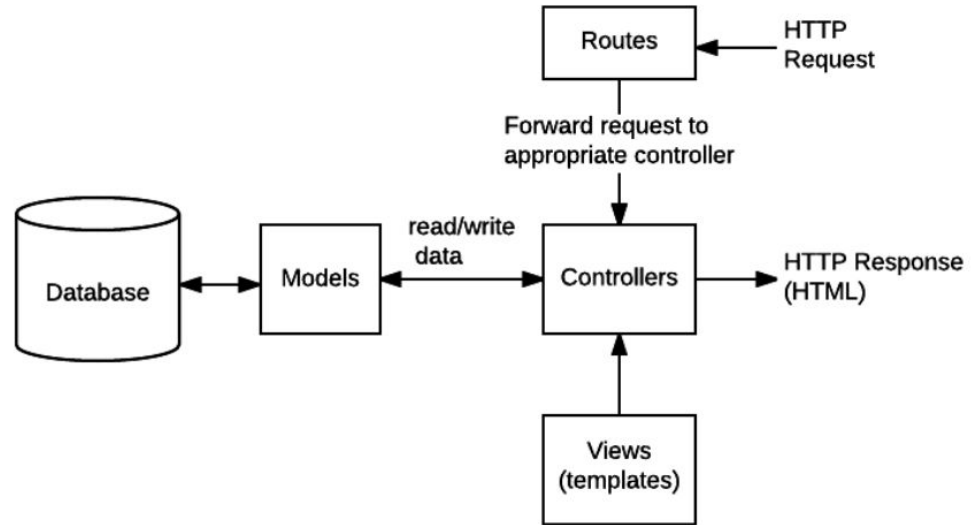
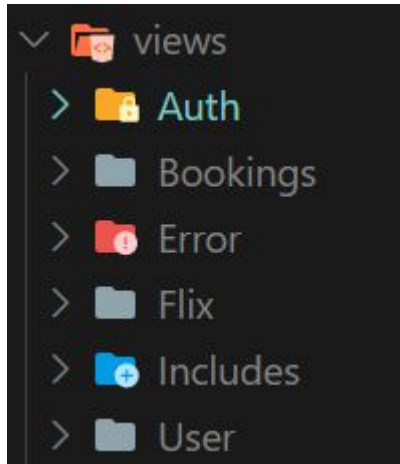
BASIC IMPLEMENTATION - BACK END

Real SQL Queries are in
'**Controllers**' Folder to get the
requested data from the
models, create an **HTML page**
displaying the data, and return it
to the user to view in the browser



BASIC IMPLEMENTATION - BACK END

The Frontend [EJS Files] is there in 'Views' Folder. **Views** used by the controllers to render the data.



KEY CONSTRAINTS

All the Key Constraints

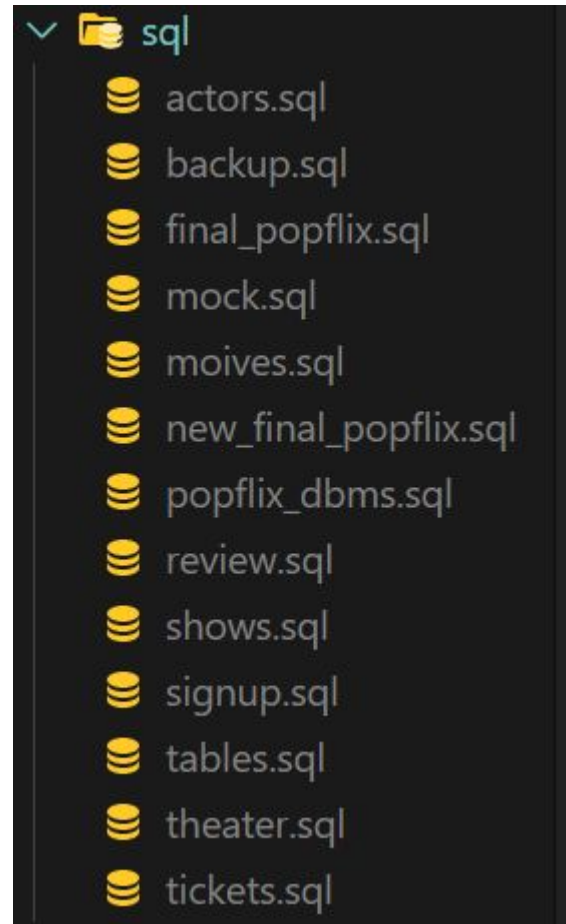
Primary Key,

Foreign Key,

Null, Unique Values.

Joins and Views also Implemented

[For Checking, we have made Queries in **sql** Folder]



VIEWS CREATED & USED in booking.js

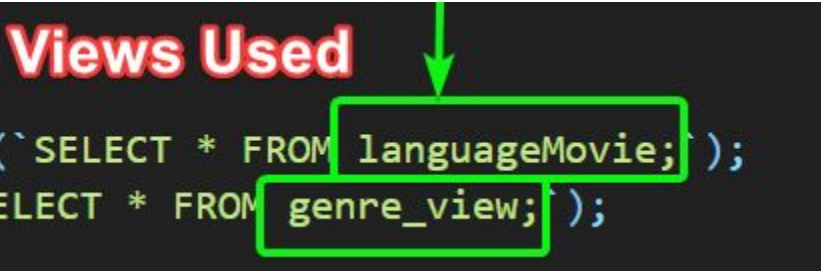
```
CREATE VIEW `locationTheater` AS SELECT Location FROM theater;
```

```
CREATE VIEW `genre_view` AS SELECT DISTINCT Genre FROM genre;
```

```
CREATE VIEW `languageMovie` AS SELECT DISTINCT LANGUAGE FROM movies;
```

```
91     mov[x].actors = actors;
92   }
93   let dropLanguage = await query(`SELECT * FROM languageMovie;`);
94   let dropGenre = await query(`SELECT * FROM genre_view;`);
95
```

Views Used




EXTENSION WITH PL/SQL

Stored Procedure Called in Exports.searchMovie [controller/booking.js]

Name	Action				Type
movieSearch	 Edit	 Execute	 Export	 Drop	PROCEDURE


```
let mov = {};  
if (language == 'English') language = 'EN';  
else if (language == 'Hindi') language = 'Hi';  
else if (language == 'Marathi') language = 'Ma';  
else language = '%'  
if (movieName == '') movieName = '%';  
if (genre == '') genre = '%';  
mov = await query(  
  `CALL movieSearch('${movieName}','${language}', '${genre}')`;  
);
```

Call to Stored Procedure




EXTENSION WITH PL/SQL

Stored Procedure Called in Exports.**searchFlix** [controller/booking.js]

Name	Action	Type
flixSearch	 Edit  Execute  Export  Drop	PROCEDURE

```
let dropRegions = await query(`SELECT Location FROM theater;`);  
  
let theaters = {};  
if (theaterLocation == 'Region') theaterLocation = '%';  
theaters = await query(`CALL flixSearch('${theaterName}', '${theaterLocation}');`);  
theaters = theaters[0];
```

Call to Stored Procedure



TRIGGERS

4 Triggers Implemented to Store the Backup of Customer Details along with its Payments and Tickets.

Triggers ⓘ							
	Name	Table	Action			Time	Event
<input type="checkbox"/>	t_customer ←	customer	Edit	Export	Drop	AFTER	INSERT
<input type="checkbox"/>	t_payment ←	payment	Edit	Export	Drop	AFTER	INSERT
<input type="checkbox"/>	t_person ←	person	Edit	Export	Drop	AFTER	INSERT
<input type="checkbox"/>	t_ticket ←	ticket	Edit	Export	Drop	AFTER	INSERT

CODE LINK

The Complete Code Link of Project:

https://github.com/BhagyaRana/POP_FLIX

SQL INJECTION PREVENTION

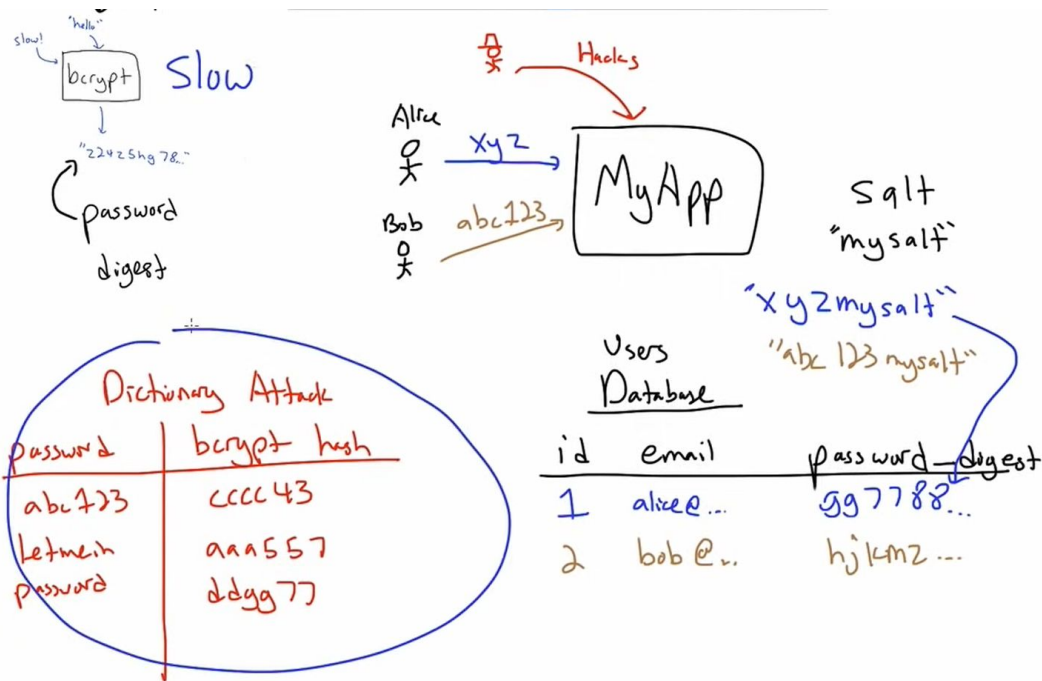
Proper Validation is Done at Frontend
[Initial Prevention]

Protect Our Users Security:

Passwords are Hashed using **bcrypt** Hashing Algorithm [Salt [To Prevent Dictionary Attack] + Original Password = Hashed Password]

Nearly Impossible to Get Real Password from Hash Generated

Login Authentication Secure




ABOUT US PAGE

POFFLIX


Book FlixBook MovieAbout UsAdminLogout

	Technology Used
FRONTEND	HTML5, CSS3, JAVASCRIPT
BACKEND	MYSQL, NODEJS


Our Team




Aastha Patel
+91 89183 65626
[Contact](#)



Anjali Singh
+91 67180 63269
[Contact](#)



Bhagya Rana
+91 73187 74141
[Contact](#)



Krishna Patel
+91 88478 57495
[Contact](#)

NORMALIZATION IN OUR PROJECT

(A) UnNormalized Form

Since Every Cell is Table Holds **Atomic Values**: Our Table is already in **1 Normal Form**
[Theoretical Concept]

Partial Dependency

1 TABLE: [HAVING PD] [1NF FORM]

(**Person_ID**, **payment_id**, Gender, First_Name, Last_Name, Email, Phone, TimeStamp_Of_Payment, Amount)

p_id	pay_id	gender	f_name	l_name	email	phone	time	amount
------	--------	--------	--------	--------	-------	-------	------	--------

(B) Partial Dependency Removed

2 TABLES: [HAVING TD] [2NF FORM]

(**Person_ID**, Gender, First_Name, Last_Name, Email, Phone, **Payment_ID**)

(**Payment_ID**, TimeStamp_Of_Payment, Amount, Ticket_ID, Show_ID, Timing)

Person_ID	gender	f_name	l_name	email	phone	payment_ID
------------------	--------	--------	--------	-------	-------	------------

Payment_ID	TimeStamp_Of_Payment	Amount	Ticket_ID	Show_ID	Timing
-------------------	----------------------	--------	-----------	---------	--------

FD's:

Person_ID, -> Gender, First_Name, Last_Name, Email, Phone, **Payment_ID**

Payment_ID -> TimeStamp_Of_Payment, Amount, Ticket_ID, Show_ID, Timing

But, This Still have Transitive Dependency

Transitive Dependency

(Person_ID, Payment_ID, Ticket_ID) [TABLE]

Person_ID	Payment_ID	Ticket_ID
-----------	------------	-----------

FD's:

Person_ID → Payment_ID

Payment_ID → Ticket_ID

(C) Transitive Dependency Removed [3NF FORM & BCNF FORM (LHS is Candidate Key)]

FURTHER, DIVIDED INTO 3 MORE TABLES:

(Person_ID, Payment_ID)(Payment_ID, Ticket_ID)(Ticket_ID, Show_id, Date) [TABLES]

Person_ID	Payment_ID
-----------	------------

Payment_ID	Ticket_ID
------------	-----------

Ticket_ID	Show_id	Date
-----------	---------	------

FD's :

Person_ID → Payment_ID

Payment_ID → Ticket_ID

Ticket_ID → Show_ID, Date

These Process is Repeated Similarly for Other Tables to Obtain the Our Final Normalized Database.

Finally, Our Tables are Normalized in **BCNF Form.**

All the Redundancy and Insert, Update and Delete Anomalies are removed.