Interview

User Flow

- Show/search/find shows
- User selects one
- If the showId of the selected show has a lottery running, the `Enter Lottery` button is visible amongst other Information about the show. This is checked with the data returned from `/api/get_show` ajax call. A large object with one of such keys being `lotteryId` - a lotteryId, 0 if not running a lottery.
- User clicks 'Enter Lottery' button.
- Page uses more screen space for visual content to give the user a strong feel for the show which they are thinking to attend. They're asked how many tickets they want, one or two. Provides a **href** link to learn about the lottery in case they are too confused to move on. And has an **exit** button in the corner.
- User is asked to select showtime preferences, is able to select multiple. Visuals of the show content above (which still takes a large majority of screen space) have changed/zoomed to keep the user engaged compared to last scene. This time the exit button is a back arrow in case they want to change how many tickets were selected.
- Now after clicking **Next** a confirmation dialog is shown. X show, X tickets, for X showtimes, is the information correct? Confirm Enter Lottery?
- At this point the information cannot be changed and the user knows they'll enter lottery after clicking *Confirm Enter*. The backend will be sent an ajax **fetch POST** `/api/lottery/enter` with the supplied details for entering lottery.
- The user will be told if they are in the lottery depending on the data returned from the POST request. They also are told where they can view the status of their tickets in the UI. Back to home button also supplied. An option to send a push notification to their device upon winning is also shown, since we're assuming they are same day discounted tickets.
- Now the user has to wait until the lottery is drawn. The more people who enter the lottery and less tickets in stock, both decrease the probability of winning the lottery. Could be good idea to have more **winners** than **tickets in stock** for the case when Users do not buy even though they won the lottery.
- User is notified of lottery win or discovers it via viewing Lottery tab.

Terminology

`bookingID` = showID with a specific showTime.
There would be a bookings table in back-end db. Which also holds reference to a seats table.
On successful booking the seats table would be marked as taken, from being vacant.

Events

on Select Show in front-end

`/api/get_show` *takes:*
* **showID**

returns showObject which contains `lotteryId` property, if greater than 0, show is running a lotto.

Used when showing `Enter Lotto` button or not. ### on Enter Lottery Event `/api/lottery/enter`, *takes:*
* **object(showID)** * **object(lotteryID)** * **number(num_tickets)** * **list(showtimes)**

returns success code

Here the back-end will append to `db[userId]["pendingLottos"]` an object with {lotteryId, showID, num_tickets, showtimes}.

onPopulate pendingLottos list ui

- `/api/lottery/get_pending` *takes :*
  userID

returns list of awaiting pendingLotto objects.
`{lotteryId,showId,num_tickets,showTimes}`

on Lotto Draw timer back-end

Lets assume that the backend appends to a list upon parsing all lottery participants at roll time.

It would have had to pop from the db[userId][“pendingLottos”] list, and if the User gets lucky and wins as output from the winLotto algorithm, then lookup a valid bookingId to be placed in `db[userId]["discountedAccess"]`.

on Purchase Tickets

`/api/lottery/get_discounted`, *takes*:
* **userID**

returns the **discountedAccess list** for userId.

Thus giving the front-end access to `bookingID` and `lotteryID`.
If any of the bookingID's match the currently viewed show/booking, then the user can apply the bonus thus reducing the cost of the ticket.
Once the purchase is complete, the back-end removes the discountedAccess entry from list.

Api's used

`api/get_show` - when selecting a show, populates the screen, reveals if lotto is active for said show.
`api/lottery/enter` - register a `pending lotto` for that user - essentially they are lotto win preferences.
`api/lottery/get_pending` - get the list of pending lottos for user.
`api/lottery/get_discounted` - get the list of discountedAccess for user. BookingId's for which a user is eligble discounted prices.

Pseudo-code.

```
/*
Decided to use naming Show instead of Event.
Because of conflict with common programming styles using event to mean something different internally.
*/

class Show {
    id: number;
    info: string;
    lotteryId: number;

    constructor(id: number, info: string, lotteryId: number) {
            this.id = id;
            this.info = info;
            this.lotteryId = lotteryId;
    }

    getSalary() : number {
        return 10000;
    }
}

class PendingLotto {
  lotteryId: number;
  showId: number;
  numTickets: number;
  showTimes: number[];
  constructor(lotteryId: number, showId: number, numTickets: number, showTimes: number ) {
      this.lotteryId = lotteryId;
      this.showId = showId;
      this.numTickets = numTickets;
      this.showTimes = showTimes;
  }
}
type Seats = {
  vacant: boolean,
  vipLevel: number
}
type Booking = {
  bookingId: number
  eventId: number,
  seats: Seats[]
```

```
}

/*
Generates show instance from ajax json response.
*/
async function onShowSelected(showId: number): Promise<Show> {
  try {
    const response = await fetch('https://example.co.uk/api/get_show',
    {
        method: 'POST', // or 'PUT'
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify(showId),
    });
    const showData = await response.json();
    if ( showId !== showData.id ) throw new Error("Not the correct show returned.");
    const show = new Show(showData.id, showData.info, showData.lotteryId);
    console.log(`show ${showId} instance created`);
    return show;
  } catch (error) {
    console.error(error);
  }
}

async function enterLottery(prefs:PendingLotto): Promise<void> {
  try {
    const response = await fetch('https://example.co.uk/api/lottery/enter',
    {
        method: 'POST', // or 'PUT'
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify(prefs),
    });
    const resp = await response.json();

    return resp.success;

  } catch (error) {
    console.error(error);
  }
}

async function buyTicket(user: number,thisBooking: Booking): Promise<boolean> {
  try {
    const response = await fetch('https://example.co.uk/api/lottery/get_discounted',
    {
        method: 'POST', // or 'PUT'
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify(user),
    });
    const resp = await response.json();

    const eligibleForDiscount: boolean = false;
    // validate thisBooking with returned array of objs
    for ( let discountForBooking in resp ) {
      if ( discountForBooking.bookingId === thisBooking.bookingId ) {
        eligibleForDiscount = true;
```

```
        break;
      }
    }
    if ( eligibleForDiscount ) {
      processPayment();
      batchTicketDelivery();
    }

  } catch (error) {
    console.error(error);
  }
}

/*
Pseudo-code
userId is supplied with login code
User selects a showId with ui
*/
function flowUserSelectShow(showId: number, userId: number) {
  try {
      const show:Show = await onShowSelected(showId);

      // Lottery button clicked
      const lotteryButton = document.getElementsByClassName("enterLottery");
      lotteryButton[0].onClick = function(e) {
        const numTickets:number = await requestNumberTickets();
        const showTimes: string[] = await requestPreferredShowTimes();

        const lottoEntered = await enterLottery(new PendingLotto(show.lotteryId,show.id,numTickets,showTimes));
        if ( resp.success === true ) {
          console.log(`You have entered into a lotery for ${show.id} with ${numTickets} tickets`);
        }
      }

      // Purchase
      const buyButton = document.getElementsByClassName("purchaseTicket");
      buyButton[0].onClick = function(e) {
        const userSelectedBooking: Booking = getVisibleBooking();
        await buyTicket(userId,userSelectedBooking);
      }
  }
  catch( error) {
      // Handle error, unable to get correct show information.
      returnToHomeScreen();
  }
}
```