Project FIN Research Brief

Eric Guleksen, Kristen Brann, Laurentiu Pavel, Tushar Narayan

The Problem

VOX is a popular student theatre group on campus, performing many popular shows per year. Their shows are often so popular that people would like to buy their tickets in advance of the viewings. The present system for handling these requests works, but doesn't have any good data retrieval or permanence and will not scale.

The Proposition and Value

As VOX becomes more popular and more tickets are able to be pre ordered, the present system will not scale to the demand. This lack of scale will lead to longer queues due to the longer lookups, and unhappy people as shows might start late. Using a simple database can make the lookup of a person's reservation faster and more accurate for the ticket collectors in front of the theatre. The faster people can receive their tickets, the less likely the show will start late, allowing more performances and happier customers per show due to shorter queue times.

Outline of Page and Components

The page itself is outlined as prescribed using 4 high level divs. The divs themselves are described below.

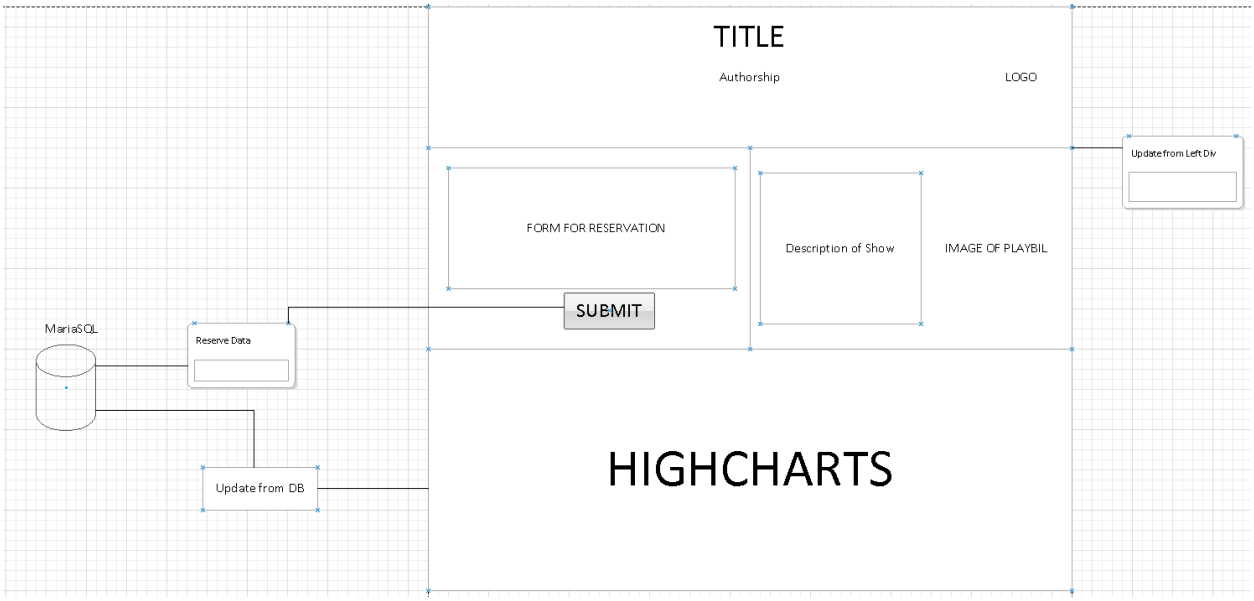
Div 1 contains a banner of the VOX group, a title and the authors of the website. It is very simple, and extends across the top of the page horizontally.

Div 2 is considered the left centered box on our website. It holds the form that the users fill out to reserve their tickets. Drop downs of the shows and their times actively update from each other as they change. A submit button calls the server and database to attempt to store the data.

Div 3 is considered the middle right box on our website. This div is actively updated given the information in Div2. This box is used to show a poster of the play selected and a small blurb about the stars/plot of the play selected.

Div 4 is the bottom div spanning the horizontal of the page. This div is reserved to show a relatively up-to-date count of the number of tickets left per show and showing. This was developed and gotten using callbacks to the server and Highcharts.

Wire Diagram



Fragments + Templates

- *layout.jade*: basic jade file. Contains the overarching structure of the HTML document, and links to the style sheets and JavaScript files.
- *index.jade*: extends *layout.jade*. Used for generating the index.html file. Only contains a header, no other content is included.
- *reservations.jade*: extends *layout.jade*, used for generating the reservations.html file. Contains all the various pieces of the document layout, including the various content blocks for each of the major four div elements, and the form and image layouts.
- *error.jade*: used by *node.js* to serve any errors generated during the parsing of the JADE template.
- *textfiles.jade*: the JADE fragment used by the application to parse the content of unformatted text files into valid HTML. Contains tags related to forming HTML paragraph tags using the text content.
- *showinfo.jade*: another JADE fragment used by the application to parse the content of unformatted text files into valid HTML. Contains tags related to forming HTML paragraph tags using the text content. The reason we made a different JADE fragment for the show related information is because it allows for easier extensibility if we want to include more types of dynamic information for shows in the future.

URL Map

Below is the map of the URLs and their functions:

- /helloworld
  - This was the python sandbox starter URL. Unused, but reserved for expansion of calling the email client via AJAX.
- /reservations.html

    - Render reservations.html (our main homepage)
- /shows
    - Gets all show times and names from the server.
- /showtimes/:fname
    - Gets all show times given a name in the URL
- /getChartData
    - Dumps all of the data from the database in the VoxShow category
- /sendconfirmation
    - Starts the Python email sender. This is not working because it isn't safe
- /addReservation
    - Adds the reservation from the client form to the database. Can throw an error if not ok parameters.
- /showInfo/:fname
    - Compile and generate text for the html items regarding the show names
- /textfiles/:fname
    - Pull all of the text files and add to the website

Sub-Applications

A separate email client was supposed to be integrated into the Node.js server, but unfortunately there was no way to safely attach it in the time allotted – the manner we found was to use Express to run a shell script that would then run the Python script that we had developed to send emails. However, we decided against doing this because arbitrary code execution on the server isn't a safe practice. The team elected to remove the feature until a better solution was attained.

Aside from the potential email client addition, the other sub applications used were Highcharts and MariaDB. Highcharts was used in the client side application to develop a simple chart of tickets remaining on the website, while MariaDB was used as a remote SQL server to store the reservation and show data in another accessible place from the server.

Listing of DOM

- Div1
    - H1
    - H3
- Container
    - Div2
        - H3
        - Div reservationInstructions
        - Form
        - Submit button
    - Div3
        - H3
        - Div infoDiv
            - p
        - Div poster

        - img
- Div4
    - Div tickets
        - Highcharts

Info Sequence

The info is designed in such a way that it has minimum span of data, but expresses just enough data for the developers. A chart of the database is described below with constraints and keys highlighted.

Figure 1: Database Design

The information is easily accessible with multiple queries defined in *sqlServer.js* inside of the *js* folder.

Odd Bugs

MariaDB does not contribute Foreign Keys as Primary Keys of the subsidiary entity, making this model ineffective at holding our promise of allowing people to have multiple reservations on multiple shows with the same set of names and email. This is breaking, and unable to be fixed reasonably without restructuring of the DB. This would be done by adding a relation in the middle called reservation which has only the keys of the *Person* and *voxShow* in it.