OBJECTIVE

Develop Single Page Application (Ticketing System) using HTML, CSS, JavaScript and Vue.js.

TIME SPENT

The application took approximately 3 days to develop with 6 hours spent each day i.e. around total of 18 hours.

DISTRIBUTION AND EXPLANATION

Following dependencies were used for creating the application apart from Vue.js:

- Bootstrap 4 CSS
- Chart.js
- Fontawesome CSS
- Vue Router

The theme of the application is taken as black.

Setup

The initial setup of the application was done using Node.js and Vue-CLI 3 in the WebStorm IDE. This took around 1 hour for setting up and installing dependencies since this was my first project using node.js.

Time taken: 1 hr

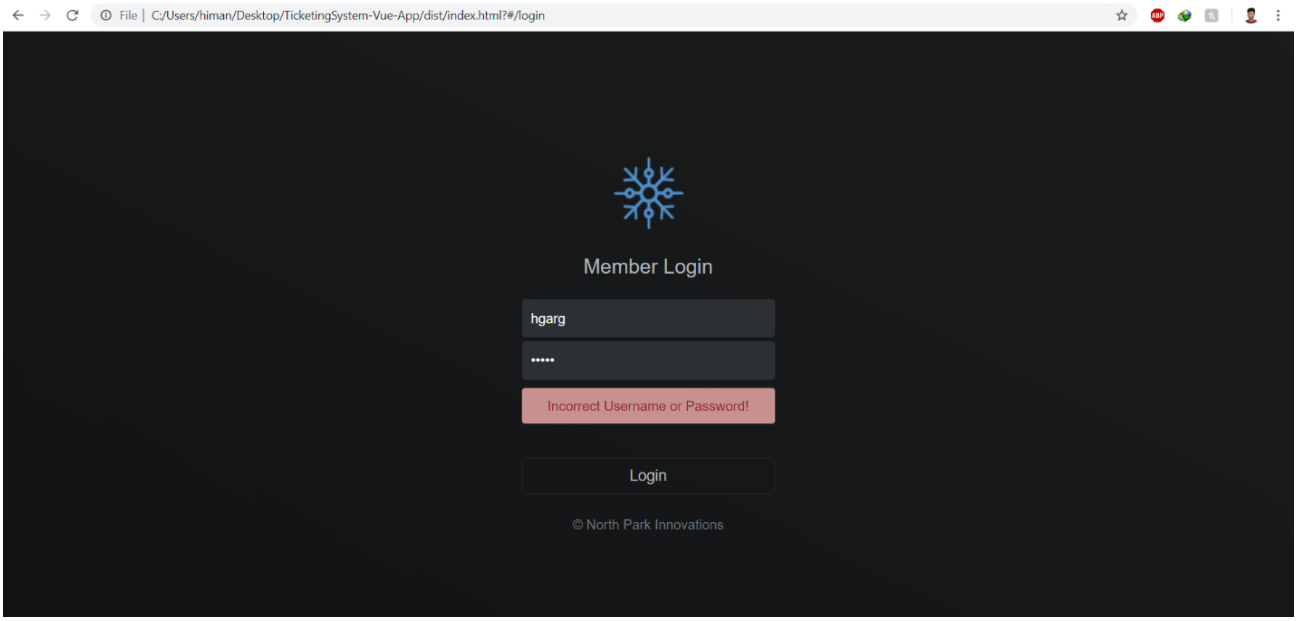
Login Page

1. First routing was setup to enable the authorisation framework where if the username and the password are correct then the login page is removed and the home page where the ticketing information is available is made visible, otherwise the user is shown an error or is redirected back to login page if he/she signs out of the application.

2. Hence the main page path which is "/" is set to "/login" which is then redirected to "/home" on correct authorization.

3. Login Page Component is created and North Park innovations logo used at the top with two HTML input boxes with bootstrap styling for username and password, and a login button at the bottom.

4. If the username and password entered are wrong and the user tries to login, a warning is shown that the username or password are incorrect which gets removed once the user tries to enter a new values. This can be seen in the following image.

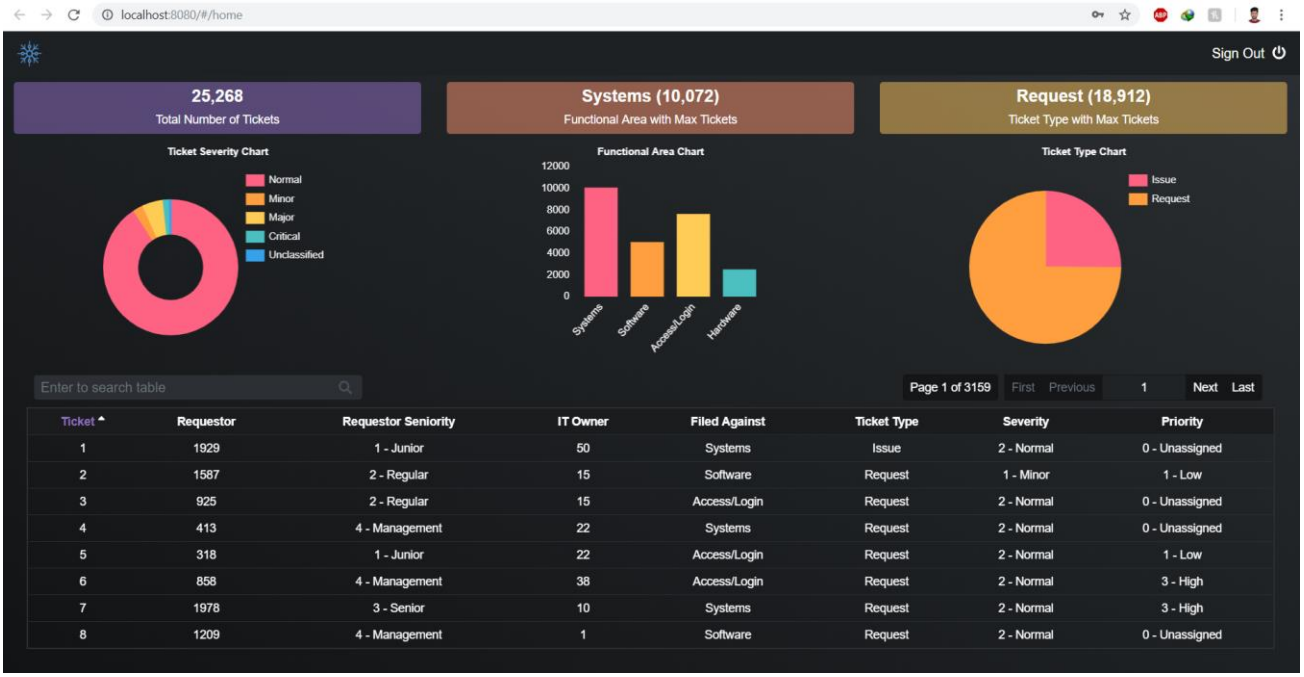


Time Taken: 3 hrs

Home Page

1. Once the User is Authenticated, the page is routed to the ticketing system homepage and the login page disappears. Here the user has the option to sign out of the application by clicking the sign out button on the top right corner of the screen in the navigation bar. This makes the homepage disappear and the user is again presented with the login page for authentication.

2. The first look of the homepage is as follows:



Ticket	Requestor	Requestor Seniority	IT Owner	Filed Against	Ticket Type	Severity	Priority
1	1929	1 - Junior	50	Systems	Issue	2 - Normal	0 - Unassigned
2	1587	2 - Regular	15	Software	Request	1 - Minor	1 - Low
3	925	2 - Regular	15	Access/Login	Request	2 - Normal	0 - Unassigned
4	413	4 - Management	22	Systems	Request	2 - Normal	0 - Unassigned
5	318	1 - Junior	22	Access/Login	Request	2 - Normal	1 - Low
6	858	4 - Management	38	Access/Login	Request	2 - Normal	3 - High
7	1978	3 - Senior	10	Systems	Request	2 - Normal	3 - High
8	1209	4 - Management	1	Software	Request	2 - Normal	0 - Unassigned

Navigation Bar

At the top of the homepage, a navigation bar is created with the logo of the company inserted on the left size and the sign out option on the right side of the bar.
Fontawesome library is used to insert the sign out icon.

Time Taken: 0.5 hr

Summary Data

After that a container is created for the main page. At the starting, a row is created with three boxes containing part of the summary.

a. The first box tells the total number of tickets that have been filed.
b. The second box tells the functional area which has the maximum number of tickets filed against it, and also tells the number of tickets.
c. The third box tells us the ticket type which has the maximum number of tickets filed against it as well as the number of tickets.

The next row contains three graphs created using the **chart.js** library. Its dependency was installed using the npm package manager.

d. The first graph is a dough-nought chart that gives us the types of severities and the count of tickets filed against each of them
e. The second graph is a bar chart that gives the different types of functional areas and the number of tickets filed against each.
f. The third graph is a pie chart of ticket type chart with number tickets filed against each type.

Time Taken: 3 hrs

Full Data

After the summary data, full data is shown in an optimized manner in a HTML table. In the table, only 8 columns out of 11 columns are shown. The process and features are as follows:

Table

A HTML table is created and the data is iterated over using v-for to populate the rows. Bootstrap is used to create a dark themed table and CSS modified to adapt to the current app theme.

Time Taken: 0.5 hr

Sorting

Each column is assigned a key and on click of each column, that columns key is passed to a function and data is sorted on that key and returned, and since it is bound to the table body, its updated automatically.

Fontawesome icon is used to show the sorting icon on the active column.

Time Taken: 1 hr

Searching

A search box is created above the table which is bound to a value inside Vue framework. When the value changes, each row or object in the data is searched for the occurrence of the search query and those rows or objects are returned which have the occurrence of that search query.

Time Taken: 1 hr

Pagination

Since we have huge amount of data we need some sort of pagination mechanism or it takes too much time for the UI to load the data.

1. A pagination bar is created above the table which shows the current page you are on and the total page.
2. Four buttons are provided “First”, ”Previous”, ”Next” and ”Last” on whose click the respective functions are performed.
3. Also, an input box is provided where user can input the page number and directly go to that page instead of clicking on the buttons multiple time to reach the page.

The challenge was to update page numbers when the data is changed on search query and also to map the page number entered by the user to the correct paged data.

Time Taken: 3 hrs

Modal

On the click of the table row corresponding to a ticket data, a modal dialogue appears where all the information about that ticket is shown in an optimized manner. The user can close the modal using the close button provided in the modal and come back to main screen.

A modal component is created and bootstrap 4 used for styling it. The component becomes visible based on a flag which gets set when the row is clicked.

Time Taken: 1 hrs

Testing

After each component creation was done, it was tested for different cases and fixed when breaking. Full testing spanning over 3 days took around **4 hrs**.