Name: Przemysław Strojek
Index number: 299760
Date: 16 VI 2020

Used technologies:

- Flask for backend, React for frontend
- Material UI for GUI
- SQLite for database
- Axios for REST requests

Application description:

Presented application is a simple chess server. The main purpose of the application is finding opponents, playing chess and watching replays from played games. The navigation is based on *Tab* component and most of the tabs follow the same structure:

```
React.useEffect(() => {

getUserList(queryString.stringify(query),setUserList,history,setRange,page,setListFragment);
},[history,page,query]);
```

getUserList is an example of function that prepares a REST request. Among its arguments there are request parameters and hooks where values would be returned.

```
<div className={classes.root}>
    <br/>
    <TextField
        size='small'
        placeholder='Name'
        variant='outlined'
        value={username}
        onChange={handleNameChange}
    />
```

Controlled text field for searching

```
    <Button
        color='primary'
        variant="contained"
        type="submit"
        onClick={handleSubmit}
    >Search</Button>
```

Search button

```
    <br/><br/>
    <Hidden xsDown>
        <Grid container>
            <Grid item xs>
                <Typography variant='h5'
                >Name</Typography>
            </Grid>
            <Grid item xs>
                <Typography variant='h5'
                >Last Seen</Typography>
            </Grid>
            <Grid item xs>
                <Typography variant='h5'
                >Elo rating</Typography>
            </Grid>
            <Grid item xs>
            </Grid>
            <br/><br/>
        </Grid>
    </Hidden>
```

Grid with labels, which is hidden on smaller screens

```
    {listFragment}
```

10 (or less) elements from the list. It is set by *handlePageChange* or *getUserList*.

```
    <Pagination
        count={range}
        shape="rounded"
        color="primary"
        page={page}
        onChange={handlePageChange}
    />
</div>
```

Page navigation:
range-number of pages
page-current page
handlePageChange-function that changes *page* and sets *listFragment* appropriately.

```
const handleSubmit=(event)=>{
    event.preventDefault();
    var q={};
    var s='';
    if(username!==''') {
        q.name = username;
        s='?'+queryString.stringify(q);
    }
    setQuery(q);
    history.push('/users'+s);
};
```

This function generates query string and redirects to appropriate URL.

```
export function getUserList
(query,setUserList,history,setRange,page,setListFragment){
    var string='';
    if(query!==''')
        string='?'+query;
    var headers={};
    headers['Content-Type']='application/x-www-form-urlencoded';
    headers['Access-Control-Allow-Origin']='*';
    if(localStorage.getItem('token')!=null)
        headers['Authorization']='Bearer '
            +localStorage.getItem('token')
    axios.get('/users'+string,{headers:headers})
        .then(res => {
            var usersList=[];
            for(var i=0;i<res['data']['list'].length;i++) {
                const link="/users/"+res['data']['list'][i][1];
                const user=res['data']['list'][i];
                usersList.push(
                    <Grid container>
                        <Grid item xs={6} sm>
                            <Typography
                                variant='body1'>
                            {user[0]}
                            </Typography>
                        </Grid>
                        <Grid item xs={6} sm>
                            <Typography variant='body1'>
                            {user[2]}
                            </Typography>
                        </Grid>
                        <Grid item xs={6} sm>
                            <Typography variant='body1'>
                            {user[3]} / {user[4]} / {user[5]}
                            </Typography>
                        </Grid>
                        <Grid item xs={6} sm>
                            <Typography variant='body1'>
                                <Link
                                    href={link}
                                    onClick={
                                        (event)=>{
                                            event.preventDefault();
                                            history.push(link)
                                        }
                                    }
                                >
                                View Profile
                                </Link>
                            </Typography>
                        </Grid>
                        <br/>
                    </Grid>
                );
            }
```

Preparing headers and body arguments (if any)

The request

Filling list of users

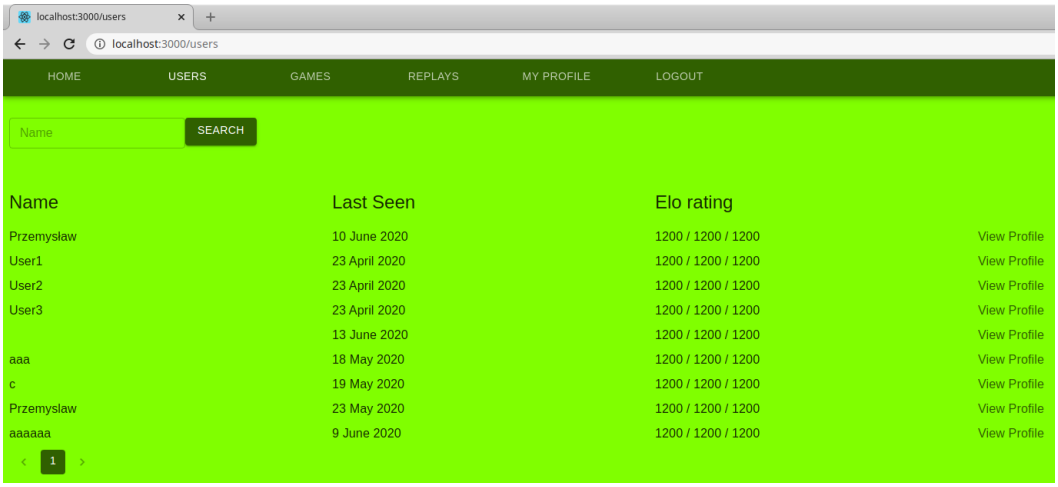
Grid contains data of a single user. Its size depends on screen width.

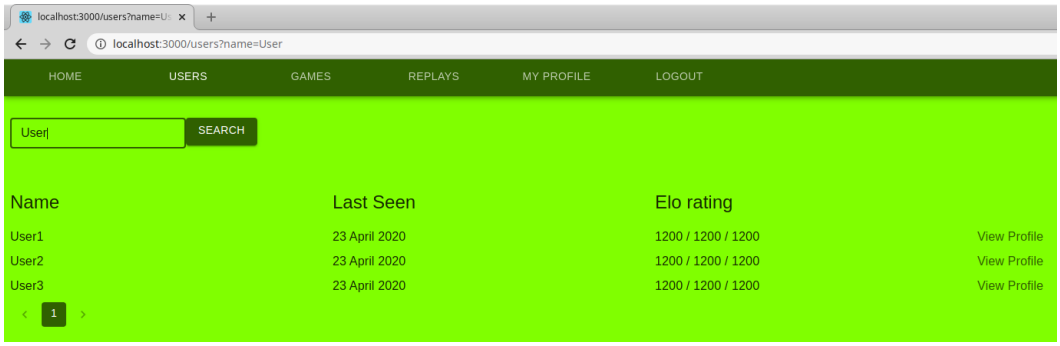
```
            setUserList(usersList);
            setRange(Math.max(1,Math.ceil(usersList.length/10)));
            setListFragment(usersList.slice((page-1)*10,page*10));
        })
        .catch(error => {
            if(error.response.status===401)
                history.push('/authorization_error');
            else history.push('/error');
        })
}
```

UserList – the entire list
Range – number of pages (length of list didived by 10, rounded up.
ListFragment – 10 elements to display.

Error handling
In case of known error redirect to appropriate page, otherwise redirect to generic error page.

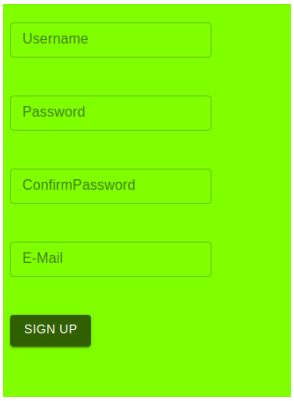
Pictures below show the result of this code.





Game offers and replays are displayed in analogic way.

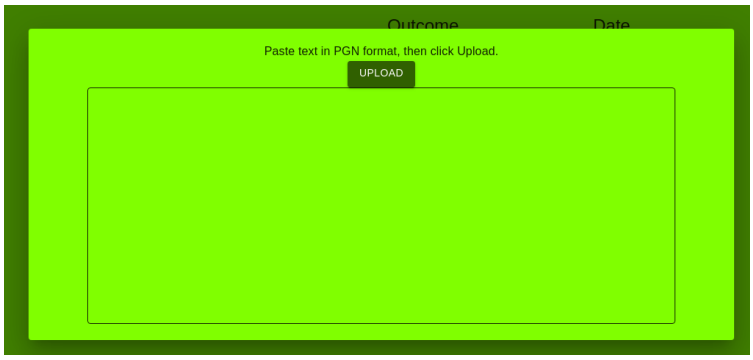
Screens below show other functionalities:



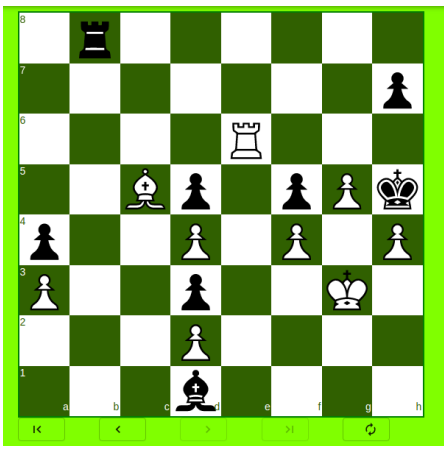
Registration form.

Viewing profile (Links for changing password and account removal are shown only when user's ID matches token identity).

Login form with password's error prop set to true.



Uploading replay.



Watching replay.