Name and Last Name:	Jacek Mstowski
Index number:	299720
Department:	Metal Engineering and Industrial Computer Science
Field of Study	Applied Computer Science
Subject:	Internet Engineering
Date:	11.09.2020

1. **Routing in project.**
   I have created new file named Routing.js. Because of routing used here the page don't need to reload everytime we wanted to go others routes.

```
export const Routing = () => {
    return (
        <>
            <Switch>
                <Route exact path = "/" component={Home}/>
                <Route path = "/reminders" component={Reminder}/>
                <Route path = "/login" component={Login}/>
                <Route path = "/register" component={Register}/>
                <Route path = "/userinfo" component={UserInfo}/>
                <Route path = "/logout" component={Logout}/>
            </Switch>
        </>
    )
}
```

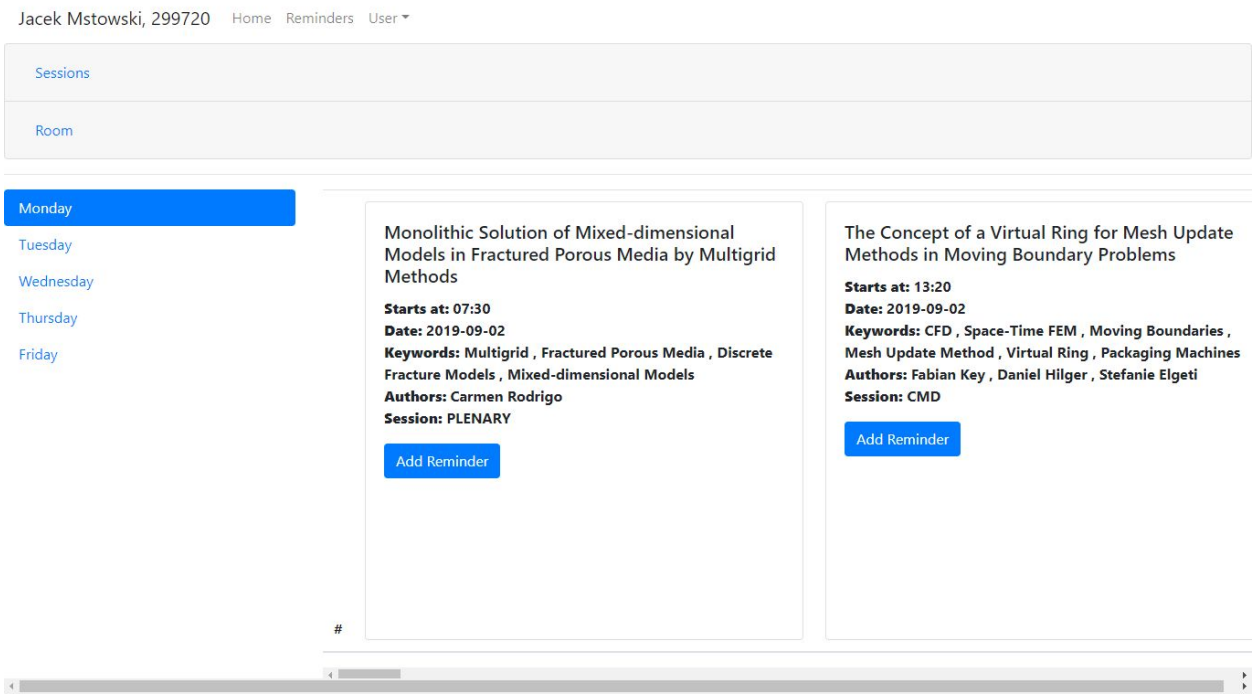
This whole element I am passing to main file App.js

```
function App() {
  return (
      <BrowserRouter>
        <Navigation/>
        <Routing/>
      </BrowserRouter>
  );
}

export default App;
```

We can see that our Navigation bar is another file to just simplify the look of main app.

```
export class Navigation extends Component {
  render() {
    return (
            <Navbar>
      <Navbar.Brand href="#home">Jacek Mstowski, 299720</Navbar.Brand>
  <Navbar.Toggle aria-controls="basic-navbar-nav" />
  <Navbar.Collapse id="basic-navbar-nav">
    <Nav className="mr-auto">
      <Nav.Link href="/">Home</Nav.Link>
      <Nav.Link href="/reminders">Reminders</Nav.Link>
      <NavDropdown title="User" id="basic-nav-dropdown">
        <NavDropdown.Item href="/login">Login</NavDropdown.Item>
        <NavDropdown.Item href="/logout">Logout</NavDropdown.Item>
        <NavDropdown.Item href="/register">Register</NavDropdown.Item>
        <NavDropdown.Item href="/userinfo">About User</NavDropdown.Item>
      </NavDropdown>
    </Nav>
            </Navbar.Collapse>
  </Navbar>
    );
  }
}
```

And the main look of my app:



2. **Communication with API.**

For communication I used the axios library. I have created configuration file for some configuration that we can do, but for now its not necessary because i have there only URL(Computed URL: http://ie2020.kisim.eu.org/api) of API. One time I used fetch to get data to check how it is working(While using fetch there was some problems with cors policy, which i can't repair).

```
export const config =  {
    baseURL: `https://ie2020.kisim.eu.org/api`
};
```

```
import axios from 'axios'
import {config} from '../configuration.js'

export default axios.create({
    baseURL: config.baseURL
});
```

With axios we can easily send queries to API to get any data we want.
There will be a few examples of using axios(Get, Put and Delete)
GET

```
export const Register = () => {
    const history = useHistory();

    const onSubmit = (event) => {
      event.preventDefault();
      let formElements = event.currentTarget.elements;
      console.log(event.currentTarget.elements.email.value);
        API.post('/users', {email: formElements.email.value, password: formElements.password.value})
            .then(response => {
                console.log('success')
                console.log('id:', response.id)

                history.push('/user')
            })
            .catch(errInfo => {

            })
    };
```

PUT

```
function remPut(event) {
    event.preventDefault();
    let formElements = event.currentTarget.elements;
    API.put(`/reminders/${rem.id}`,{ "presentationId": `${pres.id}`, "notes": formElements.note.value,
        "enabled": true }, {headers:{Authorization:`Bearer ${props.token}`}})
        .then(response=>{
            console.log("Note changed")
            setRem({"id": rem.id, "presentationId": rem.presentationId, "notes": formElements.note.value, "enabled": tr
        })
        .catch(err=>{
            console.log(err)
        })
}
```

DELETE

```
function remDelete() {
    API.delete(`/reminders/${rem.id}`, {headers:{Authorization:`Bearer ${props.token}`}})
        .then(res=>{
            console.log("Reminder Deleted");
            window.location.reload();
        })
        .catch(err=>{
            console.log("Error")
        })
}
```

3. **JWT Token storage.**

   Token is stored in cookies. There is easy access to it. We can import “js-cookies” library to manage this.

   Setting token on login:

```
console.log('Login successful.');
Cookies.set('token', response.data.token, {expires: 1});
```

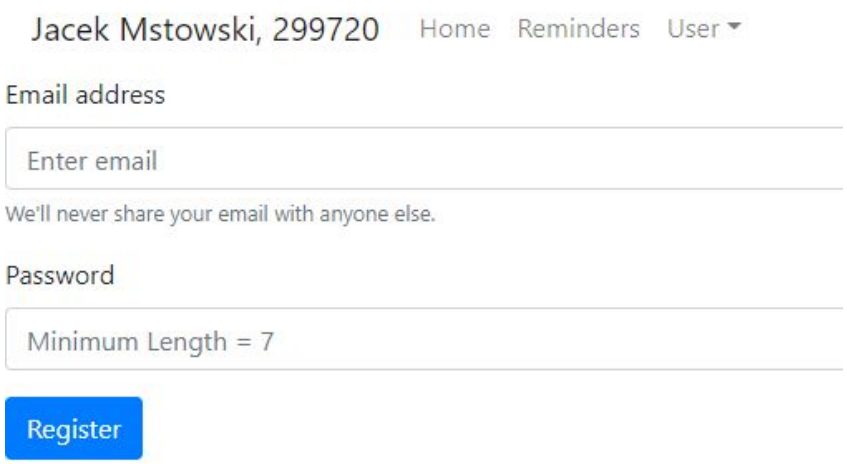
   Getting token to do something(with fetch example):

```
const token=Cookies.get('token');

if(token != null) {
    setLogged(true);
    fetch(`${config.baseURL}/users/me`, {headers:{Authorization:`Bearer ${token}`}})
        .then(res => {
            console.log('Data claimed.');
            return res.json();

        })
        .then(data =>{
            setId(data.id)
            setEmail(data.email);
            setCreatedAt(data.createdAt);
        })
        .catch(errInfo => {
            console.log('Data error: ', errInfo)
        });
}
```

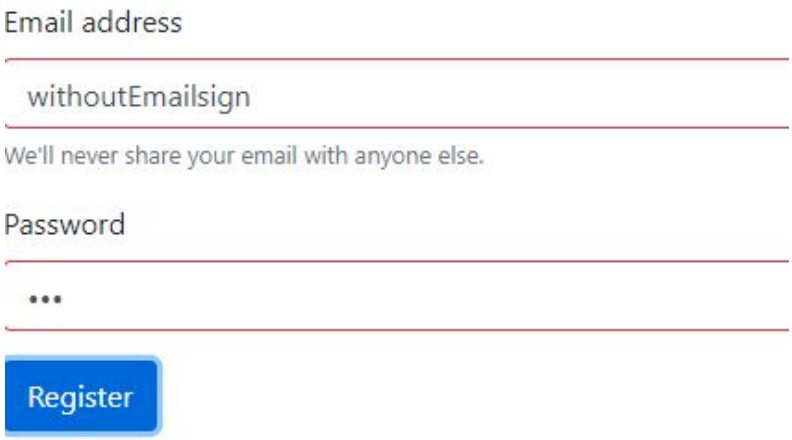
4. **Validation**

Same validation is used in Logging.

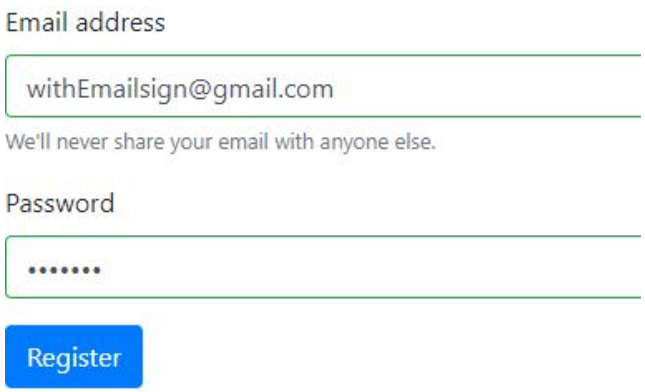
Blank Forms(inform us about requirements)



No Validate



Validate



We can see that color change if form is validate or not.
Here is a code for it

```
export const Register = () => {
    const history = useHistory();
    const [validated, setValidated] = useState(false);

    const onSubmit = (event) => {
      event.preventDefault();
      let formElements = event.currentTarget.elements;
      if (event.currentTarget.checkValidity() === false) {
        event.preventDefault();
        event.stopPropagation();
      }
      setValidated(true);
      console.log(event.currentTarget.elements.email.value);
        API.post('/users', {email: formElements.email.value, password: formElements.password.value})
            .then(response => {
                console.log('success')
                console.log('id:', response.id)

                history.push('/user')
            })
            .catch(errInfo => {

            })
    };
```

```
return (
    <Form  noValidate validated={validated} onSubmit={onSubmit}>
        <Form.Group controlId="formBasicEmail">
            <Form.Label>Email address</Form.Label>
            <Form.Control required type="email" name="email" placeholder="Enter email" />
            <Form.Text className="text-muted">
            We'll never share your email with anyone else.
            </Form.Text>
        </Form.Group>

        <Form.Group controlId="formBasicPassword">
            <Form.Label>Password</Form.Label>
            <Form.Control required type="password" minlength="7" name="password" placeholder="Minimum Length = 7"/>
        </Form.Group>
        <Button variant="primary" type="submit">
            Register
        </Button>
    </Form>
);
```

5. **Link to repository**

   https://github.com/jmstowski19/IE2020

6. Components

Project has three main components: presentations, reminders and user Presentations:

```
export const Presentation = (props) => {
    const [presentations, setPresentations] = useState([]);

    useEffect( () => {
            API.get(`/presentations?date=${props.date}`, {})
                .then(response => {
                    console.log('Presentations claimed.');
                    setPresentations(response.data);
                })
                .catch(errInfo => {
                    console.log('Presentations error: ', errInfo)
                });
        }, //eslint-disable-next-line
        []);

    return (
        <Table responsive>
        <thead>
          <tr>
            <th>#</th>
            {presentations.map(pres => (
              <th key={pres.id}><PresentationDisplay keywords={pres.keywords} authors={pres.authors} title={pres.title}
                                                     filename={pres.filename} session={pres.session} id={pres.id}
                                                     date={pres.date}/></th>
            ))}
          </tr>
        </thead>
        </Table>

    )
};
```

```
export const PresentationDisplay = (props) => {
    const token=Cookies.get('token');

    function moreInfo() {
        return(
            <>
                <b>Starts at:</b> {props.date.substr(11, 5)}<br/>
                <b>Date:</b> {props.date.substr(0, 10)}<br/>
                <b>Keywords:</b> {props.keywords.join(' , ')}<br/>
                <b>Authors:</b> {props.authors.join(' , ')}<br/>
                <b>Session:</b> {props.session}<br/>
            </>
        )
    }

    function remPost() {
        console.log(props.id)
        API.post(`/reminders`,
            {   "presentationId": props.id,
                "notes": "",
                "enabled": true},
            {headers: {Authorization: `Bearer ${token}`}})
            .then(response => {
                console.log('Reminder posted.');
            })
            .catch(errInfo => {
                console.log('Reminder post error: ', errInfo)
            });
    }

      const AddReminder= () => (
        <Form onSubmit = {remPost}>
            <Button type="submit">Add Reminder</Button>
        </Form>


      );


    return (
                        <Card style={{height: '30rem', width: "30rem"}}>
                        <Card.Body>
                          <Card.Title>{props.title}</Card.Title>
                          <Card.Text>
                          {moreInfo()}
                          </Card.Text>
                          <AddReminder/>
                        </Card.Body>
                      </Card>
```

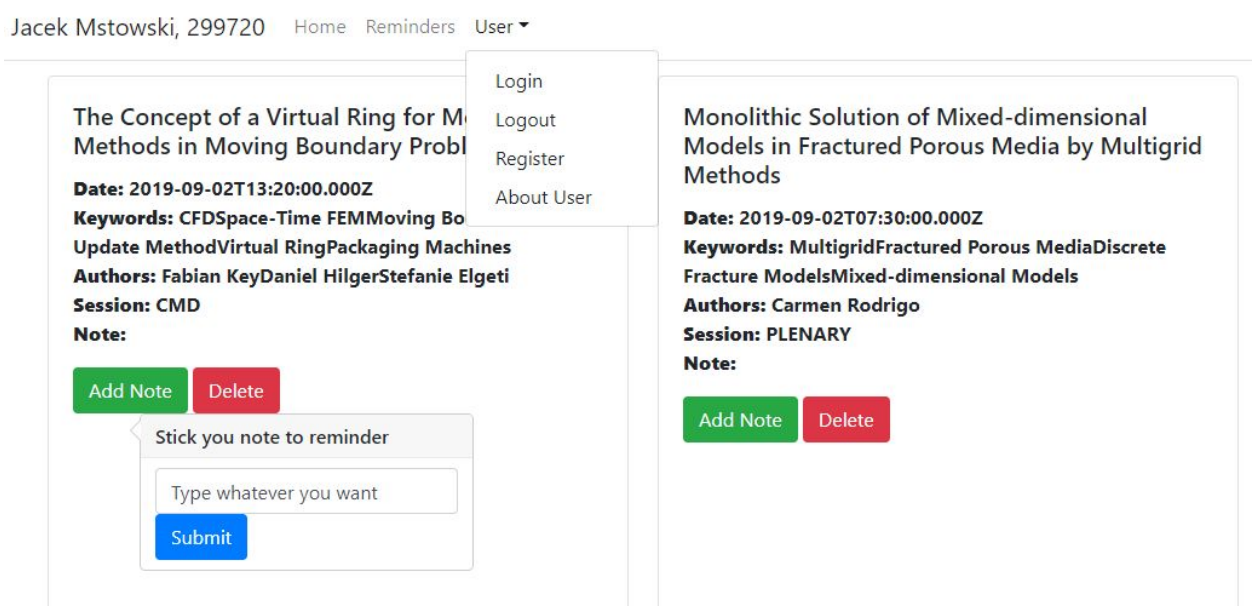
User info

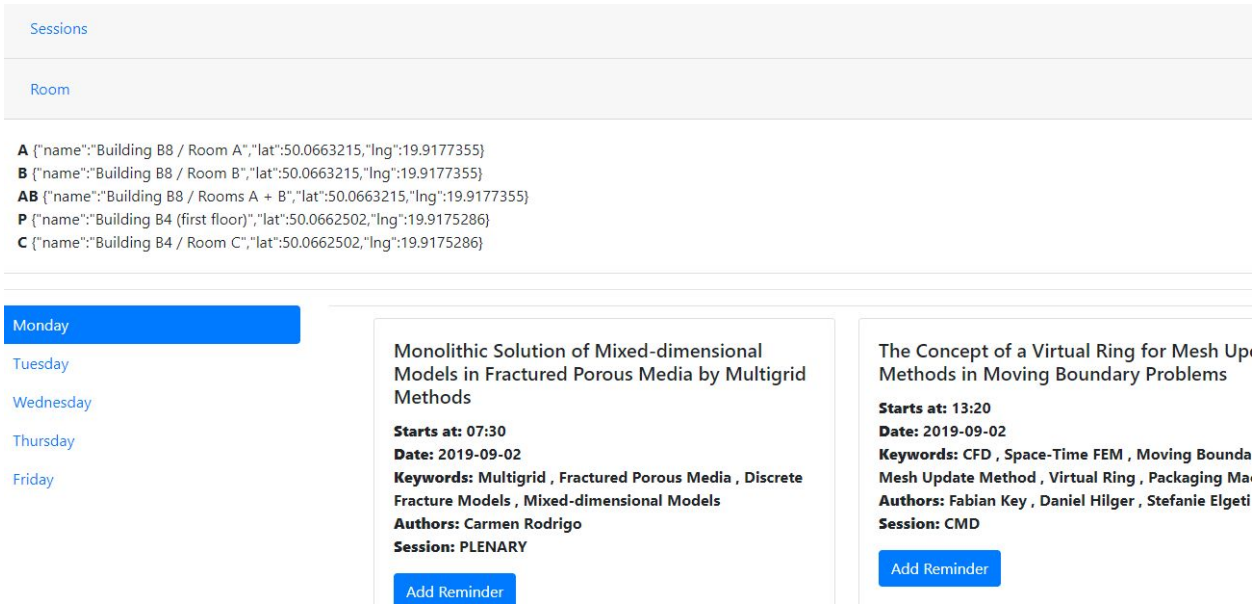
```
export const UserInfo = () => {
    const [id, setId] = useState('');
    const [email, setEmail] = useState('');
    const [createdAt, setCreatedAt] = useState('');
    const [logged, setLogged] = useState(false);

    useEffect( () => {
        const token=Cookies.get('token');

        if(token != null) {
            setLogged(true);
            fetch(`${config.baseURL}/users/me`, {headers:{Authorization:`Bearer ${token}`}})
                .then(res => {
                    console.log('Data claimed.');
                    return res.json();
                })
                .then(data =>{
                    setId(data.id)
                    setEmail(data.email);
                    setCreatedAt(data.createdAt);
                })
                .catch(errInfo => {
                    console.log('Data error: ', errInfo)
                });
        }
    }, //eslint-disable-next-line
    []);
```

Reminders:

```
export const Reminder = () => {
    const [logged, setLogged] = useState(false);
    const [reminders, setReminders] = useState([]);
    const token=Cookies.get('token')
    useEffect( () => {
        if(token != null) {
            setLogged(true);
            API.get('/reminders', {headers:{Authorization:`Bearer ${token}`}})
                .then(response => {
                    console.log('Reminders claimed.');
                    setReminders(response.data);
                })
                .catch(errInfo => {
                    console.log('Reminders error: ', errInfo)
                });
        }
        }, //eslint-disable-next-line
        []);

    if(logged)
        return (
            <Table responsive>
        <thead>
          <tr>
            <th>#</th>
            {reminders.map(rem => (
              <th key={rem.id}><ReminderDisplay presId={rem.presentationId} remId={rem.id} token={token}/></th>
            ))}
          </tr>
        </thead>
        </Table>
        );
    else
        return (
            <>The user is not logged.</>
```

7. **State**
   States are storage in api memory. I used react hooks to manage all of states in components.
8. **UI Library**
   I used the bootstrap for creating the interface. This library comes with some standard solutions which i used. I think that the look of the page is minimalistic and readable.
   Some Examples:


Jacek Mstowski, 299720
Home
Reminders
User
Login
Logout
Register
About User
The Concept of a Virtual Ring for M
Methods in Moving Boundary Probl
Date: 2019-09-02T13:20:00.000Z
Keywords: CFDSpace-Time FEMMoving Bo
Update MethodVirtual RingPackaging Machines
Authors: Fabian KeyDaniel HilgerStefanie Elgeti
Session: CMD
Note:
Add Note
Delete
Stick you note to reminder
Type whatever you want
Submit
Monolithic Solution of Mixed-dimensional Models in Fractured Porous Media by Multigrid Methods
Date: 2019-09-02T07:30:00.000Z
Keywords: MultigridFractured Porous MediaDiscrete Fracture ModelsMixed-dimensional Models
Authors: Carmen Rodrigo
Session: PLENARY
Note:
Add Note
Delete


Sessions
Room
A {"name":"Building B8 / Room A","lat":50.0663215,"lng":19.9177355}
B {"name":"Building B8 / Room B","lat":50.0663215,"lng":19.9177355}
AB {"name":"Building B8 / Rooms A + B","lat":50.0663215,"lng":19.9177355}
P {"name":"Building B4 (first floor)","lat":50.0662502,"lng":19.9175286}
C {"name":"Building B4 / Room C","lat":50.0662502,"lng":19.9175286}
Monday
Tuesday
Wednesday
Thursday
Friday
Monolithic Solution of Mixed-dimensional Models in Fractured Porous Media by Multigrid Methods
Starts at: 07:30
Date: 2019-09-02
Keywords: Multigrid , Fractured Porous Media , Discrete Fracture Models , Mixed-dimensional Models
Authors: Carmen Rodrigo
Session: PLENARY
Add Reminder
The Concept of a Virtual Ring for Mesh Up
Methods in Moving Boundary Problems
Starts at: 13:20
Date: 2019-09-02
Keywords: CFD , Space-Time FEM , Moving Bounda
Mesh Update Method , Virtual Ring , Packaging Ma
Authors: Fabian Key , Daniel Hilger , Stefanie Elgeti
Session: CMD
Add Reminder

9. Others

As we can see app compiled without any warning.