# c More about styles

In part 2 we examined two different ways of adding styles to our application: the old-school single CSS file and inline-styles. In this part we will take a look at a few other ways.

## Ready-made UI libraries

One approach to defining styles for an application is to use a ready-made "UI framework".

One of the first widely popular UI frameworks was the Bootstrap toolkit created by Twitter, that may still be the most popular framework. Recently there has been an explosion in the number of new UI frameworks that have entered the arena. In fact, the selection is so vast that there is little hope of creating an exhaustive list of options.

Many UI frameworks provide developers of web applications with ready-made themes and "components" like buttons, menus, and tables. We write components in quotes, because in this context we are not talking about React components. Usually UI frameworks are used by including the CSS stylesheets and JavaScript code of the framework in the application.

There are many UI frameworks that have React-friendly versions, where the framework's "components" have been transformed into React components. There are a few different React versions of Bootstrap like reactstrap and react-bootstrap.

Next we will take a closer look at two UI frameworks, Bootstrap and MaterialUI. We will use both frameworks to add similar styles to the application we made in the React-router section of the course material.

## React Bootstrap

Let's start by taking a look at Bootstrap with the help of the react-bootstrap package.

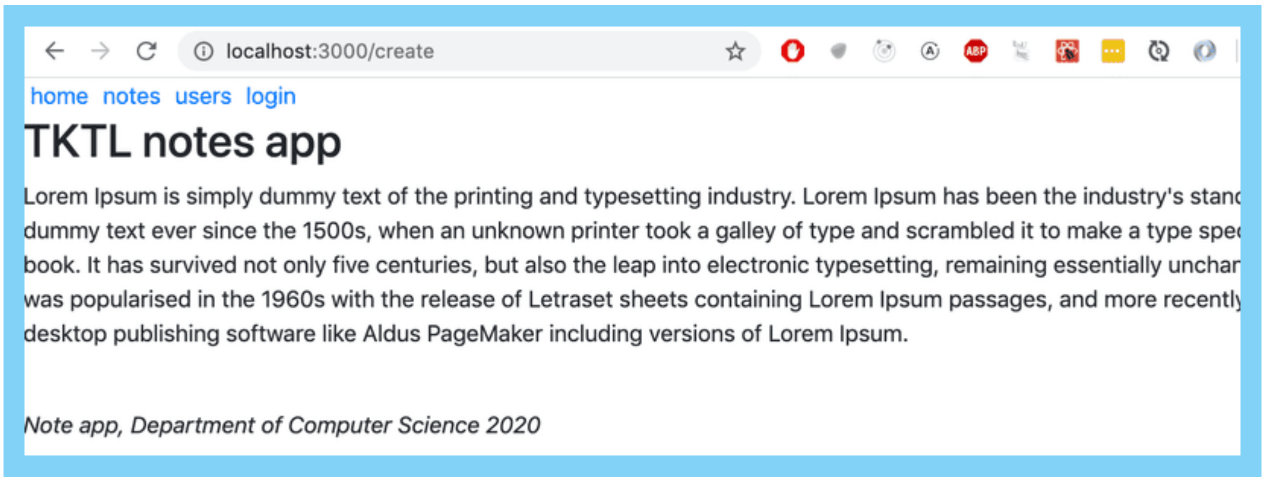
Let's install the package with the command:

```
npm install react-bootstrap
```

Then let's add a link for loading the CSS stylesheet for Bootstrap inside of the *head* tag in the *public/index.html* file of the application:

```
<head>
  <link
    rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
    integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
    crossorigin="anonymous"
  />
  // ...
</head>
```

When we reload the application, we notice that it already looks a bit more stylish:



TKTL notes app

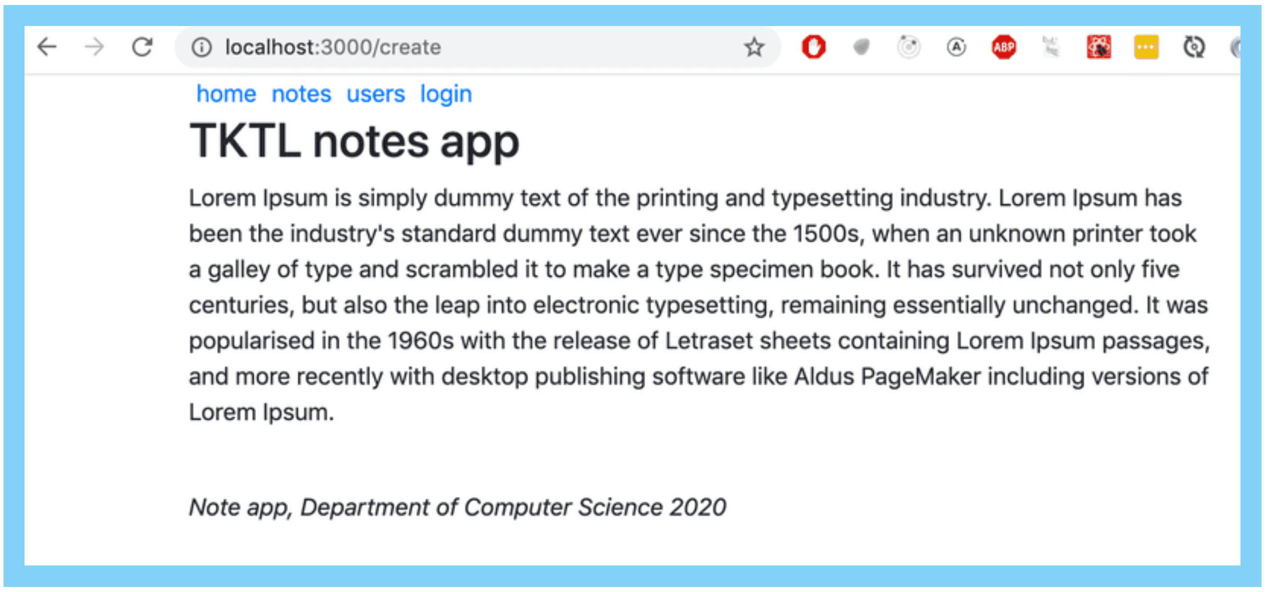
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's stanc dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type spec book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchar was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

*Note app, Department of Computer Science 2020*

In Bootstrap, all of the contents of the application are typically rendered inside of a container. In practice this is accomplished by giving the root `div` element of the application the `container` class attribute:

```
const App = () => {
  // ...

  return (
    <div className="container">
      // ...
    </div>
  )
}
```

We notice that this already has an effect on the appearance of the application. The content is no longer as close to the edges of the browser as it was earlier:



TKTL notes app

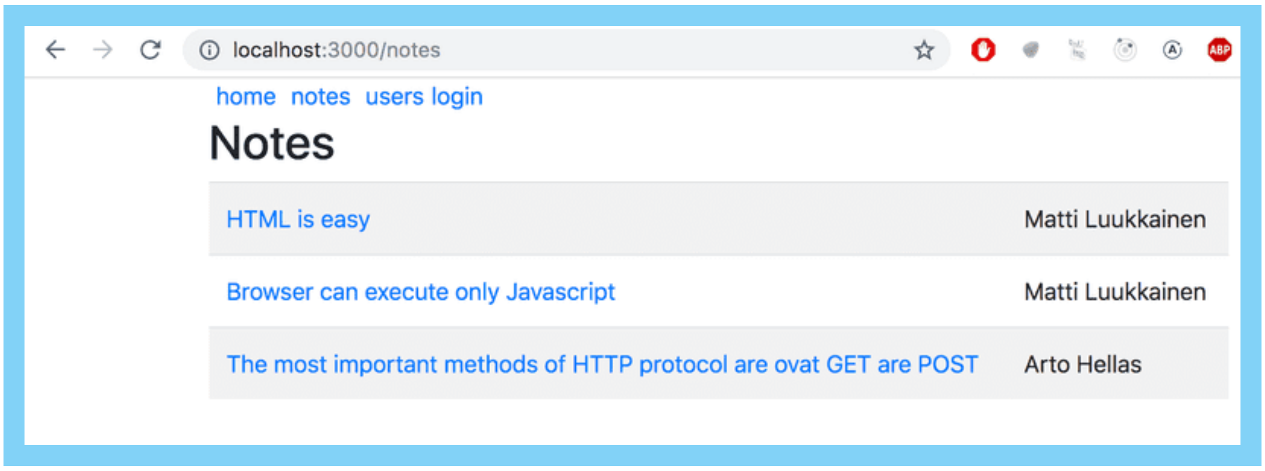
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

*Note app, Department of Computer Science 2020*

Next, let's make some changes to the *Notes* component, so that it renders the list of notes as a table. React Bootstrap provides a built-in Table component for this purpose, so there is no need to define CSS classes separately.

```
const Notes = (props) => (
  <div>
    <h2>Notes</h2>
    <Table striped>
      <tbody>
        {props.notes.map(note =>
          <tr key={note.id}>
            <td>
              <Link to={`/notes/${note.id}`}>
                {note.content}
              </Link>
            </td>
            <td>
              {note.user}
            </td>
          </tr>
        )}
      </tbody>
    </Table>
  </div>
)
```

The appearance of the application is quite stylish:


localhost:3000/notes

home notes users login

Notes

| | |
|---|---|
| HTML is easy | Matti Luukkainen |
| Browser can execute only Javascript | Matti Luukkainen |
| The most important methods of HTTP protocol are ovat GET are POST | Arto Hellas |

Notice that the React Bootstrap components have to be imported separately from the library as shown below:

```
import { Table } from 'react-bootstrap'
```

### Forms

Let's improve the form in the *Login* view with the help of Bootstrap forms.

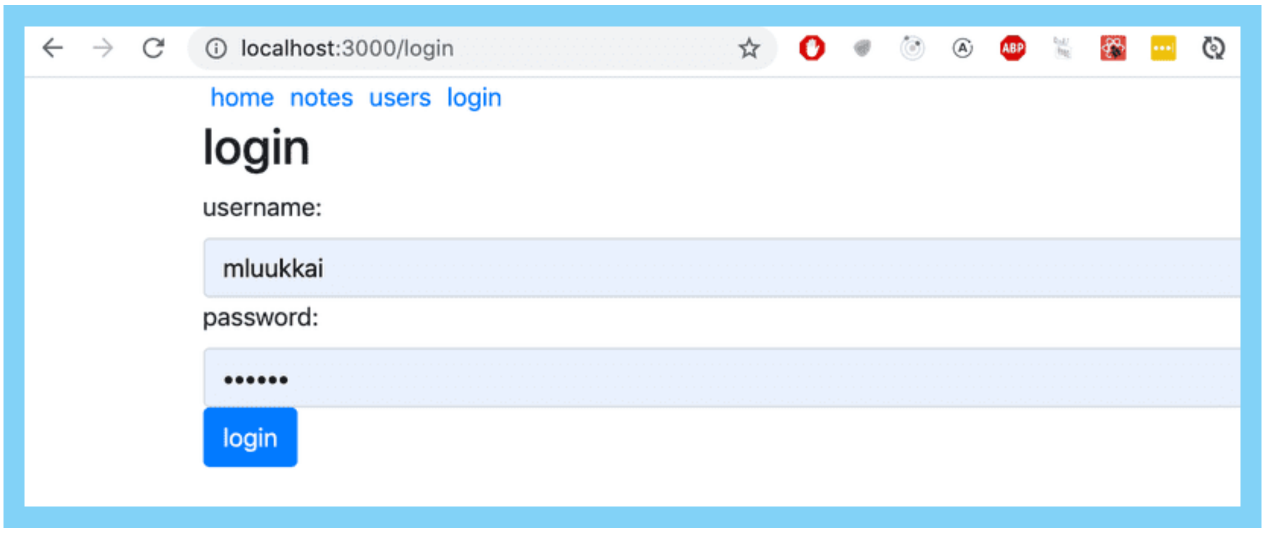
React Bootstrap provides built-in components for creating forms (although the documentation for them is slightly lacking):

```
let Login = (props) => {
  // ...
  return (
    <div>
      <h2>login</h2>
      <Form onSubmit={onSubmit}>
        <Form.Group>
          <Form.Label>username:</Form.Label>
          <Form.Control
            type="text"
            name="username"
          />
          <Form.Label>password:</Form.Label>
          <Form.Control
            type="password"
          />
          <Button variant="primary" type="submit">
            login
          </Button>
        </Form.Group>
      </Form>
    </div>
)}
```

The number of components we need to import increases:

```
import { Table, Form, Button } from 'react-bootstrap'
```

After switching over to the Bootstrap form, our improved application looks like this:



localhost:3000/login

home notes users login

login

username:

mluukkai

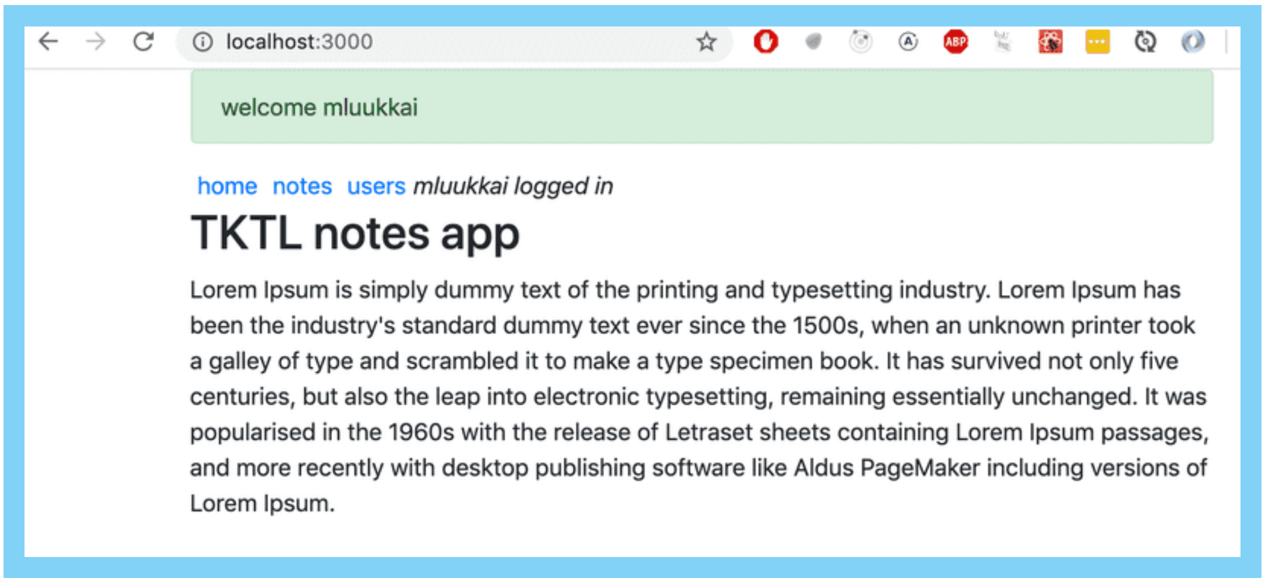
password:

••••••

login

### Notification

Now that the login form is in better shape, let's take a look at improving our application's notifications:



localhost:3000

welcome mluukkai

home notes users *mluukkai logged in*

TKTL notes app

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Let's add a message for the notification when a user logs in to the application. We will store it in the `message` variable in the *App* component's state:

```
const App = () => {
  const [notes, setNotes] = useState([
    // ...
  ])

  const [user, setUser] = useState(null)
  const [message, setMessage] = useState(null)

  const login = (user) => {
    setUser(user)
    setMessage(`welcome ${user}`)
    setTimeout(() => {
      setMessage(null)
    }, 10000)
  }
  // ...
}
```

We will render the message as a Bootstrap Alert component. Once again, the React Bootstrap library provides us with a matching React component:

```
<div className="container">
  {(message &&
    <Alert variant="success">
      {message}
    </Alert>
  )}
  // ...
</div>
```

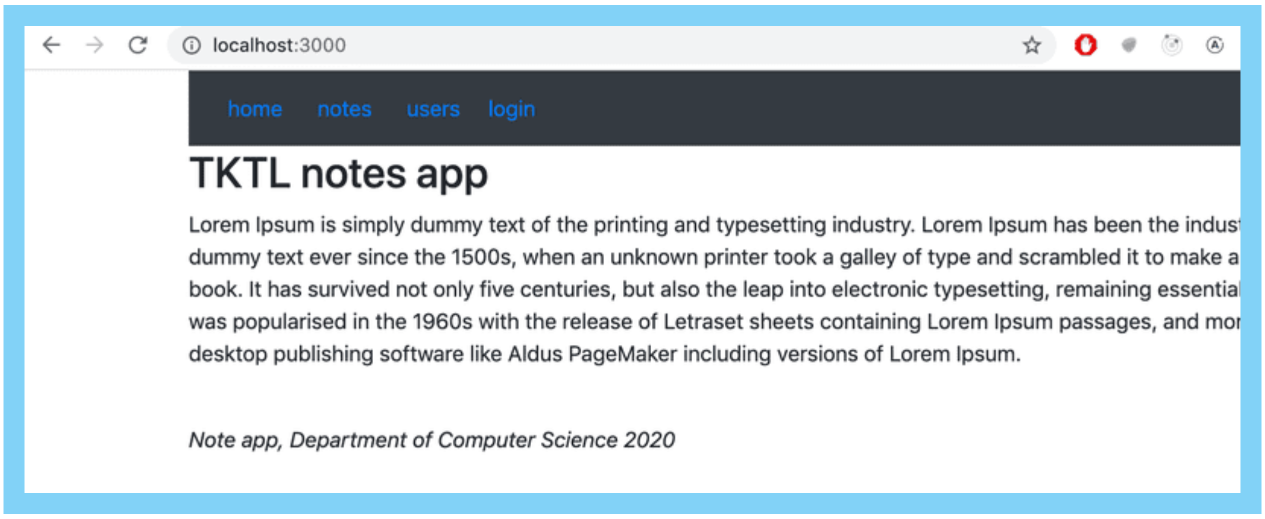
### Navigation structure

Lastly, let's alter the application's navigation menu to use Bootstrap's Navbar component. The React Bootstrap library provides us with matching built-in components. Through trial and error, we end up with a working solution in spite of the cryptic documentation:

```
<Navbar collapseOnSelect expand="lg" bg="dark" variant="dark">
  <Navbar.Toggle aria-controls="responsive-navbar-nav" />
  <Navbar.Collapse id="responsive-navbar-nav">
    <Nav className="mr-auto">
      <Nav.Link href="#" as="span">
        <Link style={padding} to="/">home</Link>
      </Nav.Link>
      <Nav.Link href="#" as="span">
        <Link style={padding} to="/notes">notes</Link>
      </Nav.Link>
      <Nav.Link href="#" as="span">
        <Link style={padding} to="/users">users</Link>
      </Nav.Link>
      <Nav.Link href="#" as="span">
```

```
          {user
            ? <em style={padding}>{user} logged in</em>
            : <Link style={padding} to="/login">login</Link>
          }
        </Nav.Link>
        </Nav>
      </Navbar.Collapse>
    </Navbar>
```

The resulting layout has a very clean and pleasing appearance:



localhost:3000

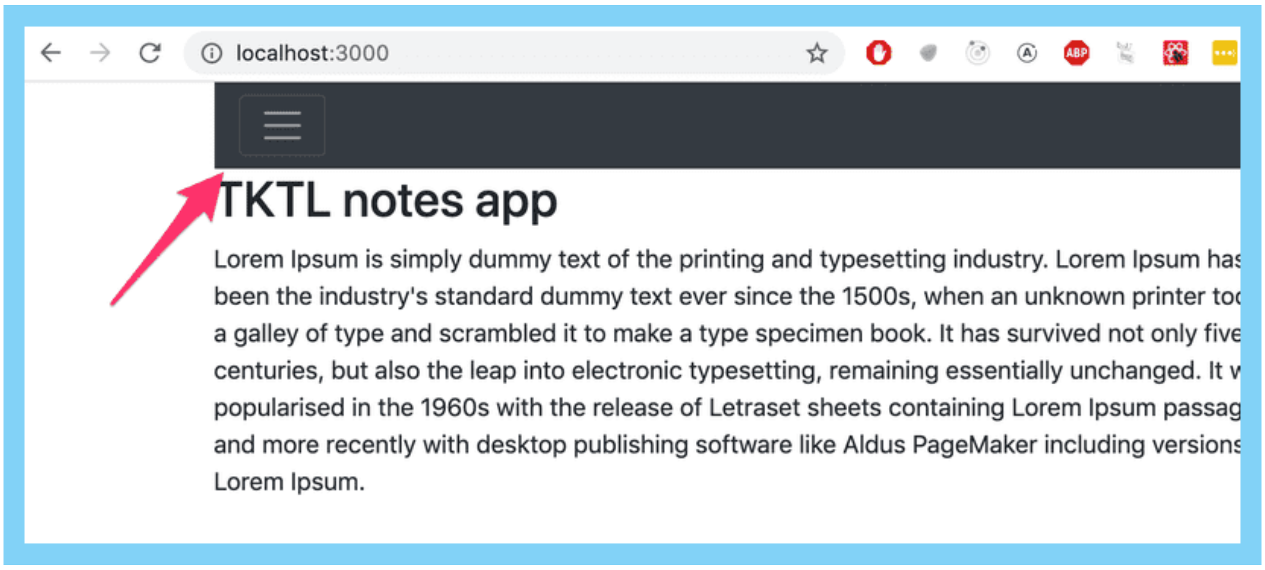
home notes users login

TKTL notes app

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the indus
dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a
book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentia
was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and mo
desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

*Note app, Department of Computer Science 2020*

If the viewport of the browser is narrowed, we notice that the menu "collapses" and it can be expanded by clicking the "hamburger" button:



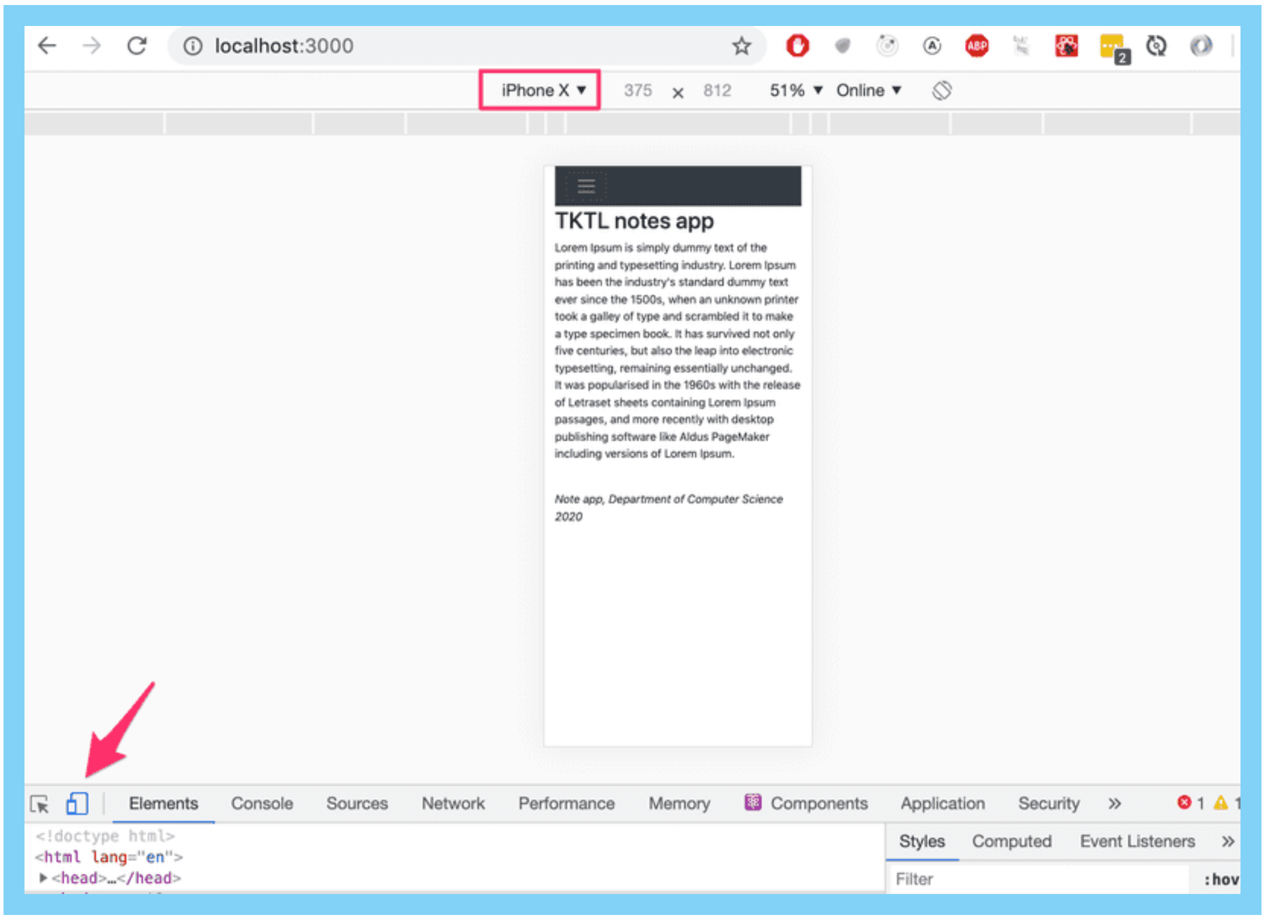
localhost:3000

TKTL notes app

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has
been the industry's standard dummy text ever since the 1500s, when an unknown printer to
a galley of type and scrambled it to make a type specimen book. It has survived not only five
centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It w
popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passag
and more recently with desktop publishing software like Aldus PageMaker including versions
Lorem Ipsum.

Bootstrap and a large majority of existing UI frameworks produce responsive designs, meaning that the resulting applications render well on a variety of different screen sizes.

Chrome developer tools makes it possible to simulate using our application in the browser of different mobile clients:


localhost:3000
iPhone X ▾ 375 × 812 51% ▾ Online ▾
Elements Console Sources Network Performance Memory Components Application Security »
Styles Computed Event Listeners »
Filter :hov

TKTL notes app

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

*Note app, Department of Computer Science 2020*

```
<!doctype html>
<html lang="en">
▶<head>…</head>
```

You can find the complete code for the application here.

## Material UI

As our second example we will look into the MaterialUI React library, which implements the Material design visual language developed by Google.

Install the library with the command

```
npm install @material-ui/core
```

Then add the following line to the *head* tag in the *public/index.html* file. The line loads Google's font Roboto.

```
<head>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=
  // ...
</head>
```

Now let's use MaterialUI to do the same modifications to the code we did earlier with bootstrap.

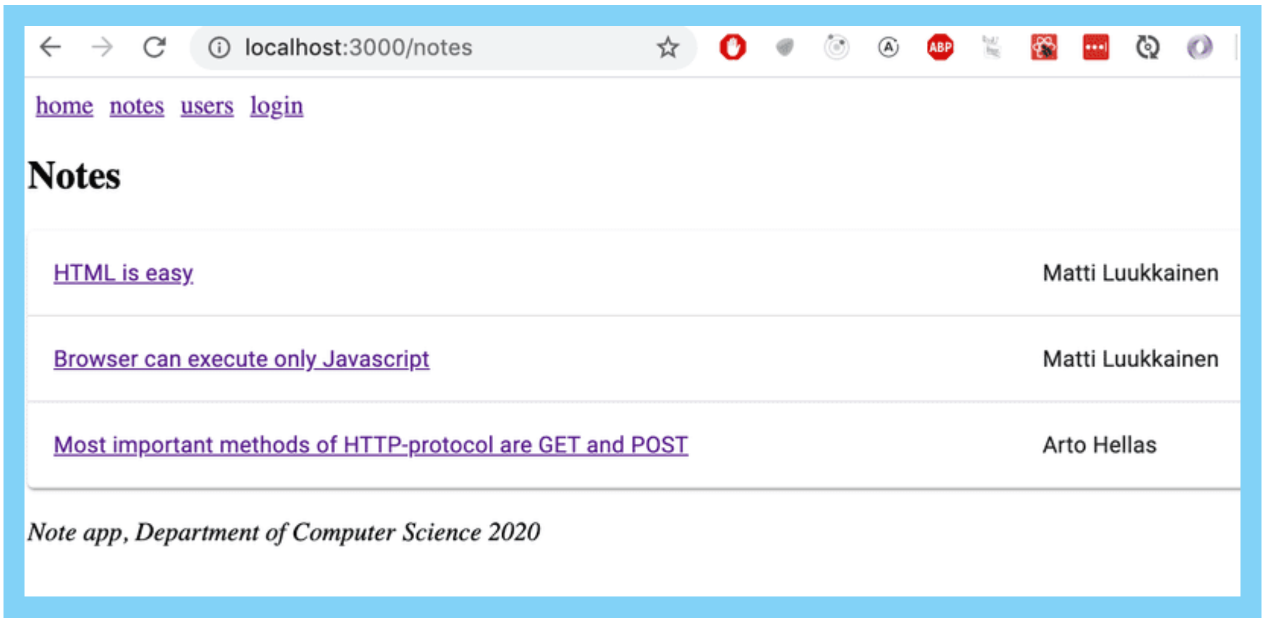
Render the contents of the whole application within a Container:

```
import Container from '@material-ui/core/Container'

const App = () => {
  // ...
  return (
    <Container>
      // ...
    </Container>
  )
}
```

Let's start with the *Notes* component. We'll render the list of notes as a table:

```
const Notes = ({notes}) => (
  <div>
    <h2>Notes</h2>

    <TableContainer component={Paper}>
      <Table>
        <TableBody>
          {notes.map(note => (
            <TableRow key={note.id}>
              <TableCell>
                <Link to={`/notes/${note.id}`}>{note.content}</Link>
              </TableCell>
              <TableCell>
                {note.user}
              </TableCell>
            </TableRow>
          ))}
        </TableBody>
      </Table>
    </TableContainer>
  </div>
)
```

The table looks like so:



Notes

| | |
|---|---|
| HTML is easy | Matti Luukkainen |
| Browser can execute only Javascript | Matti Luukkainen |
| Most important methods of HTTP-protocol are GET and POST | Arto Hellas |

*Note app, Department of Computer Science 2020*

One less pleasant feature of Material UI is, that each component has to be imported separately. The import list for the notes page is quite long:

```
import {
  Container,
  Table,
  TableBody,
  TableCell,
  TableContainer,
  TableRow,
  Paper,
} from '@material-ui/core'
```

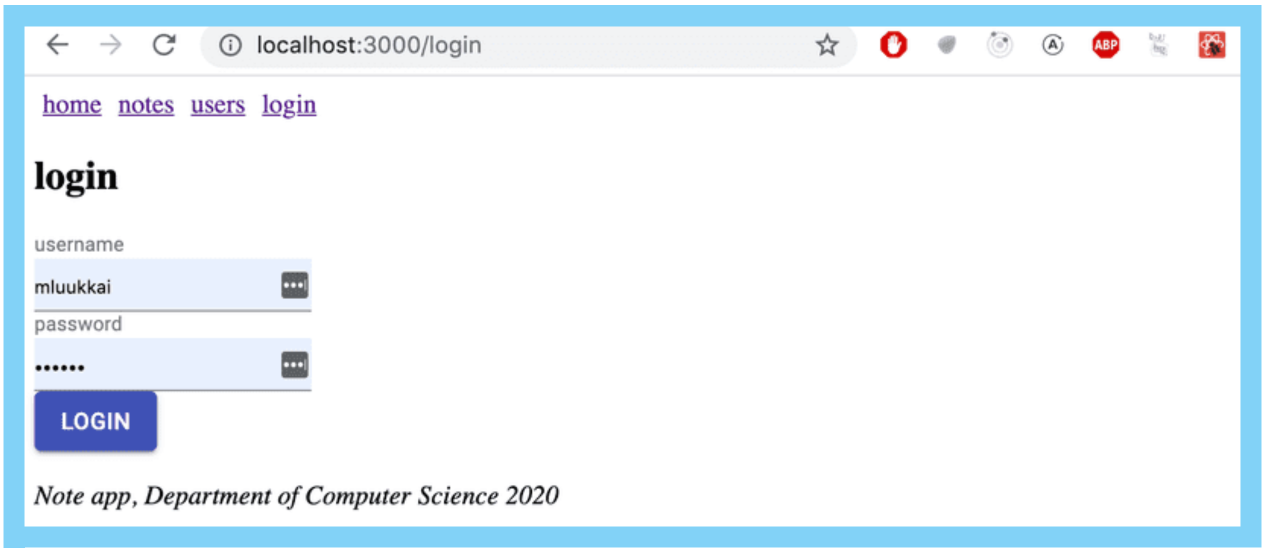
## Form

Next let's make the login form in the *Login* view better using the TextField and Button components:

```
const Login = (props) => {
  const history = useHistory()

  const onSubmit = (event) => {
    event.preventDefault()
    props.onLogin('mluukkai')
    history.push('/')
  }

  return (
    <div>
      <h2>login</h2>
      <form onSubmit={onSubmit}>
        <div>
```

```
          <TextField label="username" />
        </div>
        <div>
          <TextField  label="password" type='password' />
        </div>
        <div>
          <Button variant="contained" color="primary" type="submit">
            login
          </Button>
        </div>
      </form>
    </div>
  )
}
```

The end result is:



MaterialUI, unlike Bootstrap, does not provide a component for the form itself. The form here is an ordinary HTML `form` element.

Remember to import all the components used in the form.

### Notification

The notification displayed on log in can be done using the `Alert` component, which is quite similiar to bootstrap's equivalent component:

```
<div>
  {(message &&
    <Alert severity="success">
      {message}
    </Alert>
  )}
</div>
```

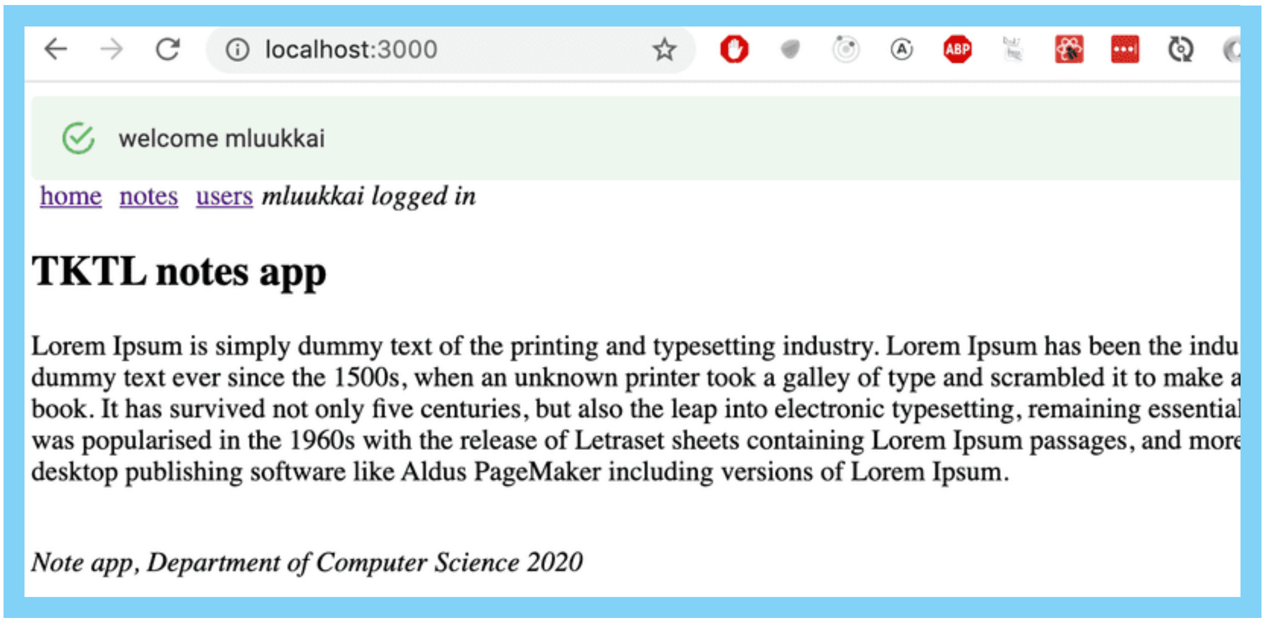
The Alert component is not yet included in the MaterialUI core package, so we have to install the lab package to use it:

```
npm install @material-ui/lab
```

Then we can import the component like so

```
import { Alert } from '@material-ui/lab'
```

Alert is quite stylish:



localhost:3000

welcome mluukkai

home notes users *mluukkai logged in*

**TKTL notes app**

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the indu
dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a
book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essential
was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more
desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

*Note app, Department of Computer Science 2020*

## Navigation structure

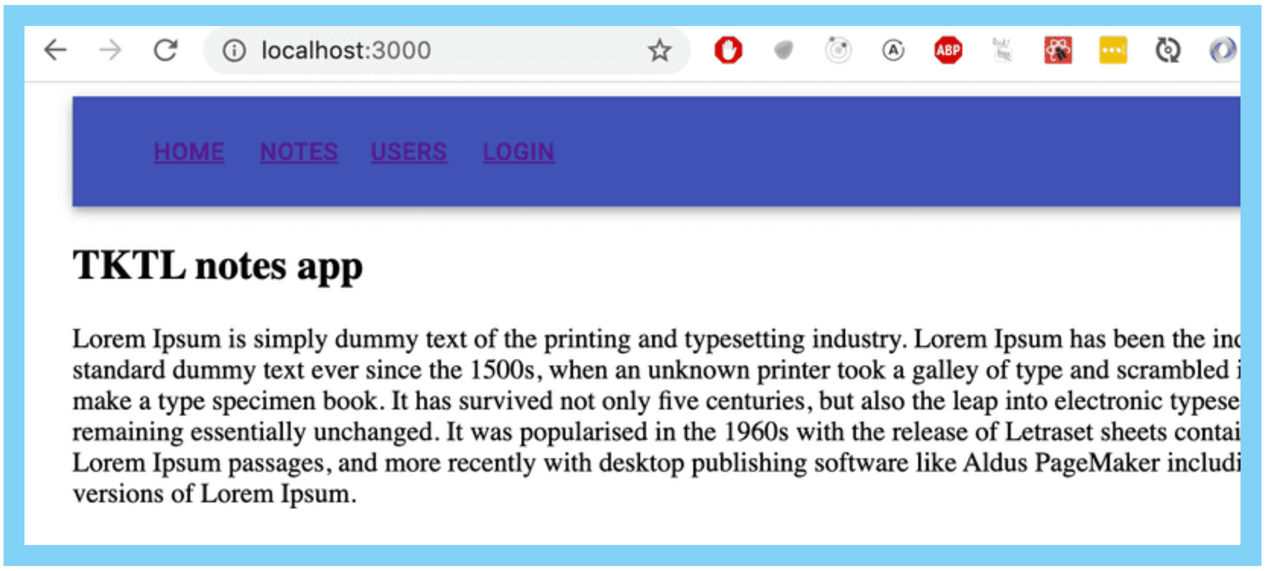
We can implement navigation using the AppBar component.

If we use the example code from the documentation

```
<AppBar position="static">
  <Toolbar>
    <IconButton edge="start" color="inherit" aria-label="menu">
    </IconButton>
    <Button color="inherit">
      <Link to="/">home</Link>
    </Button>
    <Button color="inherit">
      <Link to="/notes">notes</Link>
    </Button>
```

```
      <Button color="inherit">
        <Link to="/users">users</Link>
      </Button>
      <Button color="inherit">
        {user
          ? <em>{user} logged in</em>
          : <Link to="/login">login</Link>
        }
      </Button>
    </Toolbar>
  </AppBar>
```

we do get working navigation, but it could look better



TKTL notes app

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the inc
standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled i
make a type specimen book. It has survived not only five centuries, but also the leap into electronic typese
remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets contai
Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker includi
versions of Lorem Ipsum.

We can find a better way from the documentation. We can use component props to define how the root element of a MaterialUI component is rendered.

By defining

```
<Button color="inherit" component={Link} to="/">
  home
</Button>
```

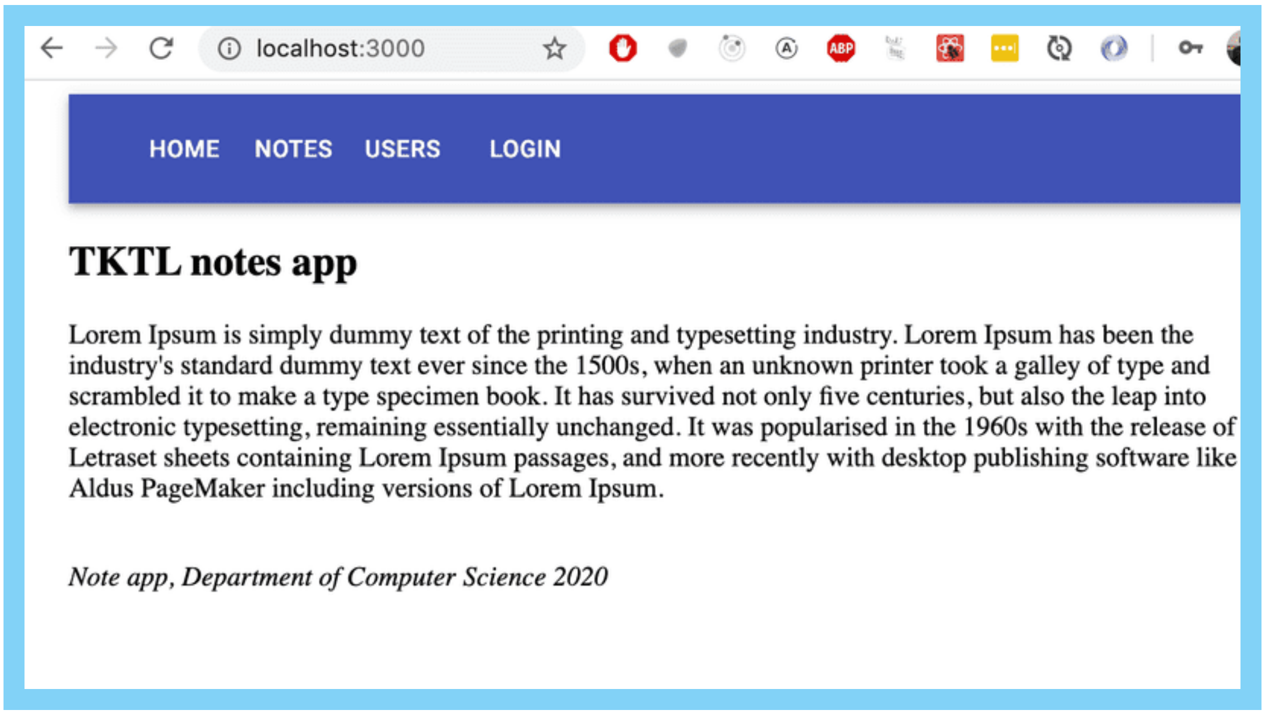
the `Button` component is rendered so, that its root component is react-router-dom `Link` which receives its path as prop field `to`.

The code for the navigation bar is the following

```
<AppBar position="static">
  <Toolbar>
    <Button color="inherit" component={Link} to="/">
```

```
        home
      </Button>
      <Button color="inherit" component={Link} to="/notes">
        notes
      </Button>
      <Button color="inherit" component={Link} to="/users">
        users
      </Button>
      {user
        ? <em>{user} logged in</em>
        : <Button color="inherit" component={Link} to="/login">
            login
          </Button>
      }
    </Toolbar>
  </AppBar>
```

and it looks like we want it to



TKTL notes app

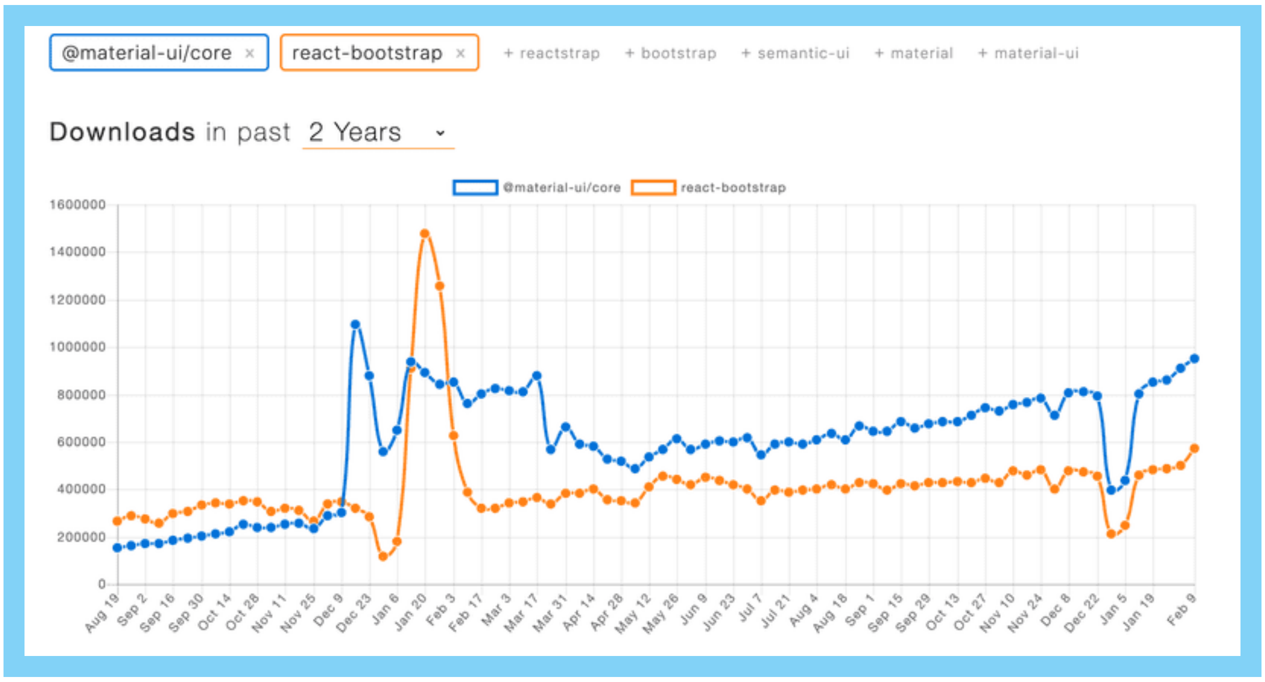
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

*Note app, Department of Computer Science 2020*

The code of the application can be found from here.

## Closing thoughts

The difference between react-bootstrap and MaterialUI is not big. It's up to you which one you find better looking. I myself have not used MaterialUI a lot, but my first impressions are positive. Its documentation is a bit better than react-bootstrap's. According to https://www.npmtrends.com/ which tracks the popularity of different npm-libraries MaterialUI passed react-bootstrap in popularity at the end of 2018:



In the two previous examples, we used the UI frameworks with the help of React-integration libraries.

Instead of using the React Bootstrap library, we could have just as well used Bootstrap directly by defining CSS classes to our application's HTML elements. Instead of defining the table with the *Table* component:

```
<Table striped>
  // ...
</Table>
```

We could have used a regular HTML *table* and added the required CSS class:

```
<table className="table striped">
  // ...
</table>
```

The benefit of using the React Bootstrap library is not that evident from this example.

In addition to making the frontend code more compact and readable, another benefit of using React UI framework libraries is that they include the JavaScript that is needed to make specific components work. Some Bootstrap components require a few unpleasant JavaScript dependencies that we would prefer not to include in our React applications.

Some potential downsides to using UI frameworks through integration libraries instead of using them "directly", are that integration libraries may have unstable API's and poor documentation. The situation with Semantic UI React is a lot better than with many other UI frameworks, as it is an official React integration library.

There is also the question of whether or not UI framework libraries should be used in the first place. It is up to everyone to form their own opinion, but for people lacking knowledge in CSS and web design they are very useful tools.

## Other UI frameworks

Here are some other UI frameworks for your consideration. If you do not see your favorite UI framework in the list, please make a pull request to the course material.

- https://bulma.io/
- https://ant.design/
- https://get.foundation/
- https://chakra-ui.com/
- https://tailwindcss.com/
- https://semantic-ui.com/

## Styled components

There are also other ways of styling React applications that we have not yet taken a look at.

The styled components library offers an interesting approach for defining styles through tagged template literals that were introduced in ES6.

Let's make a few changes to the styles of our application with the help of styled components. First, install the package with the command:

```
npm install styled-components
```

Then let's define two components with styles:

```
import styled from 'styled-components'

const Button = styled.button`
  background: Bisque;
  font-size: 1em;
  margin: 1em;
  padding: 0.25em 1em;
  border: 2px solid Chocolate;
  border-radius: 3px;
`

const Input = styled.input`
  margin: 0.25em;
`
```

The code above creates styled versions of the *button* and *input* HTML elements and then assigns them to the *Button* and *Input* variables.

The syntax for defining the styles is quite interesting, as the CSS rules are defined inside of backticks.

The styled components that we defined work exactly like regular *button* and *input* elements, and they can be used the same way:

```
const Login = (props) => {
  // ...
  return (
    <div>
      <h2>login</h2>
      <form onSubmit={onSubmit}>
        <div>
          username:
          <Input />
        </div>
        <div>
          password:
          <Input type='password' />
        </div>
        <Button type="submit" primary=''>login</Button>
      </form>
    </div>
  )
}
```

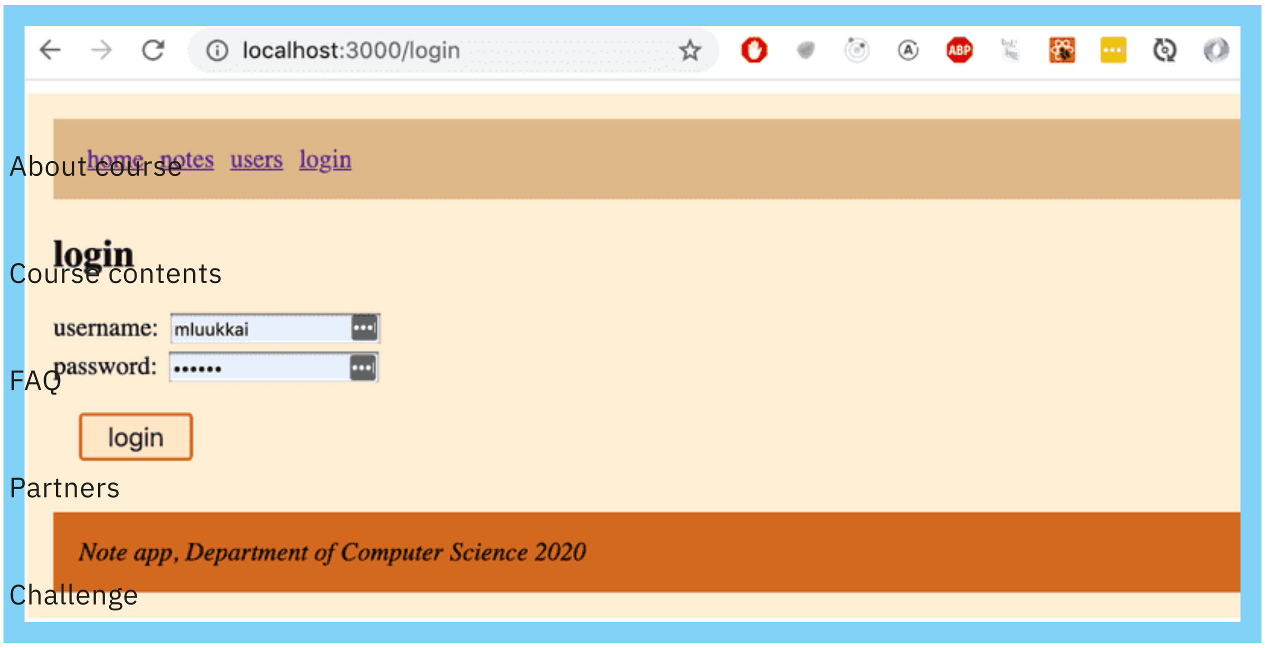
Let's create a few more components for styling that application, that are styled versions of *div* elements:

```
const Page = styled.div`
  padding: 1em;
  background: papayawhip;
`

const Navigation = styled.div`
  background: BurlyWood;
  padding: 1em;
`

const Footer = styled.div`
  background: Chocolate;
  padding: 1em;
  margin-top: 1em;
`
```

Let's use the components in our application:

```
const App = () => {
  // ...

  return (
    <Page>
      <Navigation>
        <Link style={padding} to="/">home</Link>
        <Link style={padding} to="/notes">notes</Link>
        <Link style={padding} to="/users">users</Link>
        {user
          ? <em>{user} logged in</em>
          : <Link style={padding} to="/login">login</Link>
        }
      </Navigation>

      <Switch>
        <Route path="/notes/:id">
          <Note note={note} />
        </Route>
        <Route path="/notes">
          <Notes notes={notes} />
        </Route>
        <Route path="/users">
          {user ? <Users /> : <Redirect to="/login" />}
        </Route>
        <Route path="/login">
          <Login onLogin={login} />
        </Route>
        <Route path="/">
          <Home />
        </Route>
      </Switch>
      <Footer>
        <em>Note app, Department of Computer Science 2021</em>
      </Footer>
    </Page>
  )
}
```

The appearance of the resulting application is shown below:



About course
Course contents
FAQ
Partners
Challenge

Styled components have seen a consistent growth in popularity in recent times, and quite a lot of people consider it to be the best way of defining styles to React applications.

## Exercises

The exercises related to the topics presented here, can be found at the end of this course material section in the exercise set for extending the blog list application.

Propose changes to material

Part 7b
Previous part

Part 7d
Next part

HOUSTON