VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

FACULTY OF FUNDAMENTAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGIES

Ashar Mushtaq

Homework 3 of Internet Technologies

Created a Basic web app using MERN Stack

Submitted to: Urtė Radvilaitė

Vilnius, 2023

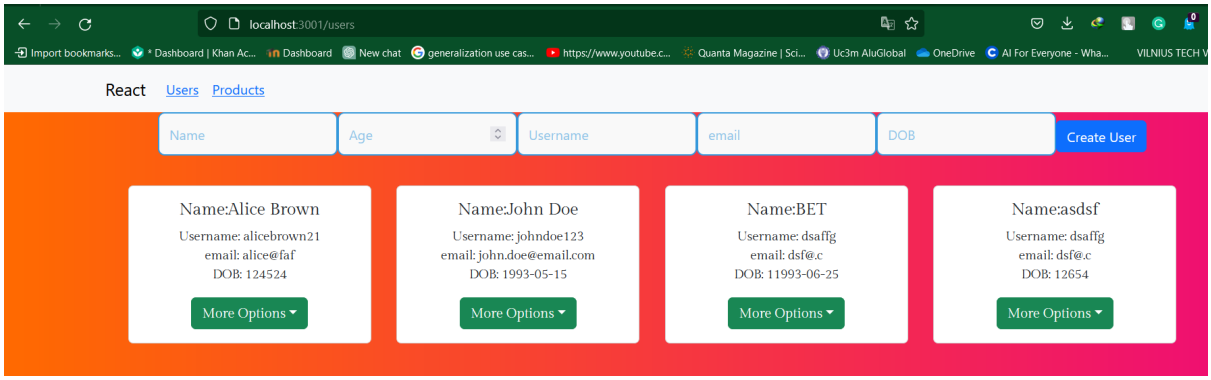
Project Overview:

This assignment encompasses practical exercises aimed at acquainting oneself with MongoDB, Node.js, Express.js, and React, collectively forming the MERN stack. The prescribed tasks were executed diligently, and each task is meticulously documented within the project. The comprehensive documentation includes explicit examples accompanied by code excerpts and the corresponding outcomes of each individual task.

Moreover, the entire project is also available on GitHub, providing a transparent and accessible platform for further review and evaluation.
https://github.com/usherardy/react.js_work/tree/master

Project Structure:

Task 1: Perform Lab 7, Create a mongodb account and collection and provide the users onto the page.



The picture shows the final outcome of the project.

```
const UserModel=require('./models/users.js')
mongoose.connect("mongodb+srv://admin:root@cluster0.fdglyqc.mongodb.net/login?retryWrites=true&w=majority")

app.listen(3000, () => {
    console.log("server running in port 3000")
});
app.get("/getUsers", (req, res)=>{
    UserModel.find().then(function(response){
    res.json(response);
    }).catch(function(err){
    res.json(err);
    })
});

app.post("/createUser", async (req, res)=>{
    const user = req.body;
    const newUser = new UserModel (user);
    await newUser.save();
    res.json(user)
 });
```

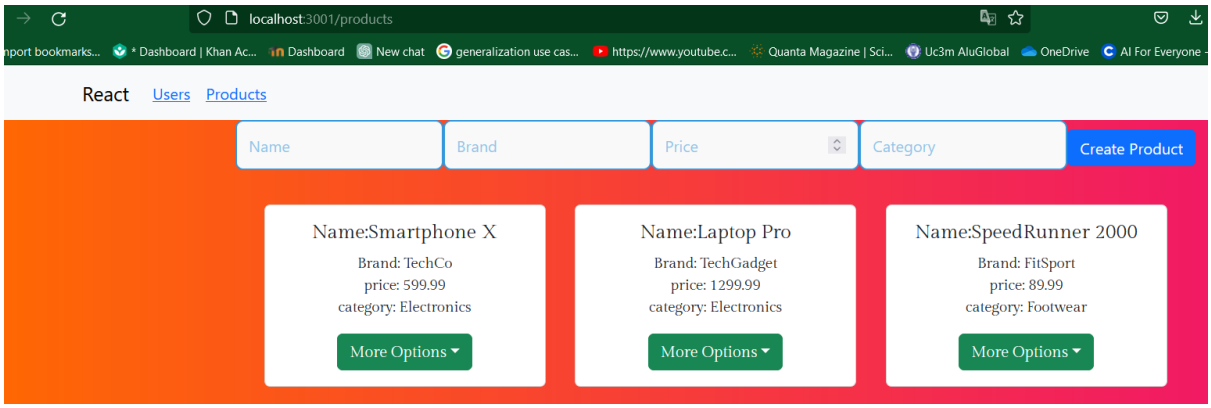
Used PostMan API to give HTTP request to the server (GET/POST)

Task 2:

You can also see the css applied to box fields. Below is the code for App.css

```css
@import url('https://fonts.googleapis.com/css2?family=Gilda+Display&family=Sofia+Sans+Conde

.App {
  text-align: center;
}


.userDisplay {
  font-family: 'Gilda Display', sans-serif;
  font-size: 15px;
  margin-top:20px;
  display:flex;
  justify-content:center;
}

body {
  background: linear-gradient(to right, #ff6a00, #ee0979);
}

input {
  padding: 12px;
  margin-bottom: 15px;
  box-sizing: border-box;
  margin-left:15px;
  margin-top: 25px;
  font-size: 16px;
  border: 2px solid #3498db;
  border-radius: 8px;
  outline: none;
  transition: border-color 0.3s ease-in-out, box-shadow 0.3s ease-in-out;
  background-color: #f9f9f9;
  color: #333;
}

input:focus {
  border-color: #2980b9;
  box-shadow: 0 0 10px rgba(41, 128, 185, 0.5);
}


input::placeholder {
  color: #3498db;
}


input:invalid {
  background-color: #f0f0f0;
}

input:hover {
  transition: background-color 0.5s ease-in-out;
  background-color: #e0e0e0;
}
```

Task 3: Create another collection



You can see the next collection showing products on the page.

```
function CreateProduct()
{
    const [listOfProducts, setListOfProducts] = useState([]);
    const [name, setName] = useState("");
    const [brand, setBrand] = useState("");
  const [price, setPrice] = useState(0);
  const [category, setCategory] = useState("");

  useEffect(() => {
    Axios.get("http://localhost:3000/getProducts").then((response) => {
      setListOfProducts(response.data);
    });
  }, []);

  const createProduct = () => {
    Axios.post("http://localhost:3000/createProduct", {
      name: name,
      brand: brand,
      price: price,
      category: category,
    }).then((response) => {
      setListOfProducts([
        ...listOfProducts,
        {
```

Task 4: create two new fields

From the first picture you might have seen the email and DOB field added.

Here is the code:

```
<input type="text" placeholder="email " onChange={(event)=>{setEmail(event.target.value);}}/>
<input type="text" placeholder="DOB" onChange={(event)=>{setdob(event.target.value);}}/>
<Button onClick={createUser}>Create User </Button>
```

```
function CreateUser(){
    const [listOfUsers, setListOfUsers] = useState([]);
    useEffect(()=> {
      Axios.get("http://localhost:3000/getUsers").then((response)=>{
      setListOfUsers(response.data);
      });
      }, []);
    const [name, setName] = useState("")
    const [age, setAge] = useState(0)
    const [username, setUsername] = useState("")
    const [email, setEmail]= useState("")
    const [dob, setdob] =useState("")


              const createUser = () => {
                Axios.post("http://localhost:3000/createUser", {
                name: name,
                age: age,
                username: username,
                email:email,
                dob:dob,
                }).then((response)=>{
                  setListOfUsers([...listOfUsers, {
                    name,
                    age,
                    username,
                    email,
                    dob,
                    },
                    ]);
                });
```

Task 5: Use bootstrap CARD and another component

From the picture, you can see the user and product details are displayed on bootstrap card, and NavBar is used.

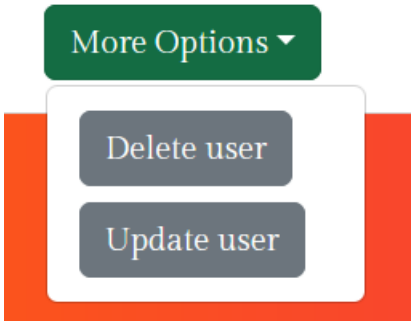
```
<Card style={{ width: '18rem', margin:"15px" }}>
      <Card.Body>
        <Card.Title>Name:{user.name}</Card.Title>
        <Card.Text>
          Username: {user.username}<br/>
          email: {user.email}<br/>
          DOB: {user.dob}<br/>
        </Card.Text>

        <Dropdown>
          <Dropdown.Toggle variant="success" id="dropdown-basic">
            More Options
          </Dropdown.Toggle>

          <Dropdown.Menu>
            <Dropdown.Item onClick={() => deleteUser(user.name)}> <Button variant="secondary" >Delete user</Button><
            <Dropdown.Item><Button variant="secondary" onClick={() => updateUser(user.name)}>Update user</Button></D

          </Dropdown.Menu>
        </Dropdown>

      </Card.Body>
```

Different Component is created from Navbar and used as Routes Links

```
function NavBar() {
  return (
    <Navbar expand="lg" className="bg-body-tertiary">
      <Container>
        <Navbar.Brand href="#home">React</Navbar.Brand>
        <Navbar.Toggle aria-controls="basic-navbar-nav" />
        <Navbar.Collapse id="basic-navbar-nav">
          <Nav className="me-auto">
          <Nav.Link>
              <Link to="/users">Users</Link>
            </Nav.Link>
            <Nav.Link>
            <Link to="/products">Products</Link>
            </Nav.Link>
          </Nav>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
}
```

Task 6: Use two more react bootstrap components

I have used bootstrap button and dropdown as two more components:



Delete and Update are in dropdown with buttons delete and update user.

```
<Dropdown>
  <Dropdown.Toggle variant="success" id="dropdown-basic">
    More Options
  </Dropdown.Toggle>

  <Dropdown.Menu>
    <Dropdown.Item onClick={() => deleteUser(user.name)}> <Button variant="secondary" >Delete user</Button></Dropdow
    <Dropdown.Item><Button variant="secondary" onClick={() => updateUser(user.name)}>Update user</Button></Dropdown.
```

Task 7: Use react-router-dom library

```
  <div className="App">
    <Router>
      <NavBar/>
       <Routes>

          <Route path="/users" element={ <CreateUser/>}/>

          <Route path ="/products" element={ <CreateProduct/>}/>

        </Routes>
      </Router>
  </div>
</>
```

```
<Nav.Link>
    <Link to="/users">Users</Link>
  </Nav.Link>
  <Nav.Link>
  <Link to="/products">Products</Link>
  </Nav.Link>
```

Routes have been used in the work with NavLinks referring to the specific components.
If we click on the users or products on the navbar, it lists the data separately.

Task 7: Use PUT and DELETE with dedicated buttons on users collection

```
app.delete("/deleteUser/:name", async (req, res) => {
    const userName = req.params.name;

    try {
        const result = await UserModel.findOneAndDelete({ name: userName });
        res.json(result);
    } catch (error) {
        res.json({ error: "User not found or couldn't be deleted." });
    }
});

app.put("/updateUser/:name", async (req, res) => {
    const userName = req.params.name;
    const updatedUser = req.body;

    try {
        const result = await UserModel.findOneAndUpdate({ name: userName }, updatedUser, { new: true });
        res.json(result);
    } catch (error) {
        res.json({ error: "User not found or couldn't be updated." });
    }
});
```

```
const deleteUser = async (name) => {
  try {
    const response = await Axios.delete(`http://localhost:3000/deleteUser/${name}`);
    const deletedUser = response.data;
    setListOfUsers(listOfUsers.filter(user => user.name !== name));
  } catch (error) {
    console.error(`Error deleting user ${name}:`, error.message);
  }
};
const updateUser = async (name) => {
  try {
    const existingUser = listOfUsers.find(user => user.name === name);
    const updatedUser = {
      name: name,
      age: age,
      username: username,
      email:email,
      dob: dob,
    };
    const response = await Axios.put(`http://localhost:3000/updateUser/${name}`, updatedUser);
    const updatedUserData = response.data;
    setListOfUsers(listOfUsers.map(user => (user.name === name ? updatedUserData : user)));
  } catch (error) {
    console.error(`Error updating user ${name}:`, error.message);
  }
};
```

```
<Dropdown.Item onClick={() => deleteUser(user.name)}> <Button variant="secondary" >Delete user</Button></
<Dropdown.Item><Button variant="secondary" onClick={() => updateUser(user.name)}>Update user</Button></Dr
```

Task 8: Use PUT and DELETE with dedicated buttons on users collection

```
const deleteProduct = async (name) => {
  try {
    const response = await Axios.delete(`http://localhost:3000/deleteProduct/${name}`);
    const deletedProduct = response.data;
    setListOfProducts(listOfProducts.filter(product => product.name !== name));
  } catch (error) {
    console.error(`Error deleting product ${name}:`, error.message);
  }
};

const updateProduct = async (name) => {
  try {
    const existingProduct = listOfProducts.find(product => product.name === name);
    const updatedProduct = {
      name: name,
      brand: brand,
      price: price,
      category: category,
    };
    const response = await Axios.put(`http://localhost:3000/updateProduct/${name}`, updatedProduct);
    const updatedProductData = response.data;
    setListOfProducts(listOfProducts.map(product => (product.name === name ? updatedProductData : product)));
  } catch (error) {
    console.error(`Error updating product ${name}:`, error.message);
  }
};
```

```
> <Button variant="secondary"onClick={() => deleteProduct(product.name)}  >Delete user</Button>
><Button variant="secondary" onClick={() => updateProduct(product.name)}>Update user</Button></D
```

```
app.delete("/deleteProduct/:name", async (req, res) => {
  const productName = req.params.name;

  try {
    const result = await ProductModel.findOneAndDelete({ name: productName });
    res.json(result);
  } catch (error) {
    res.json({ error: "Product not found or couldn't be deleted." });
  }
});

app.put("/updateProduct/:name", async (req, res) => {
  const productName = req.params.name;
  const updatedProduct = req.body;

  try {
    const result = await ProductModel.findOneAndUpdate(
      { name: productName },
      updatedProduct,
      { new: true }
    );
    res.json(result);
  } catch (error) {
    res.json({ error: "Product not found or couldn't be updated." });
  }
});
```