Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Инженерно-технологическая академия
Институт компьютерных технологий и информационной безопасности

Отчет по практической работе № 3
на тему «**Авторизация в Web-приложениях Node.js**»
по дисциплине «Технологии разработки серверной части веб-приложений»

Выполнила
студентка группы КТбо3-4 _____ Н. И. Селевцова

Принял
доцент кафедры МОП ЭВМ _____ А. Н. Шкурко

Таганрог 2023

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ.....3

ХОД РАБОТЫ.....4

ЗАКЛЮЧЕНИЕ.....10

ЛИСТИНГ ПРОГРАММЫ.....11

1.1 index.js.....11

1.2 login.liquid.....13

1.3 home.liquid.....13

1.4 list.liquid.....14

1.5 error.liquid.....14

1.6 users.json.....15

1.7 style.css.....15

ЦЕЛЬ РАБОТЫ

Данная практическая работа направлена на изучение и практическое использование методов авторизации в web-приложениях Node.js.

Задание

В рамках выполнения практической работы требуется выполнить следующие задачи и продемонстрировать полученные навыки:

– Создать новое node.js приложение;

– Подключить к приложению библиотеки express.js и liquid;

– Разработать HTML-шаблоны для страниц приложения;

– Реализовать функциональность авторизации и аутентификации пользователя;

– Реализовать логику обработки на сервере согласно варианту задания.

Вариант задачи: 1

Необходимо реализовать простое приложение, содержащее форму для ввода логина и пароля, а также следующие страницы:

- страницу информации о текущем пользователе;

- страницу со списком пользователей;

- страницу, указывающую на отсутствие доступа к ресурсу.

Страница информации о пользователе должна отображать данные текущего авторизованного пользователя и должна быть доступна только после ввода пароля. Также при авторизации пользователя должна устанавливаться его роль (user или admin). Страница со списком пользователей должна быть доступна только пользователям с ролью admin. В случае отсутствия доступа к странице, должна отображаться соответствующая страница. Страница информации о пользователе должна содержать кнопку logout, которая возвращает пользователя на страницу ввода пароля. Пароли пользователей и их роли должны храниться в файле.

Необходимо использовать механизм сессий express.js.

ХОД РАБОТЫ

Проект разрабатываемого веб-приложения имеет следующую файловую структуру – рисунок 1. В приложение были подключены библиотеки express.js и liquid, установлено соединение с локальным сервером.

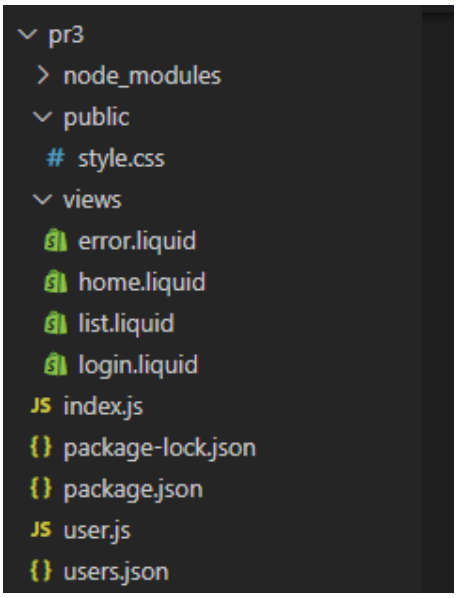

Рисунок 1 – Файловая структура приложения

```
const express = require('express');
const bodyParser = require('body-parser');
var cookieParser = require('cookie-parser');
var session = require('express-session');
var { Liquid } = require('liquidjs');
const users = require("./users.json");

var engine = new Liquid();


const app = express();
app.use(express.static('public'));

const port = 3000;

app.engine('liquid', engine.express());
app.set('views', './views');
app.set('view engine', 'liquid');

app.use(cookieParser());

app.use(bodyParser.urlencoded({
  extended: true
}));

```

Рисунок 2 – Подключенные библиотеки и файлы

На языке HTML были созданы шаблоны страниц приложения. Был использован шаблонизатор Liquid вместе с промежуточным обработчиком bodyParser. Во время выполнения механизм шаблонов заменяет переменные в файле шаблона фактическими значениями и преобразует шаблон в HTML-файл, отправляемый клиенту.

Были реализованы функциональность авторизации и аутентификации пользователя (рисунки 3-4) и логика обработки на сервере согласно варианту задания.

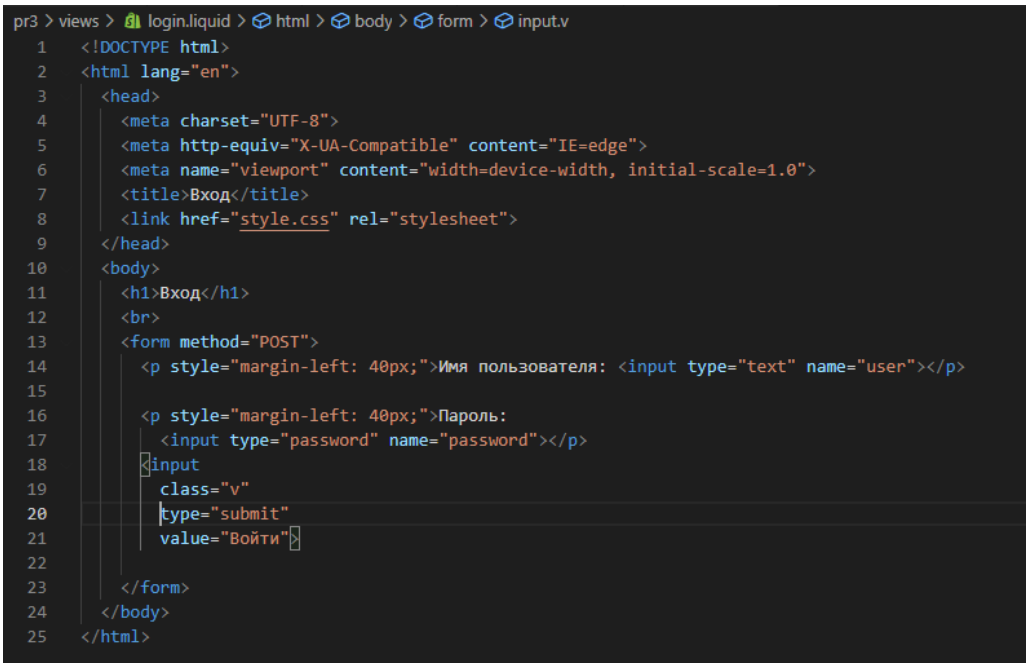

Рисунок 3 – Форма авторизации

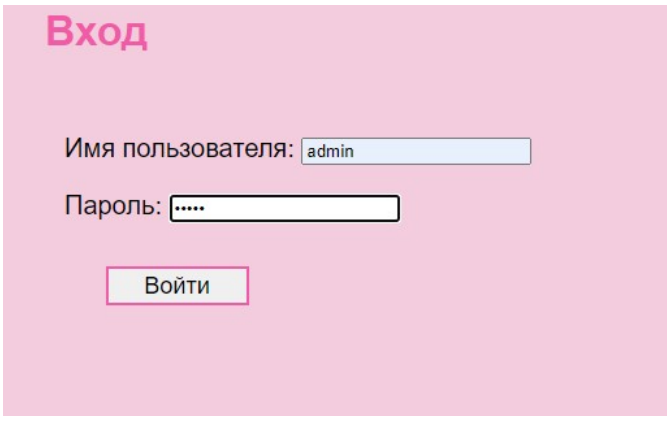

Рисунок 4 – Внешний вид страницы авторизации

Также был создан файл JSON с данными пользователей для их аутентификации на сервере (рисунок 5). Для каждого пользователя устанавливается его роль (user или admin). Информация о текущем

авторизированном пользователе отображается только после ввода пароля на странице информации о пользователе (рисунки 6-8).

pr3 > {} users.json > {} admin > about

```json
{
  "user1": {
    "name": "user1",
    "pass": "1",
    "role": "user",
    "avatar": "https://cdn.fastcup.net/avatars/users/17526_7cl8epj53.jpg",
    "hobby": "коньки, шахматы, бокс, рисование",
    "about": "ничего не скажу)))"
  },
  "admin": {
    "name": "admin",
    "pass": "1",
    "role": "admin",
    "avatar": "http://esx.bigo.sg/eu_live/2u4/27sGxC.jpg",
    "hobby": "администрирование",
    "about": "Я всего лишь страница для проверки работы списка пользователей"
  },
  "user2": {
    "name": "user2",
  "pass": "1",
    "role": "user",
    "avatar": "https://davidmichie.com/wp-content/uploads/2015/03/cute-pappillon-300x300.jpg",
    "hobby": "наука",
    "about": "Человеческая наука, по существу, рациональная в своих основах и по своим методам, може
  }
}
```

Рисунок 5 – Файл с информацией о пользователях

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Моя страница</title>
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <h1>Моя страница
    </h1>
    <img
      src="{{avatar}}"
      alt="Avatar"
      class="avatar">
    <div id="us">
      <p>
        <b>Меня зовут:</b>
        {{ userId }}
        </p>
          <p><b>Хобби:</b>
          {{ hobby }}</p>
          <p align="justify"><b>О себе:</b>
          {{ about }}
        </p>
      </div>
      <br><br>
      <form action="/logout">
        <button>Выход</button>
      </form>
    </body>
  </body>
</html>
```

Рисунок 6 – Шаблон страницы с информацией пользователя

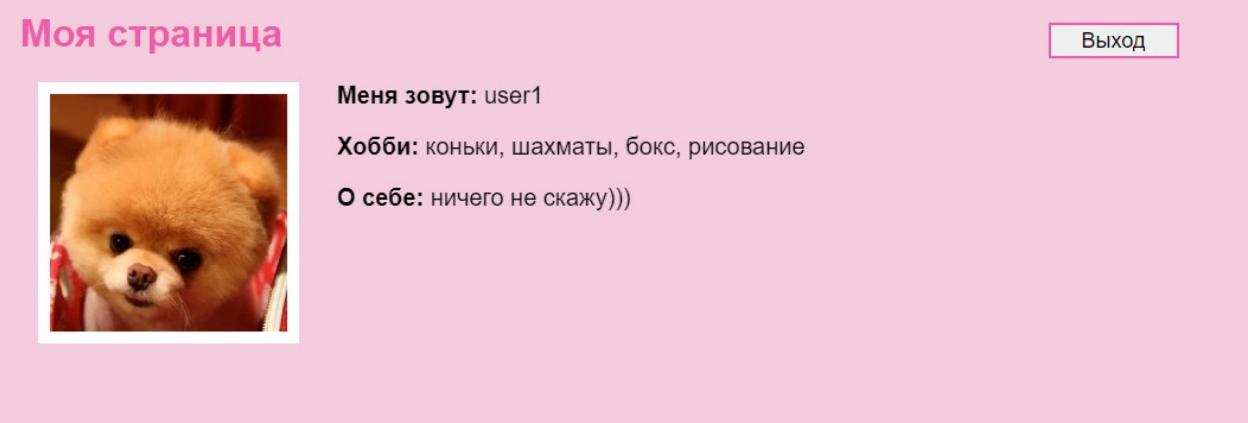

Рисунок 7 – Пример страницы пользователя

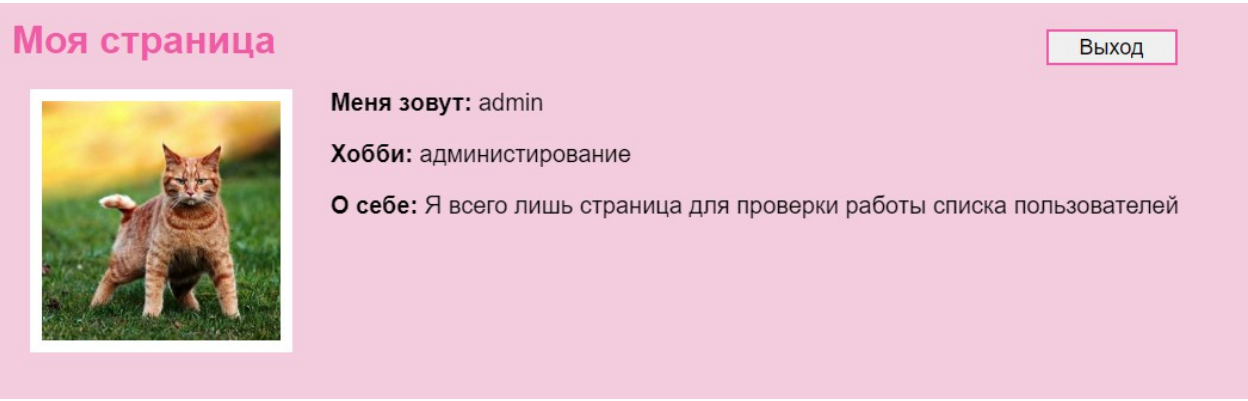

Рисунок 8 – Пример страницы пользователя [2]

```
function auth(req, res, next) {
    if (req.user) {
        next();
    } else {
        res.redirect('/login');
    }
}

app.use((req, res, next) => {
    if (req.session.user) {
        req.user = {
            id: req.session.user
        }
    }
    next();
});

app.get('/login', (req, res) => res.render('login'));
```

Рисунок 9 – Код для авторизации и аутентификации

```
  if (users[req.body.user] && users[req.body.user].pass === req.body.password) {
    req.session.user = users[req.body.user].name;
    res.cookie("role", users[req.body.user].role);

      res.redirect("/");
      return;
    } else {
        res.redirect('/login');
    }
  });

  app.get('/logout', (req, res) => {
    req.session.user = '';
    req.cookies.role = '';
    res.redirect('/login');
});

app.get('/', auth, (req, res) => {
  res.render('home', {
      userId: req.user.id,
      avatar: users[req.session.user].avatar,
      hobby: users[req.session.user].hobby,
      about: users[req.session.user].about
  })
})
```

Рисунок 10 - Код для авторизации и аутентификации [2]

Также была создана страница, отображающая список пользователей. Она доступна только пользователям с ролью admin. В случае отсутствия доступа к странице, отображается страница, указывающая на отсутствие доступа к ресурсу.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Список пользователей</title>
    <link href="style.css" rel="stylesheet">
  </head>
  <body>

    <h2>Список пользователей</h2>
    <ul>
      <li>{{ name1 }}</li>
      <br>
      <li>{{ name2 }}</li>
      <br>
      <li>{{ name3 }}</li>
    </ul>
  </body>
  <script></script>
</html>
```

Рисунок 11 – Шаблон списка пользователей

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ошибка</title>
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <h2>Отказано в доступе</h2>

    <form action='/'>
      <button class="kl">Вернуться на страницу</button>
    </form>
  </body>
</html>
```

Рисунок 12 – Шаблон страницы с ошибкой

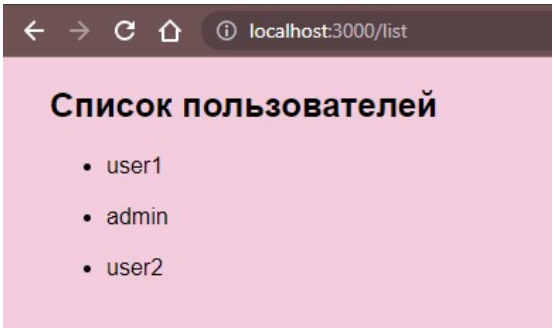

Рисунок 13 – Внешний вид страницы по пути /list для admin

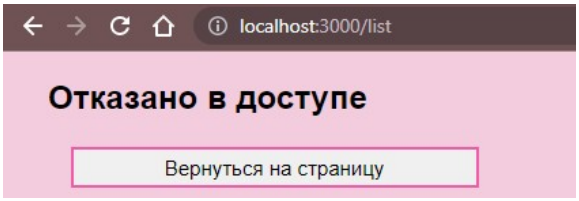

Рисунок 14 – Внешний вид страницы по пути /list для user

```
app.get('/list', auth, (req, res) => {
 if (req.cookies.role === "admin") {
   let name1=users['user1']['name'];
   let name2=users['admin']['name'];
   let name3=users['user2']['name'];
   res.render('list', {
     name1,
     name2,
     name3,
   });
 } else {
   res.render('error');
 }

});
```

Рисунок 15 – Реализация проверки роли пользователя для доступа к странице

ЗАКЛЮЧЕНИЕ

В ходе практической работы были изучены и освоен метод авторизации в web-приложениях Node.js. Было создано новое node.js приложение с подключенными к приложению библиотеками express.js и liquid, разработаны HTML-шаблоны для страниц приложения, реализована авторизация и аутентификация пользователя и логика обработки на сервере согласно варианту задания.

ЛИСТИНГ ПРОГРАММЫ

1.1 index.js

```
const express = require('express');
const bodyParser = require('body-parser');
var cookieParser = require('cookie-parser');
var session = require('express-session');
var { Liquid } = require('liquidjs');
const users = require("./users.json");

var engine = new Liquid();

const app = express();
app.use(express.static('public'));

const port = 3000;

app.engine('liquid', engine.express());
app.set('views', './views');
app.set('view engine', 'liquid');

app.use(cookieParser());
app.use(bodyParser.urlencoded({
  extended: true
}));

app.set('trust proxy', 1);
app.use(session({
    secret: 'catcatcat',
    resave: false,
    saveUninitialized: true,
  }))

  function auth(req, res, next) {
    if (req.user) {
        next();
    } else {
        res.redirect('/login');
    }
}

app.use((req, res, next) => {
  if (req.session.user) {
      req.user = {
          id: req.session.user
      }
  }
  next();
```

```
});

app.get('/login', (req, res) => res.render('login'));

app.post("/login", (req, res) => {
  if (users[req.body.user] && users[req.body.user].pass ===
req.body.password) {
    req.session.user = users[req.body.user].name;
    res.cookie("role", users[req.body.user].role);
      res.redirect("/");
      return;
    } else {
        res.redirect('/login');
    }
  });

  app.get('/logout', (req, res) => {
    req.session.user = '';
    req.cookies.role = '';
    res.redirect('/login');
 });

 app.get('/', auth, (req, res) => {
  res.render('home', {
      userId: req.user.id,
      avatar: users[req.session.user].avatar,
      hobby: users[req.session.user].hobby,
      about: users[req.session.user].about
  })
})

 app.get('/list', auth, (req, res) => {
  if (req.cookies.role === "admin") {
    let name1=users['user1']['name'];
    let name2=users['admin']['name'];
    let name3=users['user2']['name'];
    res.render('list', {
      name1,
      name2,
      name3,
    });
  } else {
    res.render('error');
  }
 });

 app.use((err, req, res, next) => {
    console.error(err.stack);
    res.status(500).send('Something broke!');
 })
```

```
app.listen(port, () => console.log('Server is running on port 3000'));
```

1.2 login.liquid

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Вход</title>
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <h1>Вход</h1>
    <br>
    <form method="POST">
      <p style="margin-left: 40px;">Имя пользователя: <input type="text"
name="user"></p>

      <p style="margin-left: 40px;">Пароль:
        <input type="password" name="password"></p>
      <input
        class="v"
        type="submit"
        value="Войти">

    </form>
  </body>
</html>
```

1.3 home.liquid

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Моя страница</title>
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <h1>Моя страница
    </h1>
    <img
      src="{{avatar}}"
      alt="Avatar"
      class="avatar">
    <div id="us">
      <p>
```

```
        <b>Меня зовут:</b>
        {{ userId }}
        </p>
          <p><b>Хобби:</b>
          {{ hobby }}</p>
          <p align="justify"><b>О себе:</b>
          {{ about }}
        </p>
      </div>
      <br><br>
      <form action="/logout">
        <button>Выход</button>
      </form>
    </body>
  </body>
</html>
```

1.4 list.liquid

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Список пользователей</title>
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <h2>Список пользователей</h2>
    <ul>
      <li>{{ name1 }}</li>
      <br>
      <li>{{ name2 }}</li>
      <br>
      <li>{{ name3 }}</li>
    </ul>
  </body>
  <script></script>
</html>
```

1.5 error.liquid

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ошибка</title>
    <link href="style.css" rel="stylesheet">
  </head>
```

```
  <body>
    <h2>Отказано в доступе</h2>
    <form action='/'>
      <button class="kl">Вернуться на страницу</button>
    </form>
  </body>
</html>
```

1.6 users.json

```
{
    "user1": {
      "name": "user1",
      "pass": "1",
      "role": "user",
      "avatar":
"https://cdn.fastcup.net/avatars/users/17526_7cl8epj53.jpg",
      "hobby": "коньки, шахматы, бокс, рисование",
      "about": "ничего не скажу)))"
    },
    "admin": {
      "name": "admin",
      "pass": "1",
      "role": "admin",
      "avatar": "http://esx.bigo.sg/eu_live/2u4/27sGxC.jpg",
      "hobby": "администрирование",
      "about": "Я всего лишь страница для проверки работы списка
пользователей"
    },
    "user2": {
      "name": "user2",
    "pass": "1",
      "role": "user",
      "avatar": "https://davidmichie.com/wp-content/uploads/2015/03/cute-
pappillon-300x300.jpg",
      "hobby": "наука",
      "about": "Человеческая наука, по существу, рациональная в своих
основах и по своим методам, может осуществлять свои наиболее значительные
завоевания лишь путем опасных внезапных скачков ума, когда проявляются
способности, освобожденные от тяжелых оков старого рассуждения, которые
называют воображением, интуицией, остроумием. Лучше сказать, ученый
проводит рациональный анализ и перебирает звено за звеном цепь своих
дедукций: эта цепь его сковывает до определенного момента; затем он от нее
мгновенно освобождается, и вновь обретенная свобода его воображения
позволяет ему увидеть новые горизонты. (с) Луи де Бройль"
    }
  }
```

1.7 style.css

```
body{
    background-color:#f3ccde;
```

```
    font-family: arial;

}
h1{
    color:#f15ba8;
    margin-left: 25px;
}
h2{
    margin-left: 25px;
}
li{
    margin-left: 25px;
}
p{
    font-size: 20px;
    margin-right: 120px;
}
.v{
    left:80px;
    top:220px;
    position: fixed;
    width: 110px;
    height: 30px;
    position: fixed;
    color:#000000;
    border: 2px solid #f15ba8;
    padding-left: 5px;
    font-size: 18px;
}
.v:hover{
    cursor: pointer;
    background-color:#a59ca1;
}
.kl{
    top: 70px;
    left: 50px;
    position: fixed;
    width: 300px;
    font-size: 15px;
}
.l{
    top: 70px;
    left: 1000px;
    position: fixed;
    width: 300px;
    font-size: 15px;
}

.avatar{
    margin-left: 40px;
```

```
    width: 200px;
    height: 200px;
    border: 10px solid #ffffff;
}
#us{
    top: 60px;
    left:300px;
    position: fixed;
}

button{
    top: 30px;
    left: 900px;
    width: 110px;
    height: 30px;
    position: fixed;
    color:#000000;
    border: 2px solid #f15ba8;
    padding-left: 5px;
    font-size: 18px;

}
button:hover{
    cursor: pointer;
    background-color:#a59ca1;
}
```