Final ProjectDocument

Mustafa Nafi Uğur
20212022049

Description:

Conseptof the Project is a Design Company's introduction and ShowCase.

Site has atwo part one part is Main Page and other is Contact InformationPage.

Both parts of links are contains 3 chapter Header ,Body, Footer.

Elements ;

Home:ThisPart contains Drop Downelements don't goesany link but they have hash tag.

About Us:This partgoesaMain Page and Itcontainsintroductinabout company.

Services: When Clicked it goes a form in MainPage and This button contains DropDown.

Dropdown elements connected with database in this part firstly you site opens with guest so you can not see the products when you login or signup services part will open and you can see the products

Contact: When you click you'll direct to Contact Page.

Sign in (User Information):Connected with database and database have control code to check users name if already user have same name it gives error.

Body:

There are three parts in Main Page;

Home Part: This part contains autophoto slayt and animationed heading.(Writenwith .jss).

About UsPart: This part gives information about company and there is a continue button to get more introduction this button written with css and js action elemnts.

Join Us Part:This part have form part when you fil lall the blanks Visitor part where inthe top-right of site itchanges to your name.Connected with database takes the user info end sends to database.

Login Part : If you already create an account you can reach there to only clicking to Clik me part where in the under the form part. This part also connected with database and if name and pass Word true you can reach the products.

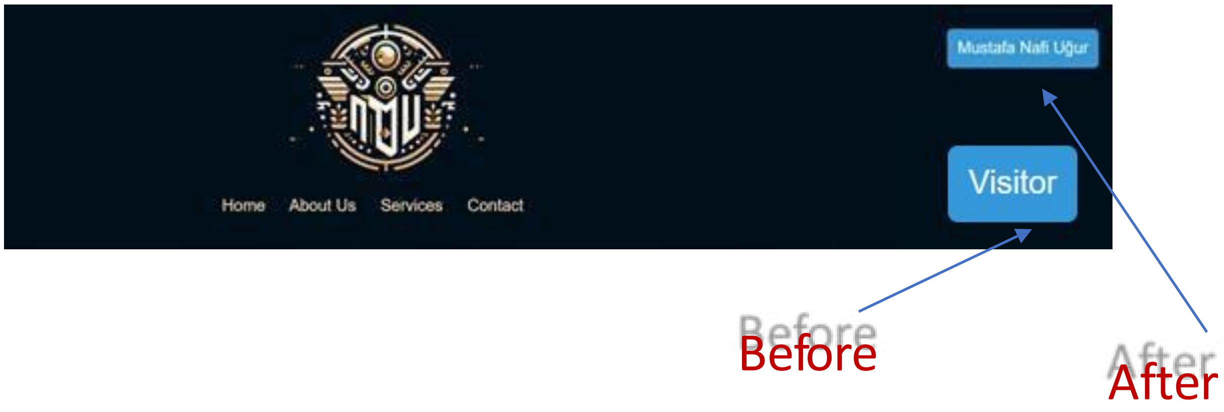
Footer:

Shows Licences; designed with css.

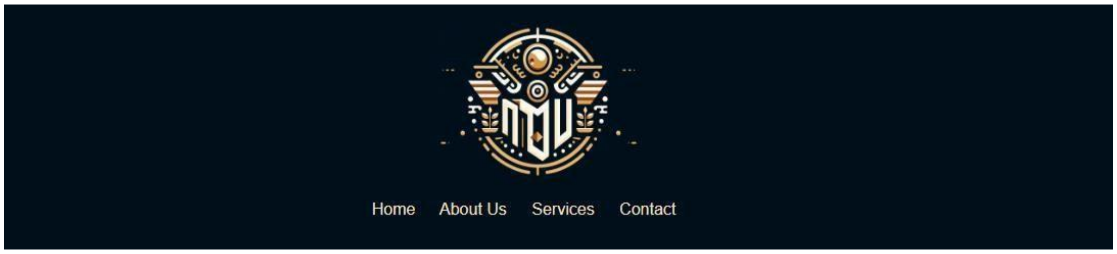
Contact Page:

This page you will direct to when you click to Contact part where in header part.

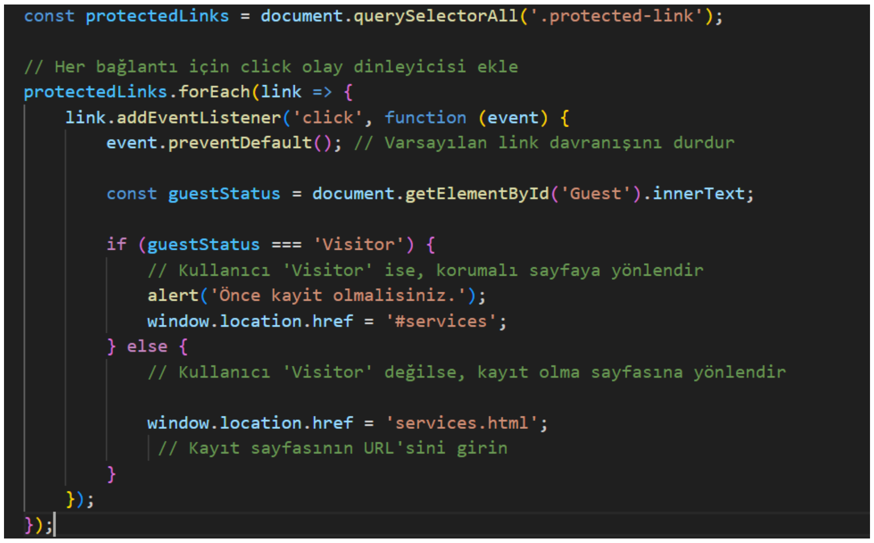
There isa table class.Itcontainscontact details(fax,mail,phone,adress and maps).



Header Part



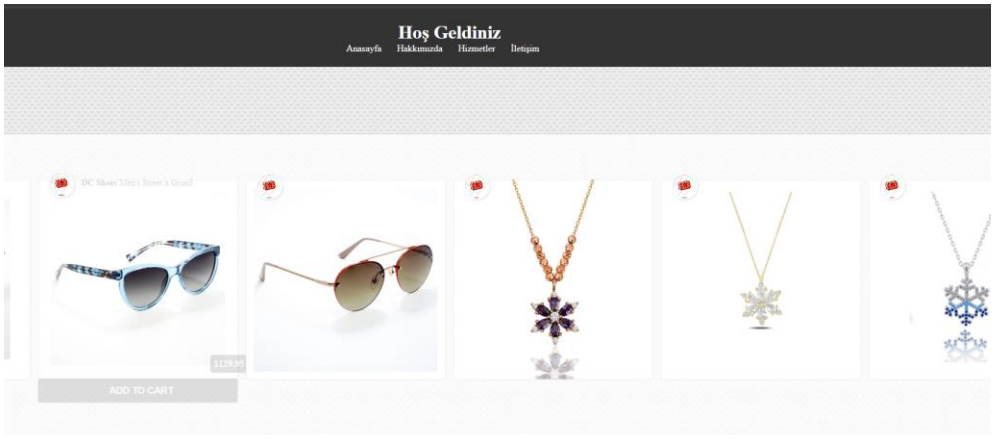
To reach Services part there is a js code the code takes “Visitor” tag if Visitor did’nt register or log in it sends to login form after then when you click you will send to Product’s page .



```
const protectedLinks = document.querySelectorAll('.protected-link');

// Her bağlantı için click olay dinleyicisi ekle
protectedLinks.forEach(link => {
    link.addEventListener('click', function (event) {
        event.preventDefault(); // Varsayılan link davranışını durdur

        const guestStatus = document.getElementById('Guest').innerText;

        if (guestStatus === 'Visitor') {
            // Kullanıcı 'Visitor' ise, korumalı sayfaya yönlendir
            alert('Önce kayit olmalisiniz.');
            window.location.href = '#services';
        } else {
            // Kullanıcı 'Visitor' değilse, kayıt olma sayfasına yönlendir

            window.location.href = 'services.html';
             // Kayıt sayfasının URL'sini girin
        }
    });
});
```

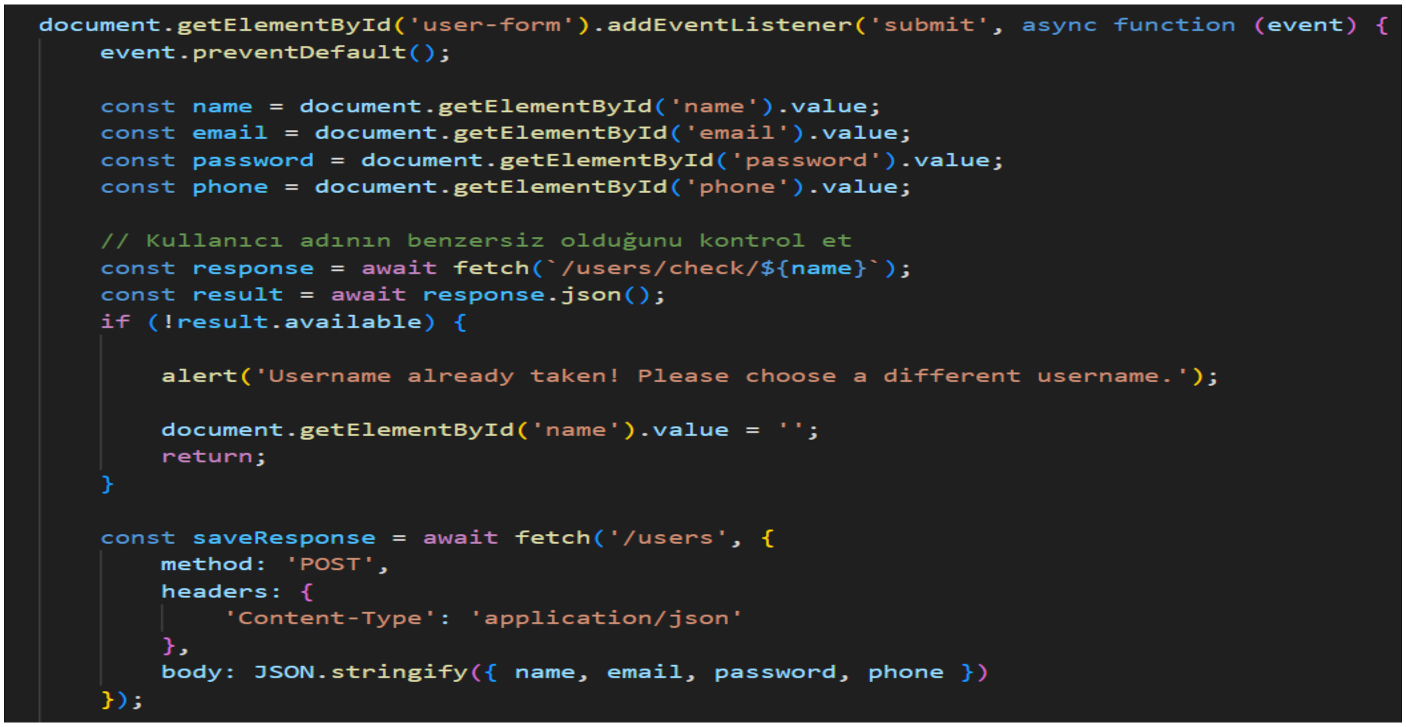
Mustafa Nafi Uğur
20212022049

Product Page



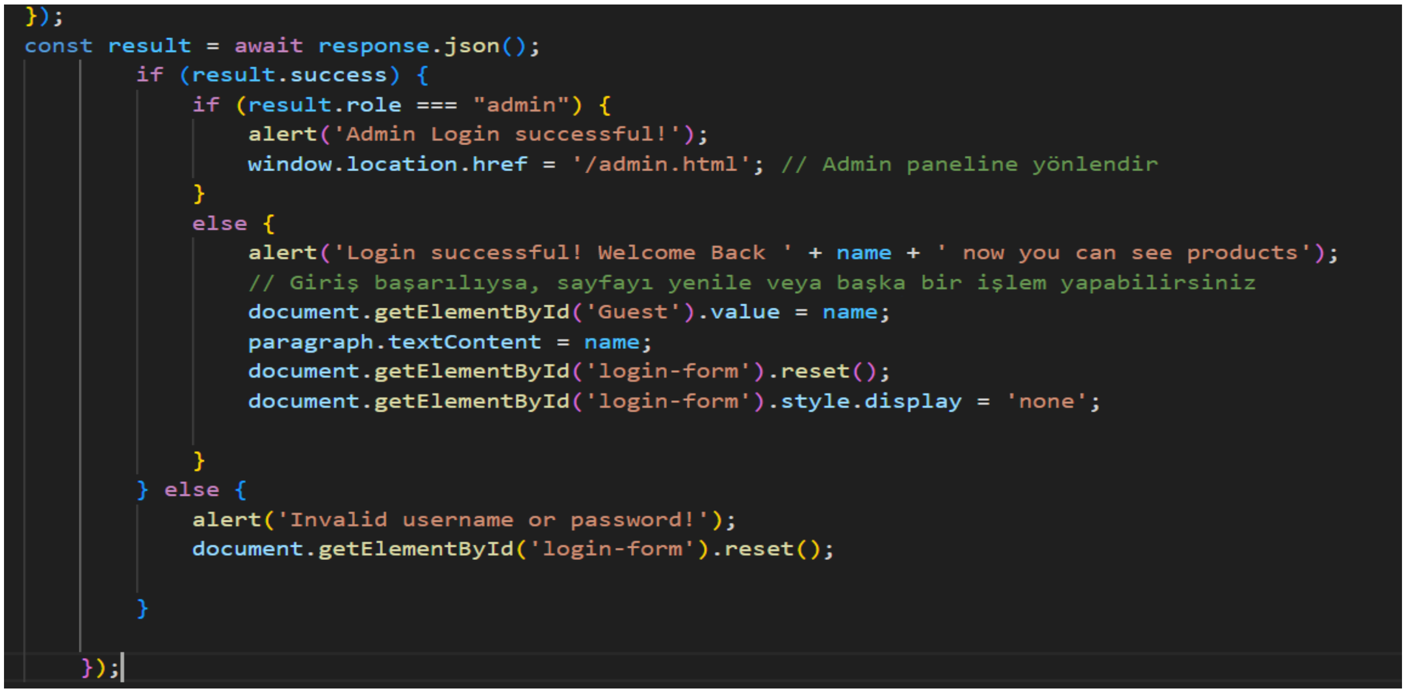
BODY PART

In body part there is a register part .When form filled correctly and submit button action happen database connection will be active and part by part items equaling with tags.After then checking name in database and trying to find if there and same name in database giving alert and empty the name box



```
document.getElementById('user-form').addEventListener('submit', async function (event) {
    event.preventDefault();

    const name = document.getElementById('name').value;
    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;
    const phone = document.getElementById('phone').value;

    // Kullanıcı adının benzersiz olduğunu kontrol et
    const response = await fetch(`/users/check/${name}`);
    const result = await response.json();
    if (!result.available) {

        alert('Username already taken! Please choose a different username.');

        document.getElementById('name').value = '';
        return;
    }

    const saveResponse = await fetch('/users', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ name, email, password, phone })
    });
```

if you have account you can click “Already have account?” and log in part will open between the register part and takes name and password .



```
});
const result = await response.json();
        if (result.success) {
            if (result.role === "admin") {
                alert('Admin Login successful!');
                window.location.href = '/admin.html'; // Admin paneline yönlendir
            }
            else {
                alert('Login successful! Welcome Back ' + name + ' now you can see products');
                // Giriş başarılıysa, sayfayı yenile veya başka bir işlem yapabilirsiniz
                document.getElementById('Guest').value = name;
                paragraph.textContent = name;
                document.getElementById('login-form').reset();
                document.getElementById('login-form').style.display = 'none';

            }
        } else {
            alert('Invalid username or password!');
            document.getElementById('login-form').reset();

        }

    });
```

Mustafa Nafi Uğur
20212022049

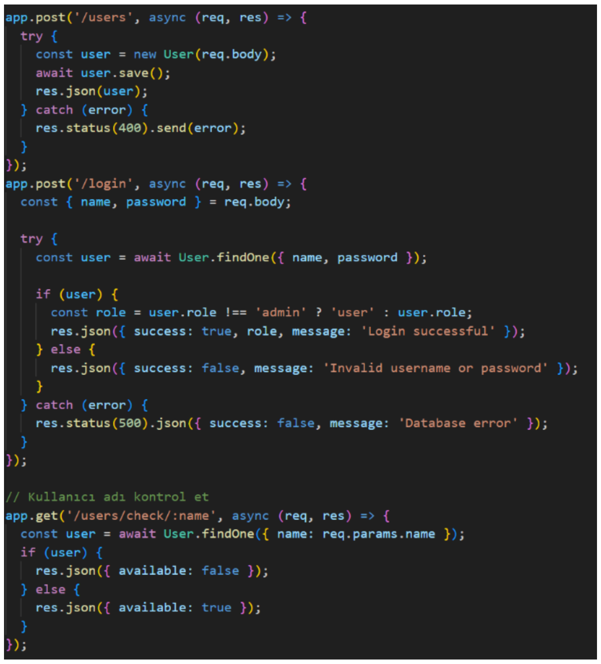
Server & Connection

This Node.js application sets up an Express server integrated with MongoDB to handle user registration, login, and standard CRUD operations on user data.

Dependencies:

- **express**: Used to construct the server and handle HTTP requests.
- **mongoose**: Provides MongoDB object modeling for Node.js applications.
- **body-parser**: Middleware for parsing request bodies.
- **path**: Utility module for managing file paths.

Setting up MongoDB:



```
app.post('/users', async (req, res) => {
  try {
    const user = new User(req.body);
    await user.save();
    res.json(user);
  } catch (error) {
    res.status(400).send(error);
  }
});
app.post('/login', async (req, res) => {
  const { name, password } = req.body;

  try {
    const user = await User.findOne({ name, password });

    if (user) {
      const role = user.role !== 'admin' ? 'user' : user.role;
      res.json({ success: true, role, message: 'Login successful' });
    } else {
      res.json({ success: false, message: 'Invalid username or password' });
    }
  } catch (error) {
    res.status(500).json({ success: false, message: 'Database error' });
  }
});

// Kullanıcı adı kontrol et
app.get('/users/check/:name', async (req, res) => {
  const user = await User.findOne({ name: req.params.name });
  if (user) {
    res.json({ available: false });
  } else {
    res.json({ available: true });
  }
});
```

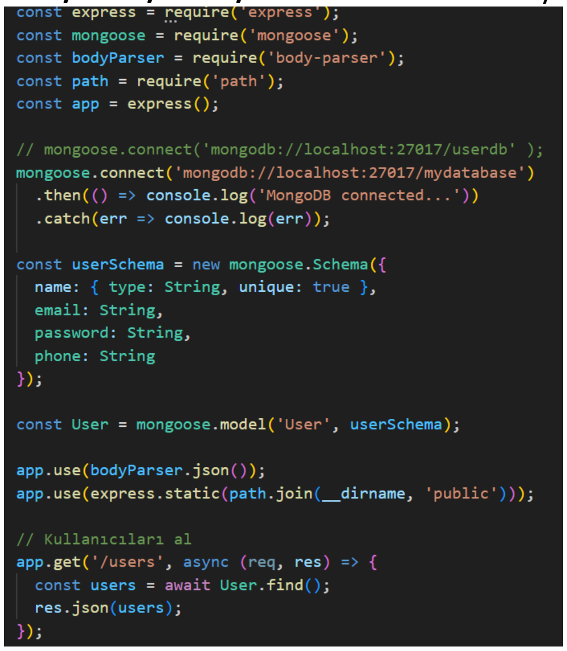
The server connects to a MongoDB database using Mongoose. Ensure that MongoDB is installed and running locally on the default port (27017). If needed, modify the database URL in the mongoose.connect() method.

Express Middleware:

body-parser: Parses JSON request bodies.

Endpoints:

1. **GET /users**: Retrieves all users from the database.
2. **POST /users**: Allows the addition of a new user to the database.
3. **POST /login**: Authenticates a user with their name and password.
4. **GET /users/check/**: Checks the availability of a username.



```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const path = require('path');
const app = express();

// mongoose.connect('mongodb://localhost:27017/userdb' );
mongoose.connect('mongodb://localhost:27017/mydatabase')
  .then(() => console.log('MongoDB connected...'))
  .catch(err => console.log(err));

const userSchema = new mongoose.Schema({
  name: { type: String, unique: true },
  email: String,
  password: String,
  phone: String
});

const User = mongoose.model('User', userSchema);

app.use(bodyParser.json());
app.use(express.static(path.join(__dirname, 'public')));

// Kullanıcıları al
app.get('/users', async (req, res) => {
  const users = await User.find();
  res.json(users);
});
```

Mustafa Nafi Uğur
20212022049

PUT /users/

: Updates user data based on ID.

DELETE /users/

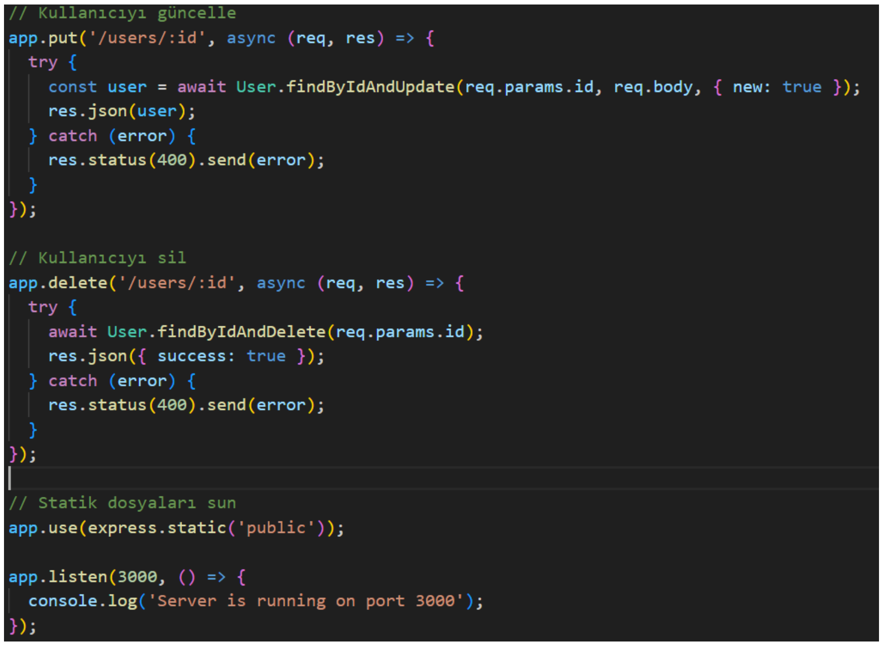
: Deletes a user based on ID.

Error Handling:

- Proper use of try-catch blocks for error handling during database operations.

Static Files:

The express.static() middleware serves static files (such as HTML, CSS, and client-side JavaScript) from the public directory.



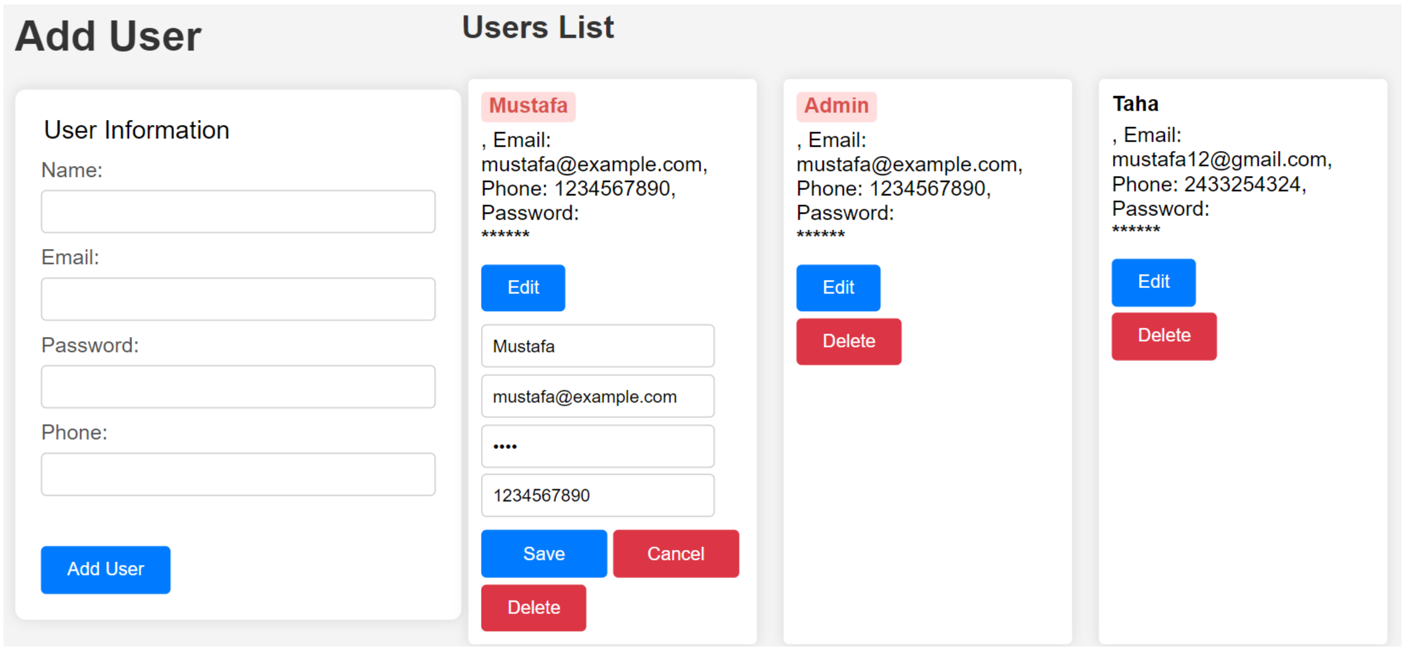
```
// Kullanıcıyı güncelle
app.put('/users/:id', async (req, res) => {
  try {
    const user = await User.findByIdAndUpdate(req.params.id, req.body, { new: true });
    res.json(user);
  } catch (error) {
    res.status(400).send(error);
  }
});

// Kullanıcıyı sil
app.delete('/users/:id', async (req, res) => {
  try {
    await User.findByIdAndDelete(req.params.id);
    res.json({ success: true });
  } catch (error) {
    res.status(400).send(error);
  }
});

// Statik dosyaları sun
app.use(express.static('public'));

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

Server Start:

- Port 3000 is opened for listening. Server start is confirmed by a console log.

Admin Panel

In admin panel there are many functions all connected with data base when new person registered shows in block and you can modify how u want only admins can Access to the panel and admins have red named tittle in server.js there is a delete user code.



Add User

User Information

Name:

Email:

Password:

Phone:

Users List

Mustafa
, Email: mustafa@example.com, Phone: 1234567890, Password: ******

Mustafa

mustafa@example.com

••••

1234567890

Admin
, Email: mustafa@example.com, Phone: 1234567890, Password: ******

Taha
, Email: mustafa12@gmail.com, Phone: 2433254324, Password: ******

Mustafa Nafi Uğur
20212022049