

|  |  |
| --- | --- |
| **Öğrenci Numarası** | **20211012252** |
| **Ad Soyad** | **Hasan Bahadır Koca** |
| **Proje Adı** | **Karekod Menü** |

**İNTERNET PROGRAMCILIĞI 2 DERSİ FİNAL PROJESİ**

**Projenizin içeriğini kısaca yazınız.**

HTML, CSS ve JS kullanılarak oluşturulan bu proje iki bölüme ayrılıyor.

İlk bölümde ziyaretçiler karekod menünün bulunduğu yerlerde bu karekodları telefonlarına okutarak herhangi bir doğrulama işlemi yapmadan HTML, CSS ve JS ile hazırlanan menü sayfasına erişebiliyor. Bu sayfa üzerinde İşletme Adı, İşletme Karşılama Metni ve Ürünler görüntülenebiliyor.

İkinci bölümde yöneticiler sisteme telefon numarası ve SMS doğrulaması ile kayıt olarak yönetim paneline erişebiliyor. Bu panel üzerinden farklı menüler oluşturabiliyor, menü içerisindeki metinleri, ürünleri ve bu ürünlere ait detayları düzenleyebilyorlar. Menüleri görüntüledikleri yerde bulunan Menü ID’si ile kendi karekodlarını oluşturup baskı alarak müşterilerinin kullanımına açabiliyorlar.

Projedeki kimlik doğrulaması Google bulut hizmetlerinden olan Firebase’in Authentication servisi ile yapılıyor. Yine SMS doğrulamasında kullanılan SMS gönderimleri de bu servis içerisinden yapılıyor.

Doğrulama yapıldıktan sonra oluşturulan menüler Authentication servisi ile bağlantılı çalışan Firebase Firestore veritabanı servisi üzerinde kayıt altına alınıyor. Hiyerarşik bir yapıya sahip bu veritabanında Authentication servisi sayesinde yöneticiler yalnızca kendilerine ait olan menüler üzerinde düzenleme yapabiliyor. Bunun yanında menülere ait veriler herkes tarafından okunabilir bir şekilde paylaşılıyor. Bu sayede ziyaretçiler menüleri incelerken veritabanındaki ilgili menü bilgilerine herhangi bir doğrulamaya ihtiyaç olmadan erişim sağlayabiliyor.

**Projenizde bulunan sayfaların görsellerini ve kodlarını ekleyiniz.**

**metin, ekran görüntüsü, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**GİRİŞ SAYFASI:**

window.onload = function () {

render();

};

let render = () => {

// [TR] reCAPTCHA için bir instance oluştur

// [EN] Create an instance for reCAPTCHA

window.recaptchaVerifier = new firebase.auth.RecaptchaVerifier(

"recaptcha-container"

);

recaptchaVerifier.render();

};

let sendnumber = () => {

// [TR] Telefon numarasını al

// [EN] Get the phone number

let number1 = document.getElementById("num").value;

console.log(number1);

// [TR] Telefon numarasının uygunluğunu kontrol et

// [EN] Check the validity of the phone number

if (!checkPhoneNumber()) {

return;

}

// [TR] Telefon numarası doğrulama kodu ile Firebase'e gönder

// [EN] Send the phone number with verification code to Firebase

firebase

.auth()

.signInWithPhoneNumber(number1, window.recaptchaVerifier)

.then((confirmationResult) => {

window.confirmationResult = confirmationResult;

coderesult = confirmationResult;

console.log(coderesult);

alert("Doğrulama kodu gönderildi.");

// [TR] Kayıt bölümünü gizle ve doğrulama bölümünü göster

// [EN] Hide the register section and show the verification section

document.getElementById("register").style.display = "none";

document.getElementById("verify").style.display = "block";

})

.catch((error) => {

alert(error.message);

});

};

let verifyUser = () => {

// [TR] Doğrulama kodunu al

// [EN] Get the verification code

let code = document.getElementById("verificationcode").value;

console.log(code);

// [TR] Doğrulama kodunu Firebase'e gönder ve kullanıcıyı doğrula

// [EN] Send the verification code to Firebase and verify the user

coderesult

.confirm(code)

.then(function (result) {

alert("Doğrulama Başarılı");

// [TR] Doğrulama başarılı olduğunda index.html sayfasına yönlendir

// [EN] Redirect to index.html when verification is successful

window.location.href = "index.html";

let user = result.user;

console.log(user);

})

.catch(function (error) {

alert(error.message);

});

};

function formatPhoneNumber() {

var phoneNumberInput = document.getElementById("num");

var phoneNumber = phoneNumberInput.value;

phoneNumber = phoneNumber.replace(/[^+0-9]/g, "");

// [TR] Telefon numarasının "+90" ile başlayıp başlamadığını kontrol et

// [EN] Check if the phone number starts with "+90"

if (!phoneNumber.startsWith("+90")) {

phoneNumber = "+90" + phoneNumber;

}

// [TR] Telefon numarasının toplam karakter sayısını kontrol et

// [EN] Check the total character count of the phone number

if (phoneNumber.length > 13) {

phoneNumber = phoneNumber.slice(0, 13);

}

phoneNumberInput.value = phoneNumber;

}

function checkPhoneNumber() {

var phoneNumberInput = document.getElementById("num");

var phoneNumber = phoneNumberInput.value;

// [TR] Telefon numarası girişi boş mu kontrol et

// [EN] Check if the phone number entry is empty

if (phoneNumber === "") {

alert("Lütfen sisteme kayıtlı telefon numaranızı giriniz.");

return false;

}

// [TR] Telefon numarasının "+90" ile başlamalı ve "+901231231212" formatında olmalıdır

// [EN] Phone number must start with "+90" and be in the format "+901231231212"

else if (!phoneNumber.startsWith("+90")) {

alert("Telefon numarası +90 ile başlamalıdır.");

return false;

} else if (phoneNumber.length !== 13) {

alert(

"Telefon numarası +90 ile başlamalı ve 10 karakterli telefon numarasını içermelidir. (Örn: +905301234567)"

);

return false;

} else {

return true;

}

}

**metin, ekran görüntüsü, yazı tipi, makbuz içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**YÖNETİM PANELİ (ANA SAYFA):**

const menuForm = document.querySelector(".menu-form");

const menuList = menuForm.querySelector("ul");

const addButton = menuForm.querySelector(".add-button");

// [TR] Sayfa yüklendiğinde kullanıcının menülerini getirir

// [EN] When the page is loaded, it brings the user's menus

window.addEventListener("load", async function () {

const user = await getUserInfo();

const menus = await getMenus(user.uid);

});

// [TR] Menü ekleme butonuna tıklandığında prompt ile kullanıcıdan işletme adı alınır

// [EN] When the add menu button is clicked, the user is prompted for the business name

addButton.addEventListener("click", function () {

const businessName = prompt("Lütfen işletme adını giriniz");

if (businessName) {

if (businessName.length > 20) {

alert("İşletme adı 20 karakterden fazla olamaz");

return;

}

createMenu(businessName);

}

});

// [TR] Menü düzenleme butonuna tıklandığında ilgili menüyü düzenleme sayfasına yönlendirir

// [EN] When the edit menu button is clicked, it directs the relevant menu to the edit menu page

menuList.addEventListener("click", function (e) {

if (e.target.classList.contains("edit-button")) {

const menuItem = e.target.parentElement.parentElement;

const menuID = menuItem.querySelector(".id").innerText.slice(1, -1);

window.location.href = "./edit-menu.html?" + menuID;

}

});

// [TR] Menü listesindeki sil butonuna tıklandığında ilgili item onay alınarak silinir.

// [EN] When the delete button in the menu list is clicked, the relevant item is deleted with confirmation.

menuList.addEventListener("click", function (e) {

if (e.target.classList.contains("delete-button")) {

const menuItem = e.target.parentElement.parentElement;

const menuName = menuItem.querySelector(".form-list-item-text").innerText;

const menuID = menuItem.querySelector(".id").innerText.slice(1, -1);

const result = confirm(

menuName + " menüsünü silmek istediğinize emin misiniz?"

);

if (result) {

deleteMenu(menuID);

menuItem.remove();

}

}

});

// [TR] Veritabanına bağlanarak menüyü oluşturur

// [EN] Connects to the database and creates the menu

async function createMenu(menuName) {

var menuID = generateId();

const user = await getUserInfo();

// [TR] Menü ID'si özgün değilse yeni bir ID oluşturur.

// [EN] If the menu ID is not unique, it creates a new ID.

const menu = await db.collection("menus").doc(menuID).get();

if (menu.exists) {

menuID = generateId();

}

// [TR] Menüyü veritabanına kaydeder

// [EN] Saves the menu to the database

db.collection("menus")

.doc(menuID)

.set({

name: menuName,

id: menuID,

created\_at: firebase.firestore.FieldValue.serverTimestamp(),

authorized\_name: user.displayName,

authorized\_phone: user.phoneNumber,

authorized\_uid: user.uid,

})

.then(() => {

addItemToList(menuList, menuName, menuID);

})

.catch((error) => {

throw error;

});

}

// [TR] Menüyü veritabanından siler

// [EN] Deletes the menu from the database

async function deleteMenu(menuID) {

console.log(menuID, "silindi");

db.collection("menus")

.doc(menuID)

.delete()

.catch((error) => {

throw error;

});

}

// [TR] Kullanıcı bilgilerini alır ve döndürür

// [EN] Gets and returns user information

async function getUserInfo() {

return new Promise((resolve, reject) => {

firebase.auth().onAuthStateChanged((user) => {

if (user) {

console.log(user);

resolve(user);

} else {

resolve(false);

}

});

});

}

// [TR] Menü için rastgele bir id oluşturur

// [EN] Generates a random id for the menu

function generateId() {

const letters = "abcdefghijklmnopqrstuvwxyz";

let id = "";

for (let i = 0; i < 6; i++) {

id += letters[Math.floor(Math.random() \* letters.length)];

}

return id;

}

// [TR] Veritabanına bağlanarak kullanıcının menü içeriklerini getirir ve ekler

// [EN] Connects to the database and retrieves and adds the user's menu content

async function getMenus(userUID) {

const menus = await getUsersMenus(userUID);

console.log(menus);

menus.forEach((menuID) => {

db.collection("menus")

.doc(menuID)

.get()

.then((doc) => {

const menu = doc.data();

addItemToList(menuList, menu.name, menu.id);

});

});

}

// [TR] Veritabanına bağlanarak kullanıcının menülerini getirir

// [EN] Connects to the database and retrieves the user's menus

async function getUsersMenus(userUID) {

const menus = await db

.collection("menus")

.where("authorized\_uid", "==", userUID)

.get();

return menus.docs.map((doc) => doc.id);

}

// [TR] Menü listesine yeni bir item ekler

// [EN] Adds a new item to the menu list

function addItemToList(form, item, id) {

const li = document.createElement("li");

li.innerHTML = `

<div class="form-list-item-text">

<span>${item} <span class="id">(${id})</span></span>

</div>

<div class="form-list-item-buttons">

<button class="edit-button">Düzenle</button>

<button class="delete-button">Sil</button>

</div>

`;

form.appendChild(li);

return li;

}

**metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**YÖNETİM PANELİ (MENÜ DÜZENLEME SAYFASI):**

const menuForm = document.querySelector(".menu-form");

const menuList = menuForm.querySelector("ul");

const addButton = menuForm.querySelector(".add-button");

// [TR] Sayfa yüklendiğinde kullanıcının menülerini getirir

// [EN] When the page is loaded, it brings the user's menus

window.addEventListener("load", async function () {

const user = await getUserInfo();

const menus = await getMenus(user.uid);

});

// [TR] Menü ekleme butonuna tıklandığında prompt ile kullanıcıdan işletme adı alınır

// [EN] When the add menu button is clicked, the user is prompted for the business name

addButton.addEventListener("click", function () {

const businessName = prompt("Lütfen işletme adını giriniz");

if (businessName) {

if (businessName.length > 20) {

alert("İşletme adı 20 karakterden fazla olamaz");

return;

}

createMenu(businessName);

}

});

// [TR] Menü düzenleme butonuna tıklandığında ilgili menüyü düzenleme sayfasına yönlendirir

// [EN] When the edit menu button is clicked, it directs the relevant menu to the edit menu page

menuList.addEventListener("click", function (e) {

if (e.target.classList.contains("edit-button")) {

const menuItem = e.target.parentElement.parentElement;

const menuID = menuItem.querySelector(".id").innerText.slice(1, -1);

window.location.href = "./edit-menu.html?" + menuID;

}

});

// [TR] Menü listesindeki sil butonuna tıklandığında ilgili item onay alınarak silinir.

// [EN] When the delete button in the menu list is clicked, the relevant item is deleted with confirmation.

menuList.addEventListener("click", function (e) {

if (e.target.classList.contains("delete-button")) {

const menuItem = e.target.parentElement.parentElement;

const menuName = menuItem.querySelector(".form-list-item-text").innerText;

const menuID = menuItem.querySelector(".id").innerText.slice(1, -1);

const result = confirm(

menuName + " menüsünü silmek istediğinize emin misiniz?"

);

if (result) {

deleteMenu(menuID);

menuItem.remove();

}

}

});

// [TR] Veritabanına bağlanarak menüyü oluşturur

// [EN] Connects to the database and creates the menu

async function createMenu(menuName) {

var menuID = generateId();

const user = await getUserInfo();

// [TR] Menü ID'si özgün değilse yeni bir ID oluşturur.

// [EN] If the menu ID is not unique, it creates a new ID.

const menu = await db.collection("menus").doc(menuID).get();

if (menu.exists) {

menuID = generateId();

}

// [TR] Menüyü veritabanına kaydeder

// [EN] Saves the menu to the database

db.collection("menus")

.doc(menuID)

.set({

name: menuName,

id: menuID,

created\_at: firebase.firestore.FieldValue.serverTimestamp(),

authorized\_name: user.displayName,

authorized\_phone: user.phoneNumber,

authorized\_uid: user.uid,

})

.then(() => {

addItemToList(menuList, menuName, menuID);

})

.catch((error) => {

throw error;

});

}

// [TR] Menüyü veritabanından siler

// [EN] Deletes the menu from the database

async function deleteMenu(menuID) {

console.log(menuID, "silindi");

db.collection("menus")

.doc(menuID)

.delete()

.catch((error) => {

throw error;

});

}

// [TR] Kullanıcı bilgilerini alır ve döndürür

// [EN] Gets and returns user information

async function getUserInfo() {

return new Promise((resolve, reject) => {

firebase.auth().onAuthStateChanged((user) => {

if (user) {

console.log(user);

resolve(user);

} else {

resolve(false);

}

});

});

}

// [TR] Menü için rastgele bir id oluşturur

// [EN] Generates a random id for the menu

function generateId() {

const letters = "abcdefghijklmnopqrstuvwxyz";

let id = "";

for (let i = 0; i < 6; i++) {

id += letters[Math.floor(Math.random() \* letters.length)];

}

return id;

}

// [TR] Veritabanına bağlanarak kullanıcının menü içeriklerini getirir ve ekler

// [EN] Connects to the database and retrieves and adds the user's menu content

async function getMenus(userUID) {

const menus = await getUsersMenus(userUID);

console.log(menus);

menus.forEach((menuID) => {

db.collection("menus")

.doc(menuID)

.get()

.then((doc) => {

const menu = doc.data();

addItemToList(menuList, menu.name, menu.id);

});

});

}

// [TR] Veritabanına bağlanarak kullanıcının menülerini getirir

// [EN] Connects to the database and retrieves the user's menus

async function getUsersMenus(userUID) {

const menus = await db

.collection("menus")

.where("authorized\_uid", "==", userUID)

.get();

return menus.docs.map((doc) => doc.id);

}

// [TR] Menü listesine yeni bir item ekler

// [EN] Adds a new item to the menu list

function addItemToList(form, item, id) {

const li = document.createElement("li");

li.innerHTML = `

<div class="form-list-item-text">

<span>${item} <span class="id">(${id})</span></span>

</div>

<div class="form-list-item-buttons">

<button class="edit-button">Düzenle</button>

<button class="delete-button">Sil</button>

</div>

`;

form.appendChild(li);

return li;

}

**metin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**MENÜ ZİYARETÇİ GÖRÜNÜMÜ:**

// [TR] URL'den menü ID'sini al

// [EN] Get menu ID from URL

const url = window.location.href;

const urlParamsIndex = url.indexOf("?");

const menuID = url.substring(urlParamsIndex + 1);

// [TR] Firebase veritabanından menüyü al

// [EN] Get menu from Firebase database

async function getMenu(menuID) {

return new Promise((resolve, reject) => {

const db = firebase.firestore();

db.collection("menus")

.doc(menuID)

.get()

.then((doc) => {

if (doc.exists) {

resolve(doc.data());

} else {

reject("No such document!");

}

})

.catch((error) => {

reject(error);

});

});

}

// [TR] Menüyü ekrana yazdır

// [EN] Print menu to screen

async function printMenu() {

const menu = await getMenu(menuID);

console.log(menu);

// [TR] Sayfa başlığı ve şirket adı

// [EN] Page title and company name

document.title = menu.name + " - QR Menu";

document.querySelector(".company-name").innerHTML = menu.name

? menu.name

: "";

// [TR] Hoşgeldiniz metni

// [EN] Welcome text

document.querySelector(".welcome-text").innerHTML = menu.welcome\_text

? menu.welcome\_text

: "";

// [TR] Menü listesini döngü ile oluştur

// [EN] Create menu list with loop

menu.menu.forEach((category) => {

const categoryName = Object.keys(category)[0];

console.log(categoryName);

// [TR] Kategori başlıklarını oluştur ve sayfaya ekle

// [EN] Create category titles and add to page

const categoryItem = document.createElement("div");

categoryItem.classList.add("category");

const categoryTitle = document.createElement("h2");

categoryTitle.classList.add("category-title");

categoryTitle.innerHTML = categoryName;

categoryItem.appendChild(categoryTitle);

document.querySelector(".menu").appendChild(categoryItem);

// [TR] Ürünleri oluştur ve sayfaya ekle

// [EN] Create products and add to page

category[categoryName].forEach((product) => {

const productItem = document.createElement("div");

productItem.classList.add("menu-item");

const menuItemInfo = document.createElement("div");

menuItemInfo.classList.add("menu-item-info");

// [TR] Ürün adı

// [EN] Product name

const menuItemName = document.createElement("h3");

menuItemName.textContent = product.title;

menuItemInfo.appendChild(menuItemName);

// [TR] Ürün açıklaması

// [EN] Product description

const menuItemDescription = document.createElement("p");

menuItemDescription.classList.add("menu-item-description");

menuItemDescription.textContent = product.description || "";

menuItemInfo.appendChild(menuItemDescription);

// [TR] Ürün fiyatı

// [EN] Product price

const menuItemPrice = document.createElement("span");

menuItemPrice.classList.add("menu-item-price");

menuItemPrice.textContent = product.price;

menuItemInfo.appendChild(menuItemPrice);

productItem.appendChild(menuItemInfo);

categoryItem.appendChild(productItem);

});

});

removeLoading();

}

printMenu();

// [TR] Sayfa yükleniyor animasyonunu kaldır

// [EN] Remove loading animation

function removeLoading() {

const loader = document.getElementById("loader");

loader.classList.add("hide");

}

**Projenize ait veri tabanı tablolarının veya diyagramın görsellerini ekleyiniz.**

ekran görüntüsü, metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

ekran görüntüsü, metin içeren bir resim

Açıklama otomatik olarak oluşturuldu