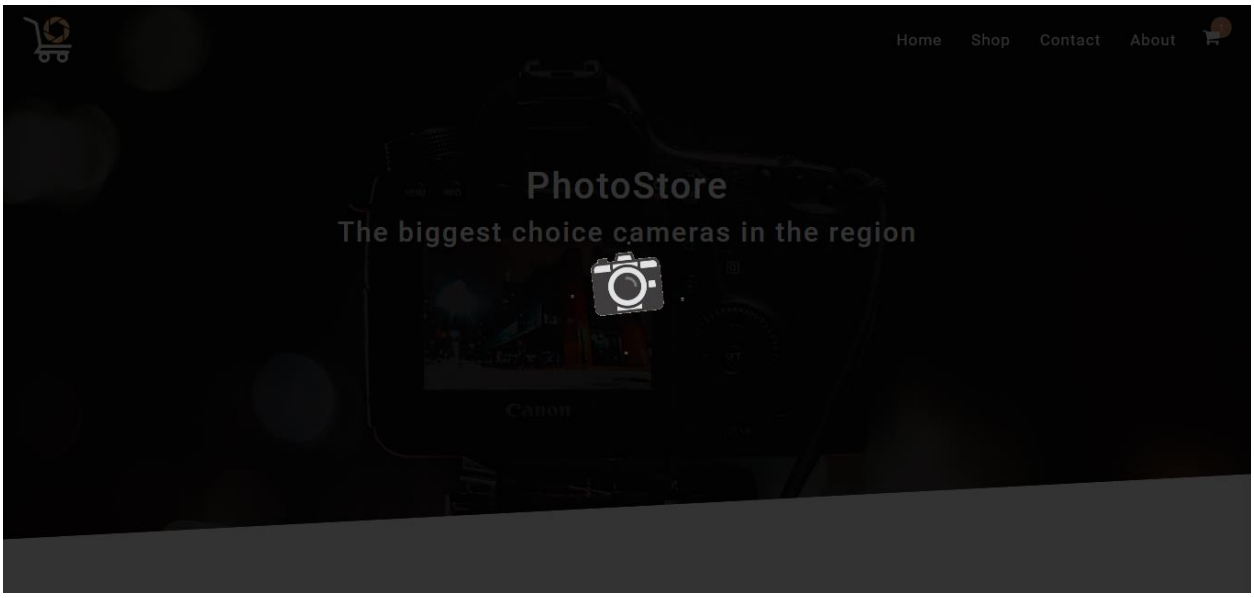
Vanja Veapi 244/19

WEB PROGRAMIRANJE 2

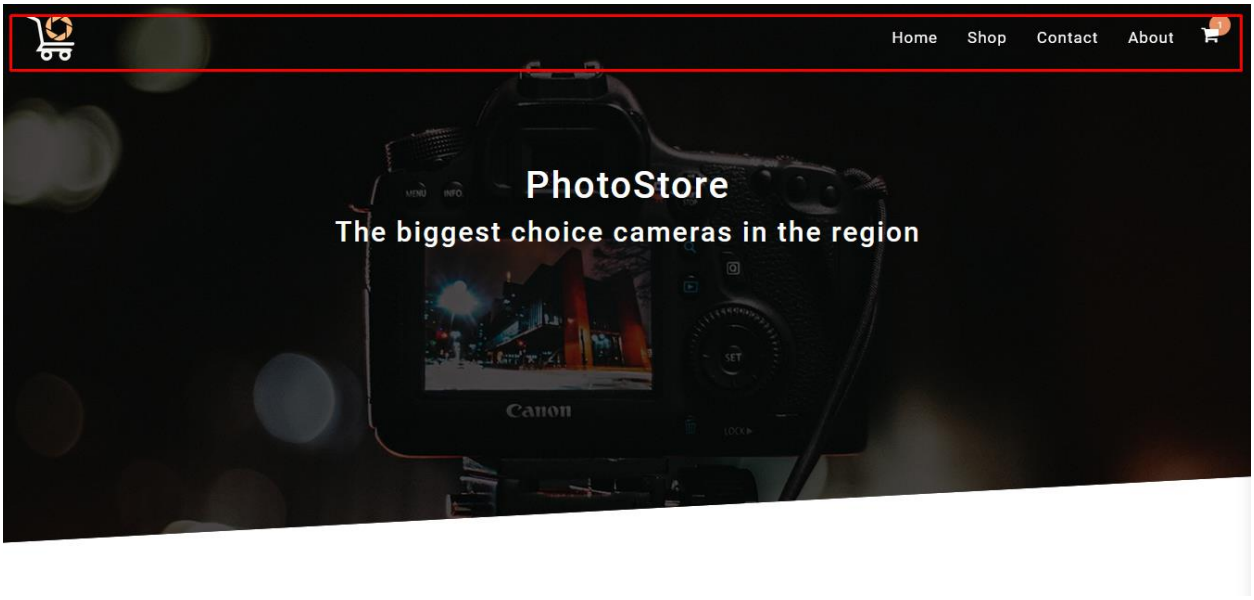
Sadrzaj

Izgled stranica

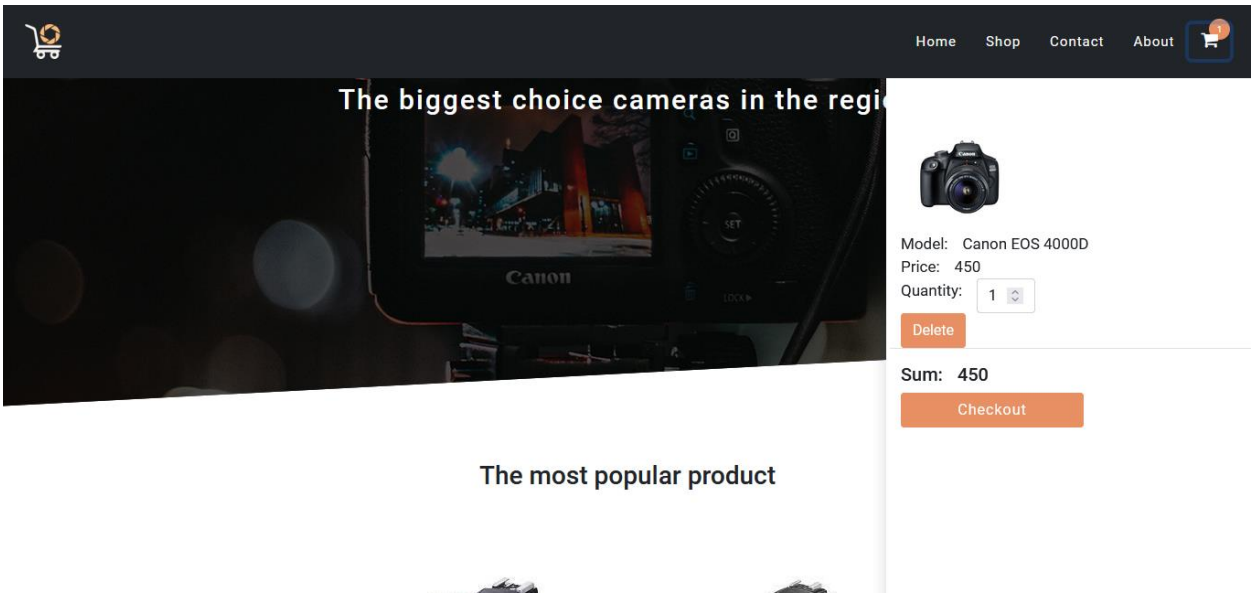
Po ucitavanju svake stranice pojavljuje se loader



Header I footer su isti na svim stranicama



Meni je transparentan dok je na vrhu, kako krenemo da skrolamo dobija pozadinu I font mu se smanjuje



Klikom na ikonicu korpe, otvara nam se sidebar u kom se nalaze proizvodi. Dugme checkout ce nas redirektovati na contact.html koji u ovom slucaju predstavlja formu za narucivanje.

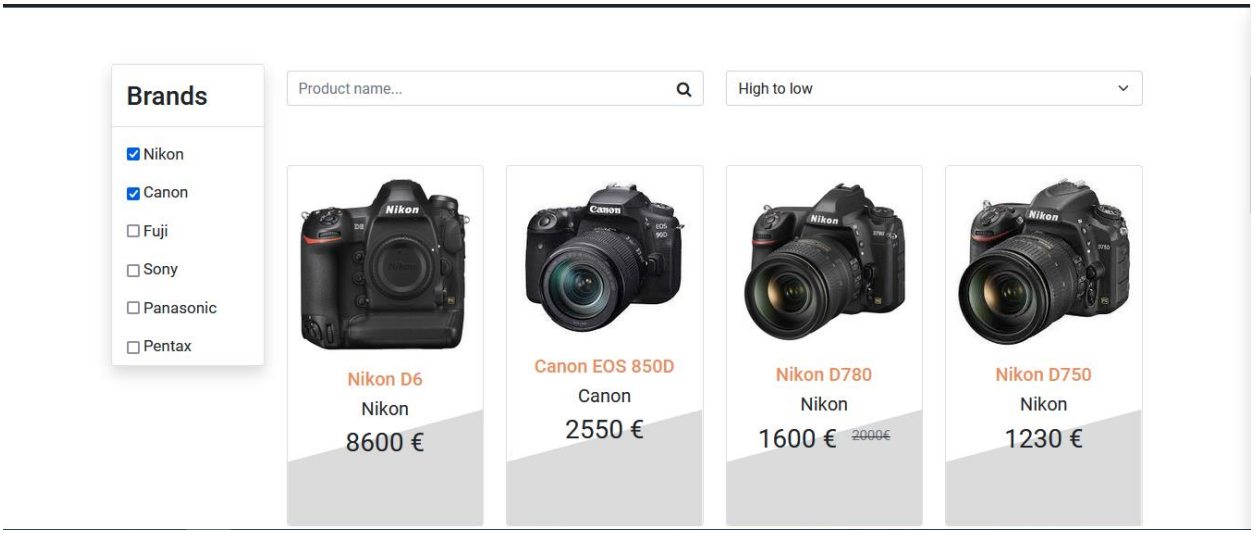
Index.html

Na pocetnoj stranici se u prvoj sekciji nalazi prikaz 2 najpopularnija proizvoda, naziv I cene.

Footer(zajedinicki za sve)

Sadrzi parallax sliku, linkove ka drustvenim mrezama I dokumentaciji.

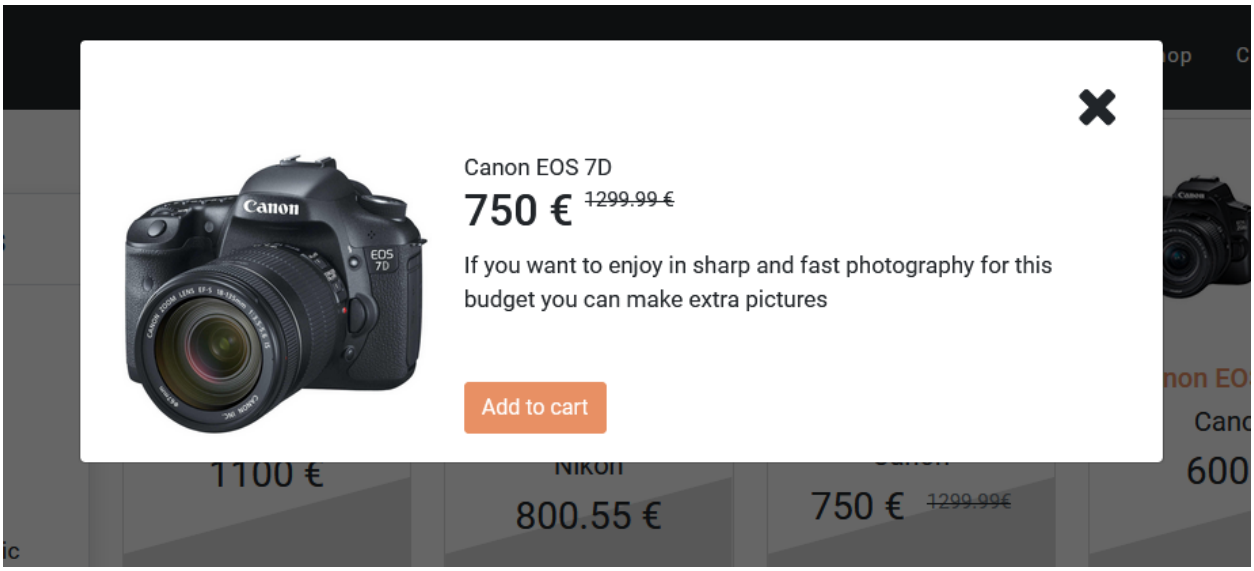
Shop.html

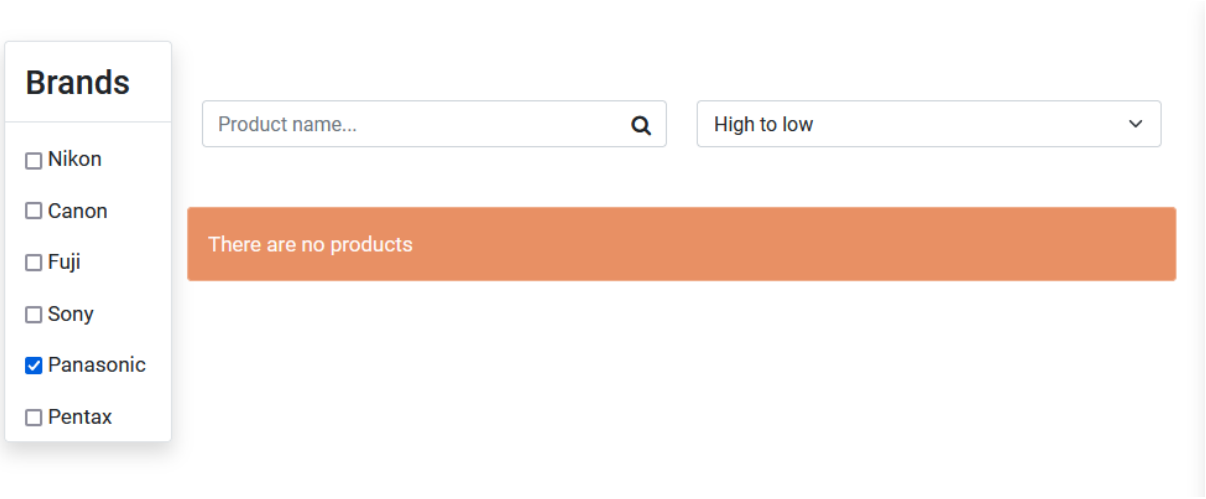


U shopu se nalazi checkbox gde koji sluzi sa filtriranje po brendovima, takodje postoji mogucnost filtriranja po nazivu modela I sortiranje po ceni.

Deo sa brendovim, na velikim ekranima prati skrol. Na dnu se nalazi paginacija.

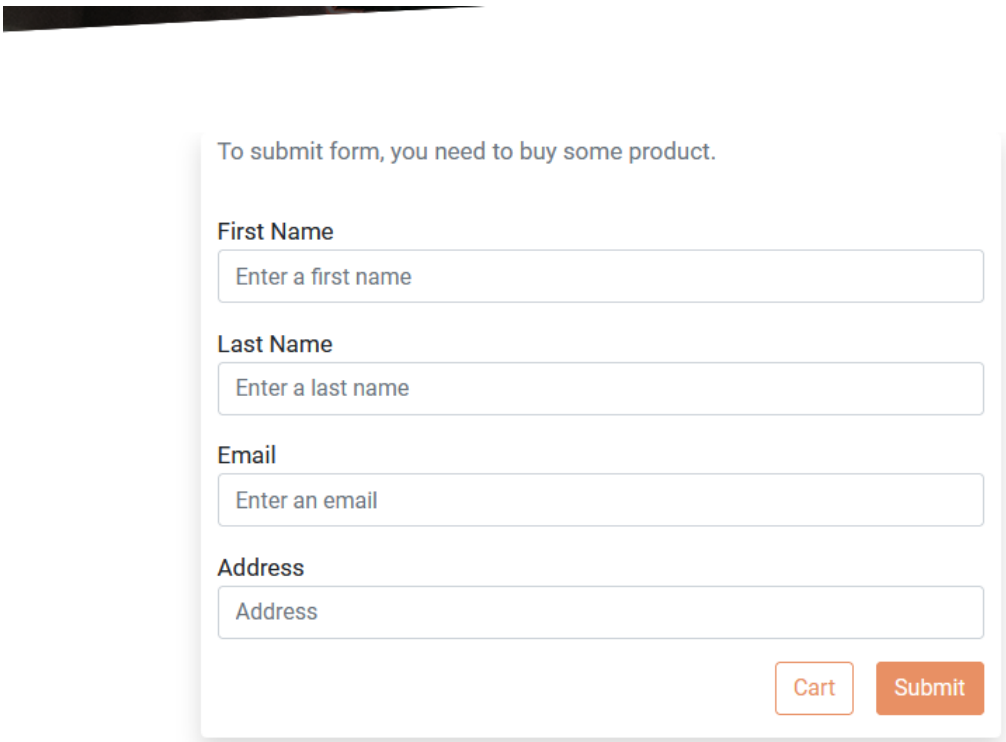
Klikom na neke od modela, otvorice se modal, koji sadrzi I opis(ako ga proizvod ima) I sadrzi dugme da se doda u korpu. Kada se klikne dugme da se doda u korpu, pojavljuje se informacija da je sadrzaj dodat u korpu





Contact.html

Na kontakt strani se nalazi forma, koja je obavezna da se popuni, ako se ne popuni izbacuju se odredjene greske. Takodje u formi postoji dugme cart(koje ima istu funkciju kao I ikonica korpica). Kada se sve popuni cisti se korpa iz local storage I redirektuje nas na pocetnu stranicu sa obavestenjem da smo uspesno narucili proizvod.


To submit form, you need to buy some product.
First Name
Enter a first name
Last Name
Enter a last name
Email
Enter an email
Address
Address
Cart
Submit

Main.js

Reusecode

```
nameId.addEventListener("blur", () => checkField(nameId, regExpName, errName));
        lastName.addEventListener("blur", () => checkField(lastName, regExpName,
errLastName));
        email.addEventListener("blur", () => checkField(email, regExpMail,
errEmail));
        address.addEventListener("blur", () => checkField(address, regExpAddr,
errAddress));
```

```
const BASE_URL = "assets/data";

const header = document.querySelector("header");
const products = document.querySelector("#products");
const search = document.querySelector("#search");
const cart = document.querySelector("#cart");
const hamburger = document.querySelector("#hamburger");
const modal = document.querySelector("#modal");
const scrollY = window.scrollY;

const mobileMenu = document.querySelector("#mobile-menu");
const productModal = document.querySelector("#product-modal");
const closeModal = document.querySelector("#close-modal");

const itemsInCart = document.querySelector("#items-in-cart");

let allCameras = []; // allCameras take all cameras and put in this array, for
global purpose.
let brandsArr = []; // BrandsArr take all brands and put in this array, for
global purpose.

let searchQuery = "";
let sortType = localStorage.getItem("sort");
let filterCameraBrandsArr = JSON.parse(localStorage.getItem("cameras")) === null
? [] : JSON.parse(localStorage.getItem("cameras"));
let cartCount = localStorage.getItem("cart-counter") === null ? 0 :
localStorage.getItem("cart-counter");

let isCartOpen = false;

const RECORDS_PER_PAGE = 10;
let currentPage = 1;
```

```
let activePage = 0;
// Contact.html
const btnSubmit = document.querySelector("#btn-submit");
const btnCart = document.querySelector("#btn-cart");

const nameId = document.querySelector("#name");
const lastName = document.querySelector("#last-name");
const email = document.querySelector("#email");
const address = document.querySelector("#address");
const regExpName = /^[A-ZŠĐŽČĆ][a-zšđžčć]{2,20}$/;
const regExpMail = /^[a-z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-
]+)+$/;
const regExpAddr = /[A-ZČĆŽŠĐ1-9]([a-zčćžšđ0-9]{2,80}\s)+([0-9]{1,4}[a-
zžđ]*|bb)$/;

const infoModal = document.querySelector("#info-modal-container");
let infoModalCounter = 0;

window.addEventListener("load", function () {
    onReady(onReadyCallback);

    fetchData(BASE_URL + "/nav.json", renderLinks);
    // checkActivePage()

    this.scrollY >= 50 ? header.classList.add("bg-dark", "size") :
header.classList.remove("bg-dark", "size");

    if (this.window.location.pathname === "/shop.html") {
        search.value = "";
        fetchData(BASE_URL + "/brands.json", renderBrands);
    }

    if (window.location.pathname === "/index.html" || window.location.pathname
=== "/" || window.location.pathname === "/contact.html") {
        fetchData(BASE_URL + "/cameras.json", renderCameras);
    }

    if (this.window.location.pathname === "/contact.html") {
        btnCart.addEventListener("click", toggleCart);
        btnSubmit.addEventListener("click", submit);

        nameId.addEventListener("blur", () => checkField(nameId, regExpName,
errName));
        lastName.addEventListener("blur", () => checkField(lastName, regExpName,
errLastName));
```

```
        email.addEventListener("blur", () => checkField(email, regExpMail,
errEmail));
        address.addEventListener("blur", () => checkField(address, regExpAddr,
errAddress));
    }

    modal.style.transform = "translateX(0%)";
    cart.addEventListener("click", toggleCart);
    hamburger.addEventListener("click", toggleMenu);
    itemsInCart.innerHTML = cartCount;
});

window.addEventListener("scroll", function () {
    if (this.scrollY >= 50) {
        header.classList.add("bg-dark", "size");
    } else if (this.scrollY <= 50 && mobileMenu?.classList.contains("d-none")) {
        header.classList.remove("bg-dark");
        modal.style.removeProperty("top");
    }

    if (this.scrollY <= 50) {
        header.classList.remove("size");
    }
});

function onReady(callback) {
    var intervalId = window.setInterval(function () {
        if (document.getElementsByTagName("body")[0] !== undefined) {
            window.clearInterval(intervalId);
            callback.call(this);
        }
    }, 1000);
}
function onReadyCallback() {
    let isVisible = setVisible("#loading", false);
    if (isVisible === "none") {
        document.querySelector("body").classList.remove("overflow-hidden");
    }
}

function setVisible(selector, visible) {
    return (document.querySelector(selector).style.display = visible ? "block" :
"none");
}
function fetchData(path, callback, method = "GET") {
```

```
    $.ajax({
        url: `${path}`,
        method,
        dataType: "json",
        success: function (data) {
            callback(data);
        },
        error: function (error) {
            console.log(error);
        },
    });
}

function sorting(data, sortType) {
    data.sort((a, b) => {
        if (sortType === "asc") {
            return a.price.current - b.price.current;
        } else if (sortType === "desc") {
            return b.price.current - a.price.current;
        }
    });

    return data;
}

// Brands
function renderBrands(brands) {
    const brandsID = document.querySelector("#brands");
    let html = "";

    for (let brand of brands) {
        html += makeListItem(brand, "brand");
    }
    brandsArr = brands;
    brandsID.innerHTML = html;

    fetchData(BASE_URL + "/cameras.json", renderCameras);
}

function renderLinks(links) {
    const usefulLeft = document.querySelector("#useful-left");
    const usefulRight = document.querySelector("#useful-right");
    const ulDesktopMenu = document.querySelector("#ul-desktop-menu");
    const ulMobileMenu = document.querySelector("#ul-mobile-menu");
```

```
    addLinks(links, ulDesktopMenu, [0, 3], "list-group-item d-none d-sm-none d-
md-block dinamicanIspis");
    addLinks(links, ulMobileMenu, [0, 3], "list-group-item d-block d-md-none");
    addLinks(links, usefulLeft, [4, 5]); //Ovo od 4 do 5 treba da ide i da iz
linka ispise te elemente
    addLinks(links, usefulRight, [6, 9]);
}

function addLinks(data, idAttr, len, className = null) {
    let html = "";
    data.forEach((link, index) => {
        if (index >= len[0] && index < len[1]) {
            html += makeListItem(link.name, className, link.href);
        } else if (index >= len[0] && index <= len[1]) {
            html += makeListItem(link.name, className, link.href);
        }
    });

    idAttr.innerHTML = html;
}

// Cameras
function renderCameras(cameras) {
    allCameras = cameras;
    if (window.location.pathname === "/index.html" || window.location.pathname
=== "/" || window.location.pathname === "/contact.html") {
        //Samo da mi pokupi sve kamere iz jsona i da ih stavi u promenljivu
        return;
    }
    const filteredArray = filtered(allCameras, searchQuery, sortType,
filterCameraBrandsArr, currentPage);
    let pageNumber = null;
    let totalPages = 1;

    pageNumber = numPages(filteredArray);

    updatePageNumber(pageNumber);
    addCameras(filteredArray, currentPage - 1);
    refreshPageNumber();

    const brands = document.querySelectorAll(".brand");
    const sort = document.querySelector("#sort");

    //Search cameras in input field
    search.addEventListener("input", function (e) {
```

```
        searchQuery = e.target.value.toLowerCase();
        const filteredArray = filtered(allCameras, searchQuery, sortType,
filterCameraBrandsArr, currentPage);

        // Ovde treba da se desi promena broja stranice
        totalPages = numPages(filteredArray);
        updatePageNumber(totalPages);

        refreshScreen(filteredArray);
    });

    // Brands checkboxes
    brands.forEach((brand) => {
        if (filterCameraBrandsArr.some((id) => id === Number(brand.value))) {
            brand.checked = true;
        }
        brand.addEventListener("change", function () {
            if (this.checked) {
                filterCameraBrandsArr.push(Number(this.value));
                localStorage.setItem("cameras",
JSON.stringify(filterCameraBrandsArr));
            } else {
                let index = filterCameraBrandsArr.indexOf(Number(this.value));
                filterCameraBrandsArr.splice(index, 1);
                localStorage.setItem("cameras",
JSON.stringify(filterCameraBrandsArr));
            }

            const filteredArray = filtered(allCameras, searchQuery, sortType,
filterCameraBrandsArr, currentPage);

            // Ovde treba da se desi promena broja stranice
            totalPages = numPages(filteredArray);
            updatePageNumber(totalPages);

            refreshScreen(filteredArray);
        });
    });

    //Cameras Sort
    for (let i = 0; i < sort.length; i++) {
        if (sort[i].value === localStorage.getItem("sort")) {
            sort[i].selected = true;
        }
    }
```

```
    sort.addEventListener("change", function () {
        sortType = this.value;
        const filteredArray = filtered(allCameras, searchQuery, sortType,
filterCameraBrandsArr, currentPage);

        addCameras(filteredArray, currentPage - 1);
        localStorage.setItem("sort", sortType);
    });
}

function refreshScreen(cameras, pageIndex = 0) {
    addCameras(cameras, pageIndex);
    refreshPageNumber();
}

function refreshPageNumber() {
    const pages = document.querySelectorAll(".pages");

    if (pages[activePage] === undefined) {
        // USAO SAM OVDE
        console.log(currentPage);
        // return refreshScreen(allCameras, currentPage - 1)
        activePage = currentPage - currentPage;
    }

    if (pages[activePage] !== undefined) {
        pages[activePage].classList.add("active");
    }

    pages.forEach((page) =>
        page.addEventListener("click", function (e) {
            e.preventDefault();

            currentPage = e.target.dataset.id;
            if (e.target !== this || e.target) {
                this.classList.add("active");
            }
            if (activePage != currentPage - 1) {
                console.log(activePage, currentPage);
                pages[activePage].classList.remove("active");
            }
            activePage = currentPage - 1;
            console.log("---[ACTIVE PAGE]---");
            console.log(activePage);
```

```
            const filteredArray = filtered(allCameras, searchQuery, sortType,
filterCameraBrandsArr, currentPage);

            // refreshScreen(filteredArray, activePage);
            refreshScreen(filteredArray, activePage);
            console.log("---[PAGES EVENT LISTENER (CLICK)]---");
            console.log(filteredArray);
            console.log(activePage);
            // console.log(filteredArray);
            // console.log("Menajj strani");
        })
    );
}
function addCameras(cameras, pageIndex) {
    let html = "";

    for (let i = pageIndex * RECORDS_PER_PAGE; i < (pageIndex + 1) *
RECORDS_PER_PAGE; i++) {
        if (cameras[i]) html += makeCamera(cameras[i]);
    }

    //If there is no product, return notification...
    if (cameras.length === 0) {
        html = '<div class="alert alert-warning" role="alert">There are no
products</div>';
    }
    products.innerHTML = html;

    //Ako mi se budu pravili neki infiniti loopovi moguce da je ovde problem
    const productCards = document.querySelectorAll(".product-cards");
    productCards.forEach((card) => {
        card.addEventListener("click", toggleProductModal);
    });

    closeModal.addEventListener("click", function () {
        productModal.classList.toggle("show");
        productModal.style.display = "none";
    });

    productModal.addEventListener("click", function () {
        productModal.classList.toggle("show");
        productModal.style.display = "none";
    });
}
```

```
function makeCamera(camera) {
    let cameraBrand = brandsArr.find((brand) => brand.id ===
camera.brandId).name;

    return `<div class="col-12 col-md-6 col-lg-4 col-xl-3 mt-3 product-cards"
data-id="${camera.id}">
    <div class="card text-center position-relative">
        <div class="d-flex justify-content-center align-items-center">
            <img src="assets/images/products/${camera.img.src}"
alt="${camera.img.alt}" width="250" class="img-fluid"/>
        </div>
        <h4 class="text-orange h5 mt-3">${camera.name}</h4>
        <h6 class="h5">${cameraBrand}</h6>
        <div class="price d-flex justify-content-center mb-3">
            <p class="h3">${camera.price.current} &euro;</p>
            ${camera.price.old === null ? "" : `<s class="ms-3"><small
class="text-muted">${camera.price.old}&euro;</small></s>`}
        </div>
    </div>
</div>`;
}

/**
 * @param {object} item - Data from json
 * @param {String} className
 * @returns
 */
function makeListItem(item, className, aHref = null) {
    if (aHref === null) {
        return `<li class="list-group-item">
        <input type="checkbox" value="${item.id}" class="${className}"
name="${className}"/> ${item.name}
    </li>`;
    }

    if (className === null) {
        return `<li class="list-group-item"><a href="${aHref}">${item}</a></li>`;
    }

    return `<li class="${className}"><a href="${aHref}">${item}</a></li>`;
}

/**
 * Function which pair all sorts and functions...
 * @param {object} data
```

```
 * @param {string} searchQuery
 * @returns
 */
function filtered(data, searchQuery, sortQuery, cameraArr, currentPage) {
    let filtered = data.filter((e) =>
e.name.toLowerCase().includes(searchQuery));
    filtered = sorting(filtered, sortQuery);

    //If brands array has no length 0, then enter in if adn go through all
filtered(cameras) which brandId is = with  numberId in array...
    if (cameraArr.length !== 0) {
        filtered = filtered.filter((e) => cameraArr.includes(e.brandId));
    }
    return filtered;
}

/**
 * @returns Number of page on site.
 */
function numPages(objJSON) {
    return Math.ceil(objJSON.length / RECORDS_PER_PAGE);
}
function changePage(pageNum, camerasData) {
    let productsOnPage = [];

    /*Formula za for je sledeca i = 2 * 10 = 20 meni ne postoji camerasData[20] i
zato on vraca undefined, meni treba da ovo i bude 0 i da krene od 0 elemtna */
    for (let i = (pageNum - 1) * RECORDS_PER_PAGE; i < pageNum *
RECORDS_PER_PAGE; i++) {
        if (camerasData[i] === undefined) {
            break;
        }
        productsOnPage.push(camerasData[i]);
    }
    return productsOnPage;
}
function updatePageNumber(pages) {
    console.log("Aktivirao sam funkciju AddPageNumber");
    const pagination = document.querySelector("#pagination");

    let html = "";
    for (let i = 0; i < pages; i++) {
        html += `<li class="list-group-item border pages" data-id="${i + 1}"><a
href="#" class="text-dark" data-id="${i + 1}">${i + 1}</a></li>`;
    }
```

```
    pagination.innerHTML = html;
}
function toggleCart() {
    let cart = JSON.parse(localStorage.getItem("cart"));
    console.log(cart);
    if (!isCartOpen) {
        if (cart?.products.length === 0 || cart === null) {
            cart = { products: [] };
            renderCart(cart);
        } else {
            renderCart(cart);
            calculateTotalCash();
            let btnDelete = document.querySelectorAll(".btn-delete");
            console.log(btnDelete);

            btnDelete.forEach((btn) => {
                btn.addEventListener("click", function () {
                    removeFromCart(Number(this.dataset.id));
                    this.parentElement.parentElement.remove();
                    calculateTotalCash();
                });
            });

            let inputQuantity = document.querySelectorAll(".input-quantity");
            inputQuantity.forEach((el, index) => {
                el.addEventListener("change", function () {
                    let id = Number(this.dataset.id);
                    let quantity = this.valueAsNumber;
                    let total = document.querySelectorAll(".total")[index];
                    let pricePerProduct = document.querySelectorAll(".price-per-
product")[index];

                    if (quantity <= 1) {
                        quantity = 1;
                        this.value = quantity;
                    }

                    total.innerHTML = Number(pricePerProduct.value) * quantity;

                    addToCart(id, quantity);
                    calculateTotalCash();
                });
            });
        }
```

```
    }

    isCartOpen = !isCartOpen;
    document.body.classList.toggle("overflow-hidden");
    return modal.style.transform === "translateX(0%)" ? (modal.style.transform =
"translateX(-100%)") : (modal.style.transform = "translateX(0%)");
}

function toggleMenu() {
    mobileMenu.classList.toggle("top-100");
    if (window.scrollY <= 50) {
        header.classList.toggle("bg-dark");
    }

    if (mobileMenu.classList[6] === "d-none") {
        return mobileMenu.classList.toggle("d-none");
    }

    mobileMenu.classList.toggle("d-none");
}
function toggleProductModal() {
    const cameraId = Number(this.dataset.id);
    const camera = findCamera(cameraId);
    showCamera(camera);
}
function findCamera(cameraId) {
    return allCameras.find((cam) => cam.id === cameraId);
}
function showCamera(model) {
    productModal.classList.add("show");
    productModal.style.display = "block";

    const productContent = document.querySelector("#product-content");
    productContent.innerHTML = `<div class='container'>
        <div class='row'>
            <div class='col-md-12 col-lg-4'>
                <img src="assets/images/products/${model.img.src}"
alt="${model.img.alt}" class="img-fluid"/>
            </div>
            <div class='col-md-12 col-lg-8'>
                <p>${model.name}</p>
                <div class="d-flex">
                    <p class="h2">${model.price.current} &euro;</p>
                    ${model.price.old === null ? "" : `<small class="ms-
2"><s>${model.price.old} &euro;</s></small>`}
```

```
                </div>
                <p>${model.description === null ? "Product has no description." :
model.description}</p>
                <button id="add-to-cart" class="btn btn-primary mt-5" data-
id=${model.id}>Add to cart</button>
            </div>
        </div>
    </div>`;

    const addCartButton = document.querySelector("#add-to-cart");
    addCartButton.addEventListener("click", function () {
        let itemsCart = Number(itemsInCart.innerText);
        let cart = null;

        if (infoModalCounter === 4) {
            infoModal.innerHTML = "";
            infoModalCounter = 0;
        }

        displayInfoCard("You have successfully added an item to the cart",
infoModal);
        infoModalCounter++; // infoModal.classList.remove("d-none");

        if (JSON.parse(localStorage.getItem("cart")) !== null) {
            cart = JSON.parse(localStorage.getItem("cart"));
            let proba = cart.products.find((item) => item.id === model.id);
            if (proba === undefined) {
                itemsInCart.innerHTML = itemsCart += 1;
            }
        } else {
            itemsInCart.innerHTML = itemsCart += 1;
        }

        localStorage.setItem("cart-counter", itemsInCart.innerHTML);
        addToCart(model.id);
    });
}
function fetchCart(porudzbina, id) {
    return porudzbina.products.find((e) => e.id === id);
}
function addToCart(modelId, quantity = null) {
    let order = JSON.parse(localStorage.getItem("cart"));

    if (!order) {
        order = {
```

```
            products: [],
        };
    }

    const article = fetchCart(order, modelId);
    if (article) {
        if (quantity === null) {
            article.quantity += 1;
        } else {
            article.quantity = quantity;
        }
    } else {
        order.products.push({
            id: modelId,
            quantity: 1,
        });
    }

    return localStorage.setItem("cart", JSON.stringify(order));
}
function removeFromCart(modelId) {
    const id = modelId;

    let order = JSON.parse(localStorage.getItem("cart"));
    const article = fetchCart(order, id);

    let newState = order.products.filter((e) => e.id != article.id);
    newState = {
        products: newState,
    };
    itemsInCart.innerHTML = Number(localStorage.getItem("cart-counter")) - 1;

    localStorage.setItem("cart-counter", Number(itemsInCart.innerHTML)); // Novi
broj artikala u korpi, nakon brisanja
    if (Number(itemsInCart.innerHTML) === 0) {
        renderCart(newState);
    }
    localStorage.setItem("cart", JSON.stringify(newState));
}
function renderCart(articles) {
    const productCart = document.querySelector("#product-cart");
    const formSubmit = document.querySelector("#form-submit");
    if (articles.products.length === 0) {
        formSubmit.classList.add("d-none");
        productCart.innerHTML = "The cart is empty!";
```

```
    } else {
        formSubmit.classList.remove("d-none");
        let res = [];
        let quantityArr = [];
        let html = "";
        res = allCameras.filter((camera) =>
            articles?.products.find((element) => {
                if (camera.id === element.id) {
                    return quantityArr.push(element.quantity);
                }
            })
        );
        res.forEach((item, index) => (html += makeCartItem({ ...item, quantity:
quantityArr[index] })));
        productCart.innerHTML = html;
    }
}

function setItemToLS(keyName, item) {
    return localStorage.setItem(keyName, JSON.stringify(item));
}
function getItemFromLS(keyName) {
    return JSON.parse(localStorage.getItem(keyName));
}

function clearItemFromLS(keyName) {
    return localStorage.removeItem(keyName);
}

function makeCartItem(item) {
    return `
    <div class='cart-item'>
        <div class="col-4 p-3">
            <img src="assets/images/products/${item.img.src}"
alt="${item.img.alt}" class="img-fluid" />
        </div>
        <div class="col-8">
            <div class="d-flex">
                <p class="me-3">Model:</p>
                <p>${item.name}</p>
            </div>
            <div class="d-flex">
                <p class="me-3">Price:</p>
                <p class="total me-3">${item.price.current * item.quantity}</p>
```

```
                <input type="hidden" class="price-per-product"
value="${item.price.current}">
            </div>
            <div class="d-flex">
                <p class="me-3">Quantity:</p>
                <input type="number" id="quantity" class="input-quantity w-25
form-control" value="${item.quantity}" data-id=${item.id} />
            </div>
            <button class="btn btn-primary btn-delete" data-
id=${item.id}>Delete</button>
        </div>
    </div>`;
}
function calculateTotalCash() {
    const total = document.querySelectorAll(".total");
    const sumId = document.querySelector("#sum");

    let sum = 0;

    total.forEach((price) => (sum += Number(price.innerText)));

    sumId.innerHTML = sum;
}
function submit() {
    const formContainer = document.querySelector("#form-container");
    const isValidForm = checkForm();

    if (cartCount == 0 || !isValidForm) {
        formContainer.classList.add("error-shake");
        setTimeout(() => formContainer.classList.remove("error-shake"), 1000);
    } else {
        displayInfoCard("You have make successfullt order.");

        clearItemFromLS("cart");
        clearItemFromLS("cart-counter");

        setTimeout(() => (window.location = "/index.html"), 2000);
    }
}
function checkForm() {
    let isValid = false;
    let errorCounter = 0;
    let isValidField;

    isValidField = checkField(nameId, regExpName, errName);
```

```
    if (!isValidField) errorCounter++;

    isValidField = checkField(lastName, regExpName, errLastName);
    if (!isValidField) errorCounter++;

    isValidField = checkField(email, regExpMail, errEmail);
    if (!isValidField) errorCounter++;

    isValidField = checkField(address, regExpAddr, errAddress);
    if (!isValidField) errorCounter++;

    if (errorCounter === 0) {
        isValid = true;
    }
    return isValid;
}

function checkField(inputName, regularExpression, errMsg) {
    let isValid = false;
    if (inputName.value === "") {
        inputName.style.border = "1px solid #ff0000";
        errMsg.innerHTML = "Name field can't be empty.";
    } else {
        if (!regularExpression.test(inputName.value)) {
            inputName.style.border = "1px solid #ff0000";
            errMsg.innerHTML = "Name is not in valid format. Name must start with
first capital letter and to have minimum 3 characters and maximum 14
characters.";
        } else {
            inputName.style.border = "1px solid #ced4da";
            errMsg.innerHTML = "";
            isValid = true;
        }
    }
    return isValid;
}

function displayInfoCard(message) {
    const infoCard = document.createElement("div");
    const cardContent = document.createTextNode(message);

    infoCard.setAttribute("class", "alert alert-warning info");

    infoCard.appendChild(cardContent);
    infoModal.appendChild(infoCard);}
```