ChatGPT

“E-Commerce Platforma” Web Loyihasi – Backend Dasturchi uchun Texnik Topshiriq (TZ)

Kirish

Ushbu hujjat **“E-Commerce Platforma”** nomli e-commerce (onlayn do‘kon) veb-loyihasining backend qismi uchun texnik topshiriqni bayon etadi. Texnik topshiriq Figma dizayniga asoslangan bo‘lib, aynan backend dasturchiga mo‘ljallangan. Loyiha Django ramkasi va Django REST Framework yordamida amalga oshiriladi. Hujjatda loyiha funksional talablari, ma’lumotlar bazasi modeli, API endpointlar, hamda har bir bo‘lim bo‘yicha aniq izohlar keltiriladi. Barcha ma’lumotlar O‘zbek tilida taqdim etiladi va hujjat .DOCX formatida shakllantirishga tayyor tarzda tuzilgan.

Loyiha Talablarining Umumiy Ko‘rinishi

Quyida **“E-Commerce Platforma”** loyihasining asosiy funksional talablari keltirilgan. Ushbu talablar foydalanuvchi interfeysi (Figma dizayn) va biznes mantiq asosida belgilanadi:

- **Erkaklar/Ayollar bo‘limini tanlash:** Bosh sahifada foydalanuvchi jins bo‘limini tanlaydi – **Erkaklar bo‘limi** (rus. *Мужской*) yoki **Ayollar bo‘limi** (rus. *Женский*). Tanlovga qarab sayt tegishli mahsulot kategoriyalarini ko‘rsatadi.
- **Asosiy kategoriyalar:** Har bir jins bo‘limida uchta asosiy kategoriya mavjud: **Oyoq kiyim (Обувь)**, **Kiyim (Одежда)** va **Aksessuarlar (Аксессуары)**. Ushbu kategoriyalar foydalanuvchiga mahsulotlarni turkumlashda yordam beradi.
- **Kichik kategoriyalar:** Har bir asosiy kategoriya ichida qo‘shimcha kichik kategoriyalar (sub-kategoriyalar) bo‘ladi. Masalan, **Oyoq kiyim (Обувь)** kategoriyasi tarkibida quyidagi sub-kategoriyalar mavjud:
- *“Кроссовки и кеды”* – krossovkalar va kedlar (sport poyabzallari)
- *“Обувь для спорта”* – sport uchun oyoq kiyimlar
- *“Ботинки”* – botinkalar (yarim etiklar)
- *“Тапки”* – shippaklar (uy oyoq kiyimlari)
- *“Сапоги”* – etiklar (baland botinkalar)
  Boshqa asosiy kategoriyalar (Kiyim, Aksessuarlar) ham xuddi shunday o‘z ichki toifalariga ega (masalan, kiyim uchun – futbolkalar, ko‘ylaklar, shimlar va hokazo; aksessuarlar uchun – sumkalar, taqinchoqlar, kamarlar va boshqalar).
- **Mahsulot ma’lumotlari:** Har bir mahsulot (tovar) uchun quyidagi atributlar saqlanadi va ko‘rsatiladi:
- **Mahsulot nomi:** Mahsulotning to‘liq nomlanishi (masalan, *“Erkaklar sport krossovkasi Nike Air Zoom”*).
- **Mahsulot rasmlari:** Mahsulotga oid bir nechta fotosuratlar. Har bir mahsulot kamida bitta asosiy rasmga ega bo‘ladi, va qo‘shimcha boshqa rakurslardan olingan rasmlar bo‘lishi mumkin.
- **Narx:** Mahsulotning amaldagi narxi. Valyuta (so‘m yoki boshqa) ham ko‘rsatilishi mumkin.
- **Rang:** Mahsulotning rangi (masalan, qora, oq, qizil). Har bir mahsulot biror bitta rang variantida keltiriladi; agar mahsulotning bir nechta rang variantlari bo‘lsa, ular alohida mahsulot sifatida yoki variant sifatida ko‘rib chiqilishi lozim.

- **O‘lchamlar (Razmerlar):** Mahsulot uchun mavjud o‘lchamlar ro‘yxati. Masalan, poyabzal uchun razmerlar: 40, 41, 42, ...; kiyim uchun o‘lchamlar: S, M, L, XL va hokazo. Mahsulot sahifasida foydalanuvchi ushbu o‘lchamlardan birini tanlay oladi.
- **Mijoz fikrlari (Отзывы клиентов):** Mahsulotga boshqa mijozlar qoldirgan sharhlar va baholar bo‘limi. Bu bo‘limda foydalanuvchilar mahsulot haqida o‘z fikr-mulohazalarini (review) ko‘rishlari mumkin.
- **Yetkazib berish va qaytarish (Доставка и возврат):** Har bir mahsulot sahifasida yetkazib berish va tovarni qaytarish siyosatiga oid ma’lumot beruvchi bo‘lim mavjud. Unda, masalan, **“Mahsulot yetkazib berish muddati 2-3 kun, 30 kun ichida bepul qaytarish imkoni”** kabi ma’lumotlar keltiriladi.
- **Sharh qoldirish formasi:** Mahsulot sahifasida foydalanuvchilar uchun sharh (fikr) qoldirish formasi taqdim etiladi. U quyidagi maydonlarni o‘z ichiga oladi:
- **Ism** – foydalanuvchining ismi (majburiy maydon).
- **Elektron pochta** – foydalanuvchining email manzili (majburiy maydon, email formatida bo‘lishi kerak).
- **Reyting** – mahsulotga qo‘yilayotgan yulduzcha baho (odatda 1 dan 5 gacha bo‘lgan reyting, majburiy maydon).
- **Fikr sarlavhasi** – sharh uchun qisqa sarlavha, maksimal 10 ta so‘z bilan (ixtiyoriy ravishda belgilangan cheklov, majburiy maydon).
- **Fikr matni** – sharhning batafsil matni, maksimal 150 ta so‘z bilan (majburiy maydon).
- Ushbu forma orqali foydalanuvchi o‘z fikrini yuborar ekan, har bir maydon bo‘yicha validatsiya tekshiruvi amalga oshiriladi. **Har bir maydon uchun xatolik xabarlari** mavjud bo‘lib, foydalanuvchi noto‘g‘ri yoki to‘ldirmagan taqdirda tegishli ogohlantirish ko‘rsatiladi.
- **Admin panel orqali boshqaruv:** Mahsulotlar katalogi va ularga bog‘liq ma’lumotlar ichki tizim orqali boshqariladi. Barcha **mahsulotlar, kategoriyalar, rasmlar va mijoz fikrlari** maxsus administrator paneli orqali kiritiladi va tahrirlanadi. Oddiy foydalanuvchilar yangi mahsulot qo‘shish yoki kategoriyalarni o‘zgartirish huquqiga ega emas; ular faqat mavjud ma’lumotlarni ko‘ra oladi va sharh qoldirishi mumkin. Admin paneli sifatida Django dasturining standarti **Django Admin** interfeysi foydalaniladi.
- **Backend texnologiyasi:** Loyiha server qismi **Python Django** veb-freyMVorki va **Django REST Framework** yordamida amalga oshiriladi. Django ma’lumotlar bazasi bilan ishlash, autentifikatsiya va admin panelni ta’minlaydi; Django REST Framework esa front-end uchun REST API endpointlar yaratish uchun qo‘llaniladi. Barcha funksionallik REST API orqali ham taqdim etiladi, bu kelajakda mobil ilova yoki boshqa front-end platformalar ulanashi uchun imkon yaratadi.
- **API endpointlar:** Backend tomonidan yaratiladigan API endpointlar orqali **kategoriyalar**, **mahsulotlar** va **mijoz fikrlari (sharhlar)** bilan ishlash imkoni bo‘ladi. Front-end qismi ushbu endpointlarga murojaat qilib, kategoriya va mahsulotlar ro‘yxatini olish, mahsulot detallari (rasmlar, atributlar, sharhlar)ni olish va yangi sharh yuborish funksiyalarini bajaradi. Quyida ushbu API lar batafsilroq tavsiflanadi.

Yuqoridagi talablar asosida backend arxitekturasi va funksional imkoniyatlar quyidagi bo‘limlarda batafsil yoritiladi.

Katalog Bo‘limlari va Kategoriyalar

Erkaklar va Ayollar Bo‘limlari

Loyihada mahsulotlar **Erkaklar** va **Ayollar** bo‘limlariga ajratilgan. Bosh sahifa (kirish sahifasi)da foydalanuvchi dastlab ushbu ikkita bo‘limdan birini tanlaydi. Tanlangan bo‘limga qarab, foydalanuvchiga tegishli mahsulot kategoriyalari va mahsulotlar ko‘rsatiladi.

- **Erkaklar bo‘limi (Мужской):** Erkaklar uchun mo‘ljallangan mahsulotlar katalogini o‘z ichiga oladi (masalan, erkaklar oyoq kiyimlari, erkaklar kiyimlari va aksessuarlari).
- **Ayollar bo‘limi (Женский):** Ayollar uchun mo‘ljallangan mahsulotlar katalogini o‘z ichiga oladi (masalan, ayollar oyoq kiyimlari, ayollar kiyimlari va aksessuarlari).

Ushbu bo‘limlar strukturaviy jihatdan saytning eng yuqori kategoriya darajasini ifodalaydi. Ma’lumotlar bazasi nuqtai nazaridan, biz buni **asosiy kategoriya** sifatida ko‘rib chiqamiz. Ya’ni, **“Erkaklar”** va **“Ayollar”** – bosh kategoriyalar bo‘lib, ularning ichida keyingi kategoriyalar joylashadi.

Backend talablari: Kategoriyalar tuzilmasi shunday tashkil etilishini ta’minlash kerakki, har bir mahsulot qaysi bo‘lim (erkaklar yoki ayollar)ga tegishli ekanini aniqlash imkoni bo‘lsin. Buning uchun: - **Kategoriya modeli**da **ota-ona kategoriya** (parent) tushunchasi joriy etiladi. Masalan, *Erkaklar* kategoriyasi hech qanday parent’ga ega emas (yuqori daraja), *Oyoq kiyim* kategoriyasining parent’i *Erkaklar* bo‘ladi. - Yoki alternativ tarzda, mahsulot modelida **gender/jins** maydoni kiritilishi va erkak/ayol bo‘limi bo‘yicha belgilanishi mumkin. Ushbu loyiha doirasida hierarchical (daraxt) ko‘rinishdagi kategoriya modeli tanlanadi, chunki bu yangi bo‘lim/kategoriyalar qo‘shishda moslashuvchanlik beradi.

Asosiy Kategoriyalar

Har bir bo‘lim ichida uchta asosiy kategoriya mavjud: 1. **Oyoq kiyim (Обувь)** 2. **Kiyim (Одежда)** 3. **Aksessuarlar (Аксессуары)**

Bu kategoriyalar sayt menyusi va navigatsiyasining asosini tashkil etadi. Foydalanuvchi Erkaklar yoki Ayollar bo‘limini tanlaganidan so‘ng, ushbu asosiy kategoriyalarni ko‘radi va keraklisini tanlab, tegishli mahsulotlar ro‘yxatiga o‘tishi mumkin.

Backend talablari: - Kategoriyalar bo‘limga bog‘langan holda saqlanadi. Misol uchun, **Erkaklar** bo‘limining **Oyoq kiyim** kategoriyasi va **Ayollar** bo‘limining **Oyoq kiyim** kategoriyasi – bu nomlari bir xil bo‘lsa-da, ikki xil kategoriya sifatida bazada saqlanishi mumkin (ota-ona bo‘limlari turlicha). - Har bir asosiy kategoriya boshqa bir parent kategoriyaga ega emas, balki ular parent sifatida Erkaklar yoki Ayollar bo‘limiga ega. Masalan, *Erkaklar/Oyoq kiyim* kategoriyasi: bu yerda *Erkaklar* – parent kategoriya, *Oyoq kiyim* – shu parent ostidagi kategoriya.

Kichik Kategoriyalar (Sub-kategoriyalar)

Har bir asosiy kategoriya o‘z navbatida bir nechta kichik (ichki) kategoriyalarga bo‘linadi. Bu kichik kategoriyalar foydalanuvchilarga mahsulotlarni yanada aniqroq filtr va turkumlash imkonini beradi.

Misol tariqasida, **Oyoq kiyim** asosiy kategoriyasi quyidagi sub-kategoriyalarga ajratilgan edi: - *“Кроссовки и кеды”* – Sport uslubidagi krossovkalar va kedlar. - *“Обувь для спорта”* – Maxsus sport uchun mo‘ljallangan oyoq kiyimlar (masalan, futbol butsalari, yugurish uchun maxsus oyoq kiyimlar). - *“Ботинки”* – Botinkalar (klassik uslubdagi yoki kuz-qish uchun mo‘ljallangan oyoq kiyimlar). - *“Тапки”* –

Uy uchun shippak va ichki xonada kiyiladigan yengil oyoq kiyimlar. - *“Canoэu”* – Etiklar (baland qoziqli, odatda ayollar yoki qishki etiklar).

Backend talablari: - Ma’lumotlar bazasida kichik kategoriyalar ham yagona **Kategoriya modeli** orqali boshqariladi. Har bir kichik kategoriya tegishli **asosiy kategoriya**ni parent sifatida ko‘rsatadi. Masalan, *“Krossovkalar va kedlar”* kategoriyasining parent’i *Oyoq kiyim* (erkaklar yoki ayollar bo‘limi uchun alohida). - Kategoriya ierarxiyasi cheksiz chuqurlikka ega bo‘lishi mumkin, lekin ushbu loyiha doirasida maksimal 3 daraja (bo‘lim -> asosiy kategoriya -> kichik kategoriya) yetarli bo‘ladi. - Kategoriyalarni qo‘shish/o‘zgartirish admin panel orqali amalga oshadi. Admin yangi asosiy kategoriya yoki kichik kategoriya qo‘shganda, uning qaysi parent ostida ekanini tanlay olishi kerak.

Front-endga ta’siri: Kategoriya iyerarxiyasi REST API orqali ham taqdim etiladi. Masalan, front-end `/api/categories/` endpointiga so‘rov yuborib, barcha kategoriyalarni daraxt ko‘rinishida yoki kamida parent-child munosabatlarini ko‘rsatadigan tarzda olishi mumkin. Bu foydalanuvchi Erkaklar/Ayollar bo‘limini tanlaganda, tegishli kategoriyalar ro‘yxatini tuzish uchun ishlatiladi.

Mahsulotlar va Ularning Xususiyatlari

Mahsulot ma’lumotlari va atributlari

E-commerce platformaning asosiy qismi – bu mahsulotlar katalogi. Har bir **Mahsulot** quyidagi muhim ma’lumotlarga ega bo‘ladi:

- **Nomi:** Mahsulotning nomi aniq va mazmunli bo‘lishi kerak. Bu maydon matn (CharField) sifatida saqlanadi (taxminan 200 belgigacha chegaralash tavsiya etiladi). Misol: *“Ayollar qishki etiklari – qizil rang”*.
- **Tavsif (description):** *(* Dizaynda alohida ko‘rsatilmagan bo‘lsa ham, platformani foydali qilish maqsadida mahsulot tavsifi maydoni ham bo‘lishi maqsadga muvofiq.*) Mahsulotning batafsil tavsifi, xususiyatlari, materiallari va boshqalar. Bu matnli (TextField) maydon bo‘lishi mumkin. Agar Figma dizaynida tavsif bo‘limi nazarda tutilgan bo‘lsa, admin panel orqali har bir mahsulot uchun tavsif kiritish imkoni bo‘ladi.
- **Narx:** Mahsulot narxi son ko‘rinishida saqlanadi. Django modelida DecimalField orqali, masalan, `max_digits=10, decimal_places=2` bilan (so‘m qiymatlarini ham ifodalay olishi uchun). Narx front-endda valyuta belgisi bilan ko‘rsatiladi (UZS so‘m, $ yoki € kabi).
- **Valyuta:** (Agar kerak bo‘lsa) Narx qaysi valyutada ekanini belgilovchi maydon bo‘lishi mumkin (masalan, so‘m, dollar). Aksar hollarda bir xil valyuta ishlatilsa, bu maydon talab qilinmasligi mumkin va statik yozuv (masalan, "so‘m") sifatida ko‘rsatiladi.
- **Rang:** Mahsulotning rangi nomi (matn sifatida, masalan, *“Qora”*, *“Oq”*, *“Ko‘k”* va hokazo). Bu uchun CharField ishlatiladi. Agar mahsulot bir nechta rang variantiga ega bo‘lsa, ularni alohida mahsulot sifatida kiritish yoki variantlar jadvali yaratish mumkin. Loyiha soddaligi uchun hozircha har bir mahsulot bitta rang atributiga ega deb olinadi.
- **O‘lchamlar (razmerlar):** Mahsulot uchun mavjud bo‘lgan o‘lchamlar ro‘yxati saqlanishi lozim. Buning yechimi bir nechta usulda bo‘lishi mumkin:
- Agar har bir mahsulot uchun bir nechta o‘lcham tanlanadigan bo‘lsa, mahsulot modeli uchun alohida *ManyToMany* munosabatda **O‘lchamlar** jadvali yaratish mumkin (masalan, `Size` modeli: nomi = "40", "41", "S", "M" va h.k.).
- Yoki soddaroq yondashuv – mahsulot modelida matn maydonida mavjud o‘lchamlarni vergul bilan ajratilgan ro‘yxat shaklida saqlash (masalan, `"40, 41, 42, 43"` yoki `"S, M, L, XL"`). Bu usul kamroq ma’qul, lekin kichik loyihalarda qo‘llanilishi mumkin.

- Eng to‘g‘ri yondashuv: **ProductVariant** konsepsiyasini joriy etish – mahsulotning har bir rang/o‘lcham kombinatsiyasi uchun alohida variant obyektlari saqlash. Lekin ushbu TZ doirasida variantlar modeli joriy etilmasdan, mahsulotga to‘g‘ridan-to‘g‘ri o‘lchamlar ro‘yxatini bog‘lash yetarli deb hisoblaymiz.
- **Kategoriya:** Har bir mahsulot ma’lum bir kichik kategoriya (sub-kategoriya)ga tegishli bo‘ladi. Masalan, *“Nike Air Zoom”* sport poyabzali mahsuloti *“Erkaklar -> Oyoq kiyim -> Krossovkalar va kedlar”* kategoriyasiga mansub. Django modelida mahsulot obyekti `ForeignKey` orqali kategoriya modeliga ulanadi. Shu tariqa, mahsulotning qaysi bo‘lim va toifa ostida ekani ma’lum bo‘ladi. Kategoriya o‘chirilsa yoki o‘zgarsa, tegishli mahsulotlarda ham bu aks etadi (on_delete=PROTECT yoki CASCADE siyosati bilan).
- **Rasmlar:** Mahsulotga biriktirilgan bir nechta rasm fayllari. Birinchi rasm odatda asosiy (cover) rasm bo‘lib xizmat qiladi, qolgan rasmlar galereya sifatida ko‘rsatiladi. Django’da bir nechta rasmni saqlash uchun alohida model – masalan, **ProductImage** modeli yaratiladi va u har bir rasm faylini va qaysi mahsulotga tegishli ekanini saqlaydi. Shu tariqa, mahsulot va uning rasmlari o‘rtasida bir-to-many munosabat o‘rnatiladi (bir mahsulotda ko‘p rasm).
- **Yetkazib berish va qaytarish ma’lumotlari:** Har bir mahsulot sahifasida “Yetkazib berish va qaytarish” bo‘limi mavjud bo‘lib, u yerda foydalanuvchi mahsulotni yetkazib berish shartlari va qaytarish siyosatini o‘qishi mumkin. Bu ma’lumot ko‘pincha barcha mahsulotlar uchun bir xil bo‘ladi (masalan, do‘konning umumiy siyosati: *“Yetkazib berish 2-5 ish kuni, 30 kun ichida qaytarish imkoniyati”*). Uni amalga oshirish uchun bir necha variant:
- Statik matn sifatida front-end kodida qo‘shilishi (kam tavsiya etiladi, chunki admin tomonidan o‘zgartirish qiyin bo‘ladi).
- Bitta global konfiguratsiya sifatida saqlash (masalan, admin panelda bitta joyda “Yetkazib berish va qaytarish siyosati” matnini saqlash).
- **Har bir mahsulot uchun alohida maydon sifatida saqlash.** Ushbu yondashuv mahsulotdan mahsulotga farqli siyosat bo‘lish ehtimolini ham ko‘zda tutadi. Texnik topshiriq doirasida har bir mahsulot modelida `delivery_and_return_info` (TextField) maydoni qo‘shiladi. Admin panelda mahsulot qo‘shishda doimiy ravishda bir xil matn kiritishi mumkin, yoki lozim bo‘lsa, muayyan mahsulot uchun o‘ziga xos ma’lumot yozadi.
- **Mavjudlik (stock) va boshqalar:** (Dizaynda ko‘zda tutilmagan qo‘shimcha atributlar, masalan, omborda qancha dona mavjudligi, SKU kodi va hokazo, hozircha kiritilmaydi.)

Mahsulotlar bo‘yicha backend talablari: - Django admin panelida mahsulot qo‘shish/ozgartirish oynasida yuqoridagi barcha maydonlar (nomi, kategoriya, narx, rang, o‘lchamlar, tavsif, yetkazib berish/qaytarish ma’lumoti) mavjud bo‘lishi va to‘g‘ri ishlashi lozim. - Mahsulot o‘chirib yuborilsa, unga tegishli rasmlarni ham o‘chirish kerak (ma’lumotlar bazasida va fayl tizimida). - Mahsulot ma’lumotlarini olish uchun API endpointlar taqdim etiladi (quyida keltirilgan).

Mahsulotlar bo‘yicha misol JSON

(Izoh: Quyidagi misol front-endga yuboriladigan JSON ko‘rinishda mahsulot detallari qanday bo‘lishini tasavvur qilish uchun keltirilmoqda. Bu backend implementatsiyasini rejalashtirishda yordam beradi.)

```
{
  "id": 101,
  "name": "Nike Air Zoom Sport Krossovka",
  "category": "Erkaklar / Oyoq kiyim / Krossovkalar va kedlar",
  "price": 299000,
  "currency": "UZS",
  "color": "Qora",
  "sizes": ["40", "41", "42", "43"],
```

```
  "images": [
    "https://.../media/products/101_main.jpg",
    "https://.../media/products/101_1.jpg",
    "https://.../media/products/101_2.jpg"
  ],
  "description": "Yengil va qulay sport krossovka, original dizayn...",
  "delivery_and_return_info": "Yetkazib berish 2-5 ish kunida. 30 kun ichida
qaytarish mumkin.",
  "reviews": [
    {
      "id": 501,
      "name": "Ali",
      "rating": 5,
      "title": "Juda a'lo",
      "content": "Mahsulot sifatli va qulay, tavsiya qilaman.",
      "date": "2025-05-10"
    },
    {
      "id": 502,
      "name": "Oygul",
      "rating": 4,
      "title": "Yaxshi",
      "content": "Narxiga nisbatan juda yaxshi oyoq kiyim. O'lchami mos
keldi.",
      "date": "2025-05-12"
    }
  ]
}
```

Yuqoridagi misolda mahsulotning to‘liq ma’lumotlari, shu jumladan sharhlar ham jamlangan. Keyingi bo‘limlarda sharhlar tizimi va API haqida batafsil ma’lumot beriladi.

Mijoz Fikrlari (Sharhlar) Tizimi

Sharhlar va Reytinglar

Mijoz fikrlari (Отзывы клиентов) bo‘limi har bir mahsulot sahifasida ko‘rsatiladi va u mahsulotga avvalgi xaridorlar tomonidan qoldirilgan sharhlar va baholarni o‘z ichiga oladi. Bu foydalanuvchilarga mahsulot sifati va haqiqiy tajribalar haqida ma’lumot olishga yordam beradi.

Backend qismida sharhlar alohida **Sharh modeli** (Review) yordamida boshqariladi. Har bir sharh quyidagi ma’lumotlardan iborat: - **Mahsulot**: Qaysi mahsulotga tegishli sharh ekanini ko‘rsatadi. Django modelida bu `ForeignKey` orqali mahsulot modeliga ulanadi (one-to-many: bir mahsulotning ko‘plab sharhlari bo‘lishi mumkin). - **Ism**: Sharh qoldiruvchi shaxsning ismi (CharField, masalan max_length=100). - **Elektron pochta**: Foydalanuvchining email manzili (EmailField, max_length ~ 100-254). Bu maydon orqali foydalanuvchining shaxsi tasdiqlanmaydi (sayt user account talab qilmaydi), lekin haqiqiy email formatiga mos yozilgan bo‘lishi lozim. - **Reyting**: Raqamli baho – odatda 1 dan 5 gacha integer qiymat. Django modelida IntegerField orqali, masalan, choices=[(1,"1"),...,(5,"5")] bilan berilishi mumkin yoki hech bo‘lmaganda min=1, max=5 cheklovlari qo‘yiladi. - **Fikr sarlavhasi**: Sharh uchun qisqacha sarlavha yoki yozuvning qisqa mazmuni. Masalan, “Juda a’lo”, “Yomon emas” kabi. Bu

maydon ixtiyoriy ravishda 1-2 jumla yoki eng ko‘pi bilan 10 taga yaqin so‘zdan iborat bo‘lishi kerak. Modelda CharField (masalan, max_length=100) bilan saqlanadi. - **Fikr matni**: Sharhning to‘liq matni – foydalanuvchi mahsulot haqidagi fikrini batafsil yozadigan qism. Bu taxminan 150 tagacha so‘z bo‘lishi belgilangan. Modelda TextField sifatida saqlanadi. Kerak bo‘lsa, max_length bilan taxminiy chegara qo‘yish yoki serializer darajasida so‘zlar sonini tekshirish orqali cheklov kiritiladi. - **Yuborilgan sana**: Sharh qoldirilgan sana-vaqti (DateTimeField, auto_now_add=True). Bu maydon foydalanuvchiga ko‘rsatilishi mumkin (masalan, “2025-05-12 da qoldirildi” kabi) va sharhlarni vaqt bo‘yicha tartiblashda yordam beradi.

Sharhlarni ko‘rish: Foydalanuvchi mahsulot sahifasini ochganda, backend dan shu mahsulotga tegishli barcha tasdiqlangan sharhlar ro‘yxati (ism, reyting, matn va sana kabi ma’lumotlar bilan) JSON orqali jo‘natiladi. Front-end ushbu ro‘yxatni chiqarib beradi. Agar sharhlar soni ko‘p bo‘lsa, sahifalash (pagination) joriy etilishi mumkin (masalan, bir sahifada 10 tadan sharh).

Sharh qoldirish (Fikr formasi) va Validatsiya

Mahsulot sahifasida foydalanuvchi yangi sharh qoldirishi uchun forma mavjud. Forma tarkibidagi maydonlar va ularning cheklovlari: - **Ism** – majburiy. Bo‘sh qoldirib bo‘lmaydi. Agar foydalanuvchi kiritmasa, xatolik xabari ko‘rsatiladi (masalan: *"Iltimos, ismingizni kiriting."*). - **Elektron pochta** – majburiy. E-mail formatiga mos bo‘lishi kerak. Kiritilmasa yoki noto‘g‘ri formatda bo‘lsa, xatolik xabari ko‘rsatiladi (masalan: *"Elektron pochtangizni to‘g‘ri formatda kiriting."*). - **Reyting** – majburiy. 1 dan 5 gacha qiymat tanlanmagan bo‘lsa, xato xabari chiqadi (masalan: *"Iltimos, mahsulot reytingini tanlang."*). Reyting qiymati ruxsat etilgan oraliqda emas (masalan, 0 yoki 6) bo‘lsa, backend uni qabul qilmaydi. - **Fikr sarlavhasi** – majburiy. Agar foydalanuvchi sarlavha kiritmasa, xato xabar (*"Fikr uchun qisqa sarlavha yozing."*) chiqadi. Agar sarlavha 10 so‘zdan oshsa, xato: *"Fikr sarlavhasi 10 ta so‘zdan oshmasligi kerak."* Sarlavhaning maksimal uzunligi taxminan 100 belgi atrofida cheklanishi lozim (10 so‘zdan iborat o‘rtacha jumla sifatida). - **Fikr matni** – majburiy. Bo‘sh bo‘lsa, xato xabar: *"Iltimos, fikringizni bayon qiling."*. Agar matn 150 ta so‘zdan oshsa, xato xabar: *"Fikr matni 150 ta so‘zdan oshmasligi kerak."*. Bu chegara foydalanuvchilarning haddan tashqari uzun sharh yozishining oldini oladi va hamma uchun o‘qishli bo‘lishini ta’minlaydi.

Backenddagi amal va tekshiruvlar: - Django REST Framework yordamida sharhlar uchun **Serializer** yaratiladi, unda yuqoridagi cheklovlar validatsiya qoidalari sifatida yoziladi. Masalan, `EmailField` invalid format uchun avtomatik xato qaytaradi; `CharField(max_length=100)` uzunlikni cheklaydi; `validate()` metodida yoki o‘ziga xos validator funksiyalarda so‘zlar sonini sanab ko‘rib cheklash mumkin. - Har bir xato holati uchun DRF avtomatik ravishda yoki custom tarzda JSON shaklida xatolik javobini qaytaradi. Front-end esa bu javobga asoslanib foydalanuvchiga tegishli xatolik xabarini ko‘rsatadi. - Sharh muvaffaqiyatli qabul qilingandan so‘ng, u **ma’lumotlar bazasida saqlanadi** va darhol tegishli mahsulot sharhlari ro‘yxatiga qo‘shiladi. Agar moderatsiya talab etilsa (TZda bunaqasi aytilmagan), sharh avval admin tomonidan tasdiqlanib keyin ko‘rinishi ham mumkin. Hozircha har bir yuborilgan sharh avtomatik tarzda ko‘rinadi deb faraz qilamiz. - Django admin panelida ham sharhlar bo‘limi bo‘ladi – admin barcha sharhlarni ko‘ra oladi, zarur bo‘lsa nojo‘ya sharhlarni o‘chirib tashlashi yoki tahrirlashi mumkin.

Django Model Tuzilmalari

Quyida loyiha uchun mo‘ljallangan ma’lumotlar bazasi modellarining Django kod ko‘rinishidagi tuzilmalari keltiriladi. Model tuzilmalari loyiha talablari va biznes qoidalarini aks ettiradi.

Kategoriya modeli

```
from django.db import models

class Category(models.Model):
    name = models.CharField(max_length=100)
    parent = models.ForeignKey('self', null=True, blank=True,
on_delete=models.CASCADE, related_name='children')

    class Meta:
        verbose_name = "Kategoriya"
        verbose_name_plural = "Kategoriyalar"

    def __str__(self):
        return f"{self.parent.name + ' -> ' if self.parent else ''}
{self.name}"
```

Izoh: - `Category` modeli **rekursiv (o‘z-o‘ziga ishora qiluvchi)** tuzilishga ega. `parent` maydoni orqali har bir kategoriya boshqa bir kategoriyani ota-ona sifatida belgilashi mumkin. Agar `parent=None` bo‘lsa, bu kategoriya eng yuqori darajada (masalan, *Erkaklar* bo‘limi yoki *Ayollar* bo‘limi). - `related_name='children'` – bu kategoriya ostidagi to‘g‘ridan-to‘g‘ri ichki kategoriyalarni chaqirish uchun ishlatiladi (masalan, `erkaklar_category.children.all()` Erkaklar bo‘limining barcha asosiy kategoriyalarini qaytaradi). - `on_delete=models.CASCADE` – agar ota kategoriya o‘chirib yuborilsa, unga bog‘liq barcha ichki kategoriyalarni ham o‘chiradi. **Diqqat:** Bu biznes qoidalari nuqtai nazaridan ehtiyotkorlik bilan qo‘llanishi kerak, chunki masalan, *Erkaklar* bo‘limini o‘chirish butun bir bo‘limdagi barcha kategoriyalarni va mahsulotlarni yo‘q qilishi mumkin. Real loyihada muhim kategoriyalarni o‘chirib qo‘ymaslik uchun `PROTECT` yoki hech bo‘lmasa admin panelda cheklov qo‘yish mumkin. Bu yerda soddalik uchun CASCADE ko‘rsatilgan. - Admin panelda kategoriyalar daraxt ko‘rinishida (parent-child) aks etadi. Kategoriya nomi noyob bo‘lishi shart emas (masalan, Erkaklar/Oyoq kiyim va Ayollar/Oyoq kiyim – nomi bir xil, parenti boshqa), lekin kombinatsiyasi noyob bo‘lishi maqsadga muvofiq.

Mahsulot modeli

```
class Product(models.Model):
    name = models.CharField("Nomi", max_length=200)
    category = models.ForeignKey(Category, on_delete=models.PROTECT,
related_name="products")
    price = models.DecimalField("Narx", max_digits=10, decimal_places=2)
    currency = models.CharField("Valyuta", max_length=10, default="UZS")
    color = models.CharField("Rang", max_length=50)
    sizes = models.CharField("O'lchamlar", max_length=100, help_text="Mavjud
o'lchamlar ro'yxati, masalan: 40,41,42")
    description = models.TextField("Tavsif", null=True, blank=True)
    delivery_and_return_info = models.TextField("Yetkazib berish va
Qaytarish", null=True, blank=True)

    class Meta:
        verbose_name = "Mahsulot"
```

```
        verbose_name_plural = "Mahsulotlar"
        ordering = ["name"]

    def __str__(self):
        return self.name
```

Izoh: - `name`: Mahsulot nomi, 200 belgigacha matn. Admin panelda `verbose_name` orqali “Nomi” deb ko‘rsatiladi. - `category`: Mahsulotning kategoriyasi. *ForeignKey* `Category` modeliga ulanadi. `on_delete=models.PROTECT` qilingan – bu kategoriya o‘chirib yuborilganda mahsulotlarning saqlanishini kafolatlaydi (kategoriya o‘chirilmasdan oldin admin avval mahsulotlarni boshqa kategoriyaga o‘tkazishi lozim bo‘ladi). `related_name="products"` – shu kategoriya ichidagi barcha mahsulotlarni olish imkonini beradi (masalan, `some_category.products.all()`). - `price`: Mahsulot narxi. DecimalField 10 raqamgacha (ikkita kasr qismi bilan). Bu 999,999,99 so‘mgacha yoki 99999.99 $ gacha qiymatlarni saqlashga yetadi. - `currency`: Valyuta belgisi yoki nomi (UZS, USD, EUR va hokazo). Standart bo‘lib "UZS" (so‘m) qabul qilingan. - `color`: Mahsulot rangi, masalan, “Qora”, “Ko‘k”. Max 50 belgi. - `sizes`: Mavjud o‘lchamlar ro‘yxati. Bu yerda soddalik uchun CharField sifatida vergul bilan ajratilgan ro‘yxat matnini saqlaymiz (masalan, `“40, 41, 42”` yoki `“S, M, L”`). **Eslatma:** Katta loyiha uchun bu usul optimal emas, ammo ayni talablar doirasida soddalik uchun tanlandi. Kelajakda alohida `Size` modeli va ManyToMany munosabat orqali yaxshilash mumkin. - `description`: Mahsulot tavsifi. Bo‘lishi shart emas (`blank=True`), lekin foydali ma’lumot kiritish uchun maydon bor. - `delivery_and_return_info`: Yetkazib berish va qaytarish bo‘yicha ma’lumot matni. Bu ham ixtiyoriy; agar to‘ldirilmasa, front-end global standart matnni ko‘rsatishi yoki bu bo‘limni yashirishi mumkin. Biroq admin har bir mahsulotga xos ma’lumot kiritish imkoniyatiga ega bo‘ladi. - `ordering = ["name"]`: Mahsulotlar default bo‘yicha nomiga qarab saralanadi (admin panel va ba’zi API chiqimlarida). Zaruratga ko‘ra boshqacha tartiblash ham o‘rnatilishi mumkin (masalan, qo‘shilgan sana bo‘yicha).

Mahsulot Rasmlari modeli

```
class ProductImage(models.Model):
    product = models.ForeignKey(Product, on_delete=models.CASCADE,
related_name="images")
    image = models.ImageField("Rasm fayli", upload_to="products/")
    alt_text = models.CharField("Rasm tavsifi", max_length=200, null=True,
blank=True)
    order = models.PositiveIntegerField("Tartib", default=0)

    class Meta:
        verbose_name = "Mahsulot rasmi"
        verbose_name_plural = "Mahsulot rasmlari"
        ordering = ["order"]

    def __str__(self):
        return f"Image for {self.product.name} ({self.id})"
```

Izoh: - `product`: Ushbu rasm tegishli bo‘lgan **Mahsulot**. *ForeignKey* bilan `Product` modeliga ulanadi. `on_delete=models.CASCADE` – mahsulot o‘chirilsa, barcha bog‘liq rasmlar ham o‘chadi (mantiqan to‘g‘ri qaror, chunki o‘sha mahsulot bo‘lmasa, uning rasmlari ham keraksiz). - `image`: Haqiqiy rasm faylini saqlovchi maydon. `ImageField` ishlatiladi, `upload_to="products/"` parametri bilan

rasmlar `MEDIA_ROOT/products/` papkasiga yuklanadi. Admin panel orqali yuklash mumkin bo‘ladi. - `alt_text`: Rasmga qisqacha tavsif yoki alt matn (masalan, “Nike Air Zoom krossovka old ko‘rinish”). Bu foydalanuvchilarga rasm chiqmasa yoki ekran o‘qiydiganlarda foydali bo‘ladi. Ixtiyoriy maydon. - `order`: Agar mahsulotning bir nechta rasmi bo‘lsa, ularni tartib bilan joylashtirish uchun raqam. Kichkina butun son (0,1,2,...) bilan belgilash mumkin. Admin panelda inline rasm qo‘shganda bu maydon orqali tartibni nazorat qilish mumkin. Front-end eng past `order` qiymatli rasmdan boshlab ko‘rsatadi (0 yoki 1 – asosiy rasm). - Meta `ordering=["order"]` – `product.images.all()` qilganda tartib bo‘yicha chiqishini ta’minlaydi.

Sharh (Fikr) modeli

```
class Review(models.Model):
    product = models.ForeignKey(Product, on_delete=models.CASCADE,
related_name="reviews")
    name = models.CharField("Ism", max_length=100)
    email = models.EmailField("Email")
    rating = models.PositiveSmallIntegerField("Reyting", choices=[(i, str(i))
for i in range(1,6)])
    title = models.CharField("Sarlavha", max_length=100)
    content = models.TextField("Matn")
    created_at = models.DateTimeField("Yuborilgan sana", auto_now_add=True)

    class Meta:
        verbose_name = "Sharh"
        verbose_name_plural = "Sharhlar"
        ordering = ["-created_at"]

    def __str__(self):
        return f"{self.product.name} - {self.name} ({self.rating}⭐)"
```

Izoh: - `product`: Sharh qaysi mahsulotga tegishli – *ForeignKey* orqali `Product`ga ulanadi. `on_delete=models.CASCADE` – mahsulot o‘chirilsa, uning barcha sharhlari ham o‘chadi (ma’qul qaror, chunki mahsulot bo‘lmasa, sharhlar ahamiyatsiz bo‘ladi). - `name`: Fikr qoldiruvchi foydalanuvchi ismi. 100 belgigacha matn. - `email`: Foydalanuvchi elektron pochtasi. Email formatini avtomatik tekshiradi. - `rating`: Reyting bahosi. `PositiveSmallIntegerField` (0 dan 32767 gacha) lekin `choices` orqali 1-5 oralig‘ida chekladik. Bu Django admin va serializerda faqat 1,2,3,4,5 ni tanlashga majbur qiladi. (Agar foydalanuvchi API orqali `6` yuborsa ham, serializer/validation darajasida rad qilinadi). - `title`: Sharh sarlavhasi, max_length=100. Shu tariqa 10 so‘z chegarasi taxminiy tarzda nazoratga olinadi (100 belgidan oshmasa, odatda 10-12 so‘zdan oshmaydi). Zarurat bo‘lsa, qo‘shimcha validatsiya bilan so‘zlar sonini tekshirish mumkin. - `content`: Sharh matni (TextField). Foydalanuvchi cheklovsiz yozishi mumkin, lekin odob-axloq va maqsadga muvofiqlik nuqtai nazaridan 150 so‘zdan oshmasligi lozim. Bu ham alohida validatsiya qoidasi sifatida qo‘shilishi mumkin (serializer yoki model `clean()` metodida). - `created_at`: Sharh yaratilgan vaqtni avtomatik belgilaydi. `ordering = ["-created_at"]` orqali admin panel va API’da standarte bo‘yicha eng yangi sharhlar birinchi ko‘rsatiladi (ya’ni teskari xronologik tartib).

Admin panel integratsiyasi: - Barcha modellar (Category, Product, ProductImage, Review) Django admin saytiga **register** qilinadi. `ProductImage` odatda `Product` modeliga *inline* tarzda qo‘shiladi, bu mahsulotni tahrirlash paytida biryo‘la rasmlarini ham qo‘shib boshqarish uchun qulay (StackedInline

yoki TabularInline). - `Category` modeli admin interfeysda **daraxt ko‘rinishi**da tartibda chiqishi uchun, masalan, django-mptt kabi paketlarsiz, oddiy ro‘yxatda parent ustida filtrlash yoki tartiblash bilan cheklanamiz. Kategoriyalarni tartibli ko‘rish uchun admin’da `list_display` va `list_filter = ['parent']` kabi sozlamalar berilishi mumkin. - `Review` modeli adminda `list_display`da mahsulot nomi, foydalanuvchi ismi, reyting va qisqa sarlavha ko‘rinishida chiqariladi. Bu moderatsiya uchun qulaylik yaratadi. Zarurat bo‘lsa, `Review` modeliga `is_approved` (tasdiqlangan) maydoni qo‘shib, dastlabki holatda `False` qilib, admin tasdiqlasa `True`ga o‘tkazish mumkin. Hozirgi talablar buni nazarda tutmagani uchun kiritilmayapti.

REST API Endpointlari

Loyihaning backend qismi REST API orqali front-endga ma’lumot uzatadi va qabul qiladi. Quyida **Django REST Framework** asosida rejalashtirilgan API endpointlar va ularning vazifalari keltiriladi. Barcha API lar bazaviy URL prefiksi sifatida, masalan, `/api/` ostida taqdim etiladi.

Eslatma: API lar JWT token orqali autentifikatsiya talab qilmaydi (foydalanuvchilar uchun ochiq), chunki sayt mehmon rejimida ishlaydi va sharh qoldirish uchun ham faqat email kiritish kifoya. Biroq adminlar uchun alohida himoyalangan endpointlar bo‘lishi mumkin (yoki admin panelidan foydalanish tavsiya etiladi). Quyida keltirilgan endpointlar autentifikatsiyasiz ochiq API sifatida ko‘zda tutiladi.

Kategoriya va bo‘limlar bo‘yicha API

- **GET** `/api/categories/` – **Barcha kategoriyalar ro‘yxati.**
  Ushbu endpoint orqali tizimdagi barcha kategoriyalarni olish mumkin. Natija daraxt shaklida yoki tekis ro‘yxat ko‘rinishida qaytarilishi mumkin. Masalan, tekis ro‘yxatda har bir kategoriya obyektida `id`, `name`, va `parent_id` maydonlari bo‘ladi. Daraxt shakli uchun esa ichki `children` ro‘yxatlari bilan JSON tuzilishi berilishi mumkin.
  Foydalanish: Front-end dastlab barcha kategoriyalarni olib, keyinchalik qidiruv yoki navigatsiya uchun ishlatishi mumkin. Yoki boshqa usul – alohida bo‘limlar bo‘yicha olish.

- **GET** `/api/categories/?parent={id}` – **Muayyan parent ostidagi kategoriyalar.**
  Masalan, `GET /api/categories/?parent=null` (yoki parent filtiri bo‘lmasligi) – yuqori darajadagi (parenti yo‘q) kategoriyalarni qaytaradi (ya’ni Erkaklar va Ayollar bo‘limlari). `GET /api/categories/?parent=1` – `id=1` kategoriya ostidagi ichki kategoriyalarni qaytaradi (masalan, Erkaklar bo‘limining ichidagi Oyoq kiyim, Kiyim, Aksessuarlar). Shu tarzda front-end faqat kerakli darajadagi kategoriyalarni olish imkoniga ega bo‘ladi.
  Izoh: Bu usul daraxtni qism-qism bilan yuklash uchun qulay. Misol uchun, foydalanuvchi Erkaklar bo‘limini tanlasa, front-end Erkaklar bo‘limining ID sini bilgan holda `/api/categories/?parent={erkaklar_id}` so‘rovini yuboradi va faqat shu bo‘limning asosiy kategoriyalari ro‘yxatini oladi.

- **GET** `/api/categories/{id}/` – **Kategoriya detali (ixtiyoriy).**
  Ushbu endpoint konkret bitta kategoriya haqida batafsil ma’lumot beradi. Masalan, kategoriya nomi, uning parent bo‘limi (agar bo‘lsa), ichki bolalar kategoriyalar ro‘yxati va hokazo. Agar daraxt strukturasini bir martada olish implementatsiyasi qiyin bo‘lsa, bosqichma-bosqich kategoriyani so‘rab, `children` maydoni orqali navbat bilan ichkariga kirish mumkin.
  Masalan: `/api/categories/10/` so‘rovi `id=10` bo‘lgan kategoriya nomi va uning ichki kategoriyalarini qaytarsin (DRF’ning nested serializer yoki custom so‘rov orqali).

Mahsulotlar bo‘yicha API

- **GET** `/api/products/` – **Mahsulotlar ro‘yxati.**
  Barcha mahsulotlarni yoki filtr bilan tanlangan mahsulotlar ro‘yxatini qaytaradi. Bunda foydalanuvchi yoki front-end quyidagi filtr parametrlaridan foydalanishi mumkin:
- `category={id}` – ma’lum bir kategoriya (yoki sub-kategoriya)ga tegishli mahsulotlargina. Masalan, `/api/products/?category=5` – `id=5` kategoriyaga tegishli mahsulotlar.
- `section=men` yoki `section=women` – jins bo‘limiga qarab filtr. Masalan, `/api/products/?section=men` – faqat Erkaklar bo‘limidagi barcha mahsulotlar (barcha kategoriyalar bo‘ylab). Bu filtr ichki jihatdan tegishli bo‘lim kategoriyalarini topib, ularning ostidagi barcha mahsulotlarni chiqarishi mumkin. Boshqa bir yo‘l – mahsulot modeliga `gender` maydoni qo‘shib, to‘g‘ridan-to‘g‘ri shunga ko‘ra filtr qilish.
- `search=kalitso'z` – mahsulot nomi yoki tavsifi bo‘yicha qidirish (Django REST Framework `SearchFilter` orqali). Bu foydalanuvchiga mahsulotni nomi orqali izlash imkonini beradi.
- Sahifalash (pagination): agar mahsulotlar juda ko‘p bo‘lsa, API standart sahifalash bilan masalan, 20 tadan chiqaradi (`?page=2` va hokazo).

Javob namunasiga har bir mahsulot uchun asosiy maydonlar kiradi: `id`, `name`, `price`, `currency`, `color`, *va ehtimol* qisqa tavsif (`short_description`) agar tavsif uzun bo‘lsa, to‘liq tavsif emas. Rasmlardan faqat bitta asosiy rasm URL adresi thumbnail sifatida berilishi mumkin. Kategoriya haqida ham minimal ma’lumot (masalan, category_id yoki nomi) qaytarilishi mumkin.

- **GET** `/api/products/{id}/` – **Mahsulot detali.**
  Muayyan **id** ga ega mahsulotning barcha detal ma’lumotlarini qaytaradi. Bunda yuqorida bayon etilgan mahsulot atributlari to‘liq beriladi: nomi, narxi, rangi, tavsifi, mavjud o‘lchamlari, **rasmlar ro‘yxati (bir nechtasi)** – masalan, `images` sifatida URL lar massivi, **tegishli kategoriya** (id va nomi yoki butun daraxt: bo‘lim/asosiy/sub nomlari bilan), **yetkazib berish va qaytarish ma’lumoti**, va **sharhlar ro‘yxati** (agar sharhlar ham shu endpointda qaytarilishi ko‘zda tutilsa). Bunda sharhlar ro‘yxati ichida har bir sharhning `name`, `rating`, `title`, `content` va `created_at` kabi maydonlari bo‘ladi (yuqoridagi JSON misolda ko‘rsatilgani kabi). Agar sharhlar soni ko‘p bo‘lsa, ularni ham sahifalash mumkin yoki mahsulot detallari bilan birga cheklangan miqdorda (masalan, 3-5 ta eng yangi sharh) jo‘natib, to‘liq sharhlar uchun alohida endpointga murojaat qilish taktikasi tanlanishi mumkin.

- **POST** `/api/products/` – **Yangi mahsulot yaratish** *(faqat adminlar uchun)*.
  Oddiy foydalanuvchilar uchun ochiq emas. Bu endpoint orqali admin (yoki tizim) yangi mahsulot qo‘shishi mumkin. So‘rov body qismida barcha kerakli maydonlar (nomi, narxi, kategoriya id, rang, o‘lchamlar va hokazo) JSON shaklda yuboriladi. Tekshiruvdan so‘ng mahsulot yaratiladi va javobda uning ma’lumotlari qaytariladi. Real loyihada, odatda adminlar uchun web-based Django Admin mavjud bo‘lgani sababli, alohida REST endpoint muhim emas. Lekin agar admin panelni ham SPA yoki frontend orqali qilish niyat qilinganda, JWT token bilan himoyalangan shu kabi endpointlar kerak bo‘ladi.

- **PUT/PATCH** `/api/products/{id}/` – **Mahsulot ma’lumotlarini tahrirlash** *(adminlar uchun)*.
  Ma’lum mahsulotni yangilash uchun. Masalan, narxini o‘zgartirish, tavsifini yangilash kabi. Bu ham autentifikatsiyalangan (admin) foydalanuvchi uchungina amal qiladi.

- **DELETE** `/api/products/{id}/` – **Mahsulotni o‘chirish** *(adminlar uchun)*.
  Bu orqali bazadan mos mahsulot (va unga bog‘liq rasmlar, sharhlar) o‘chirib yuboriladi. Tasdiqlangan token yoki admin huquqlari bilan himoyalangan bo‘lishi lozim.

(Adminlar uchun mo‘ljallangan POST/PUT/DELETE endpointlar umumiy foydalanuvchilardan himoyalanadi. Ushbu hujjatda ko‘proq ochiq API talablari e’tiborga olingan.)

Sharhlar (Mijoz fikrlari) bo‘yicha API

- **GET** `/api/products/{product_id}/reviews/` – **Mahsulotning barcha sharhlari ro‘yxati.** Muayyan mahsulotga tegishli barcha sharhlarni qaytaradi. Agar mahsulot detallari API’da sharhlar allaqachon berilayotgan bo‘lsa, bu endpoint shart emas. Ammo, ba’zan sharhlarni alohida sahifada yoki o‘zicha yuklash kerak bo‘lsa, ushbu alohida endpointdan foydalaniladi. Javob ichida sharhlar ro‘yxati va ehtimol total sharhlar soni kabi metadata bo‘lishi mumkin.

- **POST** `/api/products/{product_id}/reviews/` – **Yangi sharh yuborish.** Foydalanuvchi tomonidan front-enddagi sharh formasi to‘ldirilgach, ushbu endpointga POST so‘rovi yuboriladi. So‘rov JSON tarkibida quyidagi ma’lumotlar bo‘ladi: `name`, `email`, `rating`, `title`, `content`. Server bu ma’lumotlarni tekshiradi (validatsiya):

- Barcha majburiy maydonlar to‘ldirilganini (bo‘sh emasligini) tasdiqlaydi.
- `email` formati to‘g‘ri ekanini tekshiradi.
- `rating` 1-5 oralig‘ida ekanini tasdiqlaydi.
- `title` va `content` so‘zlar soni chegarasiga rioya qilinganini tekshiradi (agar bu logika implementatsiya qilingan bo‘lsa).

Agar biror xatolik bo‘lsa, **400 Bad Request** javobi qaytariladi va unda qaysi maydon xato ekani hamda xatolik xabari JSON ko‘rinishida beriladi. Masalan:

```
{
  "email": ["Elektron pochtangiz noto‘g‘ri formatda kiritilgan."],
  "title": ["Fikr sarlavhasi 10 ta so‘zdan oshmasligi kerak."]
}
```

Agar hammasi to‘g‘ri bo‘lsa, yangi sharh yaratiladi (`Review` yozuvi bazaga qo‘shiladi) va **201 Created** javobi qaytariladi. Javob JSON tarkibida yaratilgan sharhning ma’lumotlari (ID, name, rating, title, content, created_at va h.k.) bo‘lishi mumkin yoki bo‘sh body bilan 201 status qaytarilishi ham yetarli (front-end qayta `/reviews/` GET orqali yangilangan ro‘yxatni olishni tanlashi mumkin).

- **DELETE/PUT** `/api/reviews/{id}/` – **Sharhni o‘chirish yoki o‘zgartirish** *(ixtiyoriy, adminlar uchun)*. Oddiy foydalanuvchilar uchun sharhni o‘chirib tashlash yoki tahrirlash imkoni odatda berilmaydi (mehmon sifatida qoldirayotgani uchun). Biroq, admin panel orqali yoki himoyalangan API orqali admin sharhni o‘chirishi mumkin. Agar shu funksionallik REST API orqali ham kerak bo‘lsa, admin autentifikatsiyasi bilan ushbu endpointlar ishga tushadi.

Boshqa (Qo‘shimcha) API endpointlar

Loyiha hajmi ortib borishi bilan quyidagilar ham qo‘shilishi mumkin (hozircha TZ doirasida talab qilinmagan, faqat to‘liq manzarani tushunish uchun keltirilmoqda): - **Auth API** (ro‘yxatdan o‘tish, tizimga kirish) – foydalanuvchi akkaunti talab qilinsa. - **Buyurtmalar (Orders) API** – xarid savati (cart) va buyurtma berish jarayoni qo‘shilsa. - **Mahsulotlarga “Like” yoki “Wishlist”** – kelajakdagi funksionallik uchun. - **Filtrlash va saralash API** – narx bo‘yicha yoki reyting bo‘yicha saralash parametrlari qo‘shimcha query param sifatida.

Hozircha ushbu funksiyalar Figma dizaynida va talablar ro‘yxatida nazarda tutilmagan, shuning uchun implementatsiya qilinmaydi.

Admin Panel va Kontent Boshqaruvi

Loyihada **Django Admin paneli** backend kontentini boshqarish uchun ishlatiladi. Admin panel orqali quyidagi vazifalar amalga oshiriladi:

- **Kategoriyalarni boshqarish:** Admin yangi bo‘limlar (masalan, *Erkaklar*, *Ayollar*) yoki yangi asosiy/kichik kategoriyalar qo‘shishi, nomini o‘zgartirishi yoki o‘chirishi mumkin. Kategoriya tuzilmasini noto‘g‘ri buzmaslik uchun, muhim kategoriyalarni o‘chirganda ogohlantirishlar ko‘rsatiladi. Kategoriya qo‘shishda uning parent’i tanlanadi.
- **Mahsulotlarni boshqarish:** Admin panelda “Mahsulotlar” bo‘limida yangi mahsulot qo‘shish formasi mavjud. Unda yuqorida sanab o‘tilgan barcha mahsulot maydonlari kiradi (nomi, narxi, rang, o‘lchamlar, kategoriya va hokazo). Shuningdek, mahsulot qo‘shilgandan so‘ng, unga rasmlar biriktirish uchun alohida bo‘lim (inline) mavjud: admin bir yoki bir nechta rasm yuklab, ularning tartibini belgilaydi.
- **Rasmlarni boshqarish:** Mahsulotlarni tahrirlash oynasida biriktirilgan rasmlarni ko‘rish, yangisini qo‘shish yoki mavjudini o‘chirish mumkin. Django admin buni ProductImage inline modeli orqali ta’minlaydi.
- **Sharhlarni boshqarish:** Admin panelda “Sharhlar” bo‘limida barcha mijoz fikrlari ro‘yxati ko‘rinadi. Admin nomaqbul sharhlarni sildirib tashlashi yoki zarur hollarda matnini tahrir qilishi mumkin (masalan, haqoratli so‘zlarni olib tashlash). Agar sharhlar dastlab moderatsiya talab qilsa, admin yangi sharhni ko‘rib chiqib, “tasdiqlash” belgisini qo‘yishi mumkin edi – lekin hozirgi talablarga ko‘ra, sharhlar avtomatik chiqadi.
- **Ma’lumotlarning yaxlitligi:** Admin panel orqali kirim qilinadigan ma’lumotlar (masalan, mahsulot narxi son bo‘lishi, kategoriya tanlangan bo‘lishi, va hokazo) maxsus forma validatsiyalari bilan tekshiriladi. Django admin bunga ko‘p hollarda o‘zi yetarli (model field turlari orqali), qo‘shimcha tarzda `ModelAdmin` class’da `formfield_clean` va boshqa usullar bilan cheklovlar qo‘shilishi mumkin.

Admin huquqlari: Faqat admin huquqiga ega foydalanuvchilar (superuser yoki staff) admin panelga kira oladi. Oddiy foydalanuvchilarning admin panelga kira olmaydi va ularning API orqali ham ma’lumot o‘zgartirish imkoniyati yo‘q (faqat sharh qoldirish kabi cheklangan interaktivlikka ega).

Texnik konfiguratsiya: Django admin paneli standart tarzda `/admin/` URL manzilida faol bo‘ladi. Loyihani ishga tushirganda dasturchi `createsuperuser` orqali birlamchi admin foydalanuvchini yaratadi. Keyinchalik shu admin orqali boshqa adminlarni ham qo‘shish mumkin.

Xulosa

Ushbu texnik topshiriqda **“E-Commerce Platforma”** veb-loyihasining backend qismi uchun barcha asosiy talablar va texnik yechimlar bayon qilindi. Dizayn talablariga muvofiq: - Ma’lumotlar modeli kategoriyalar, mahsulotlar, rasmlar va sharhlarni qamrab olgan holda tuzildi. - Django va Django REST Framework asosida API lar loyihalashtirildi va ularning vazifalari ko‘rsatib o‘tildi. - Har bir bo‘lim va funksional qismlar uchun batafsil izohlar berildi: jins bo‘limlari va kategoriyalar iyerarxiyasi, mahsulotlar atributlari, sharhlar tizimi va validatsiya qoidalari. - Admin paneli orqali kontentni boshqarish ssenariylari tasvirlandi.

Mazkur hujjatni asos sifatida olib, backend dasturchi loyihani amalga oshirishi mumkin. Barcha mazmun DOCX formatdagi hujjatga ko‘chirish va taqdim etishga tayyor. Endi loyiha implementatsiyasiga kirishishda, ushbu texnik topshiriq **mo‘ljal** bo‘lib xizmat qiladi. Backend dasturchidan belgilangan model strukturalari va API endpointlar bo‘yicha kod yozish hamda ularni front-end bilan integratsiya qilish kutiladi.

Loyiha muvaffaqiyatli amalga oshirilishi uchun ushbu texnik topshiriqdagi ko‘rsatmalarga rioya qilish talab etiladi. Backend qismi tayyor bo‘lgach, front-end (UI) bilan birgalikda sinovdan o‘tkaziladi va Figma dizayndagi funksional holatlarga mosligi tekshiriladi.
