**O’ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI**

**MUHAMMAD AL-XOZAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI**

**DT arxitekturasi fanidan**

**MUSTAQIL ISH**

**TEKIN MARKET WEB SAHIFASINI ISHLAB CHIQISH**

**Guruh: 313-21 guruh talabasi**

**Bajardi: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Tekshirdi:Otaxonova Bahrixon**

**TEKIN MARKET WEB SAHIFASINI ISHLAB CHIQISH**

**REJA:**

* KIRISH
* Tekin market web sahifasining arxitekturasini tanlash
* Even drive arxitekturasini loyihalash
* Web sahifa uchun Microservis arxitekturasni qo’llash
* Loyihani yaratishda turli xil arxitekturalardan foydalanish
* Xulosa
* Foydalanilgan adabiyotlar va web sahifalar

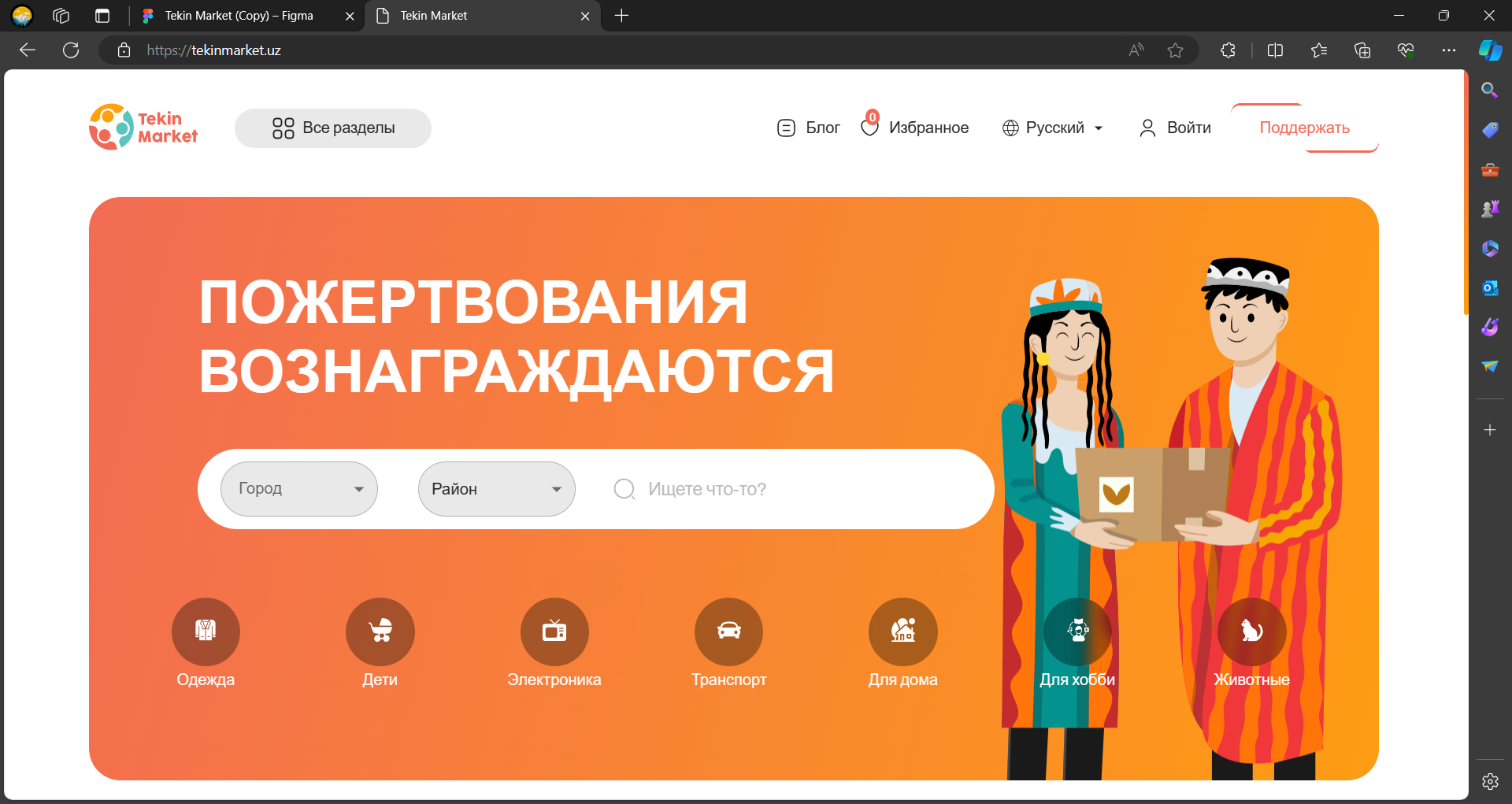
KIRISH

**TEKIN MARKET WEBSITE**

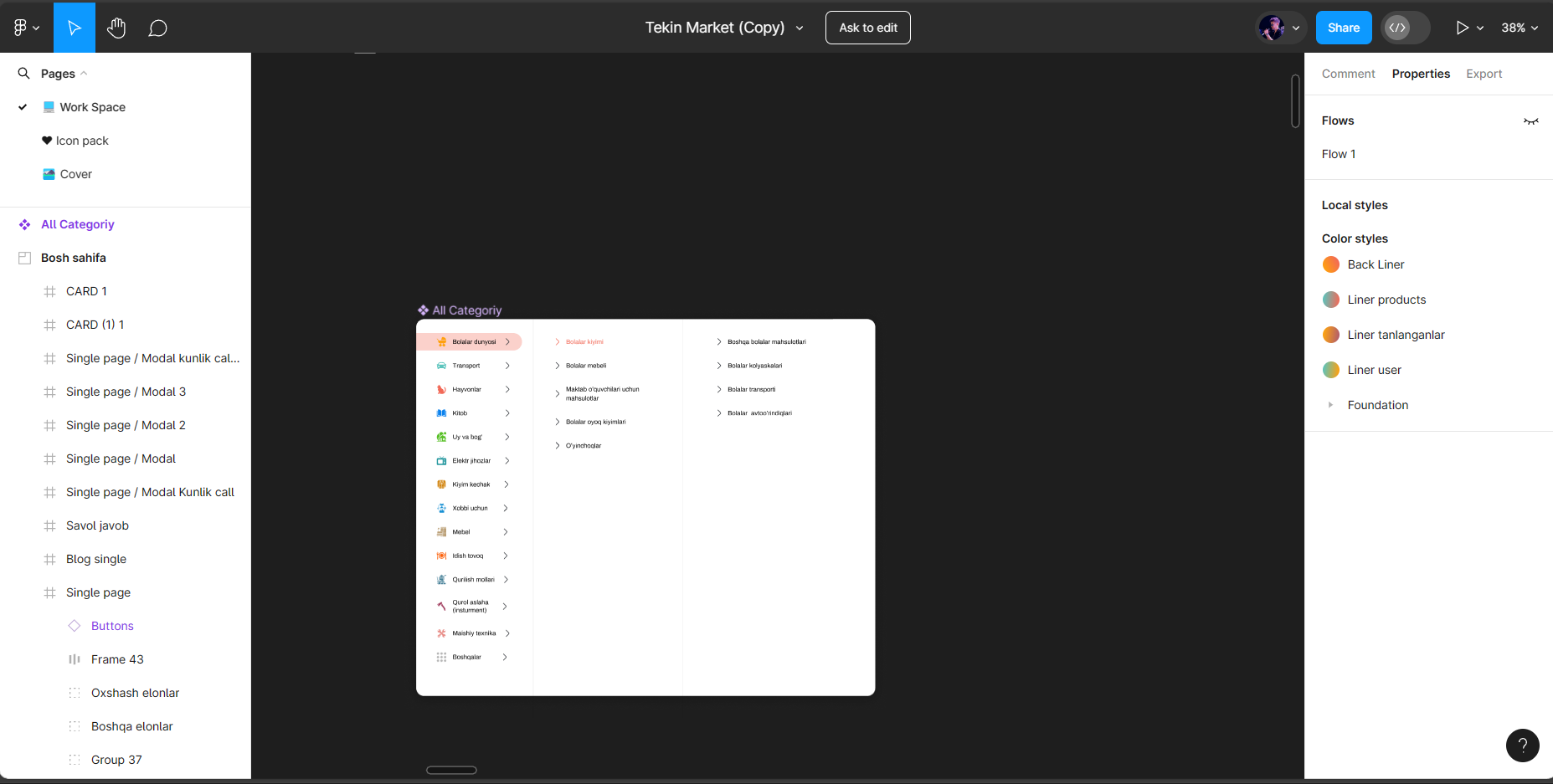
**Tekin market web sahifasida insonlar o’z mahsulotlarining elonlarini berishi mumkin. Bu web sahifa foydalanuvchilarga qulaylik va soddaligi bilan foydalanishni osonlashtiradi. Bu web sahifada mahsulotlar uchun kataloglar va qulay qidiruv tizimi mavjud.**

Bu yerda web sahifaning home page qismi keltirilgan.

Tekin market web sahifasi manzili --- [Tekin Market](https://tekinmarket.uz/)

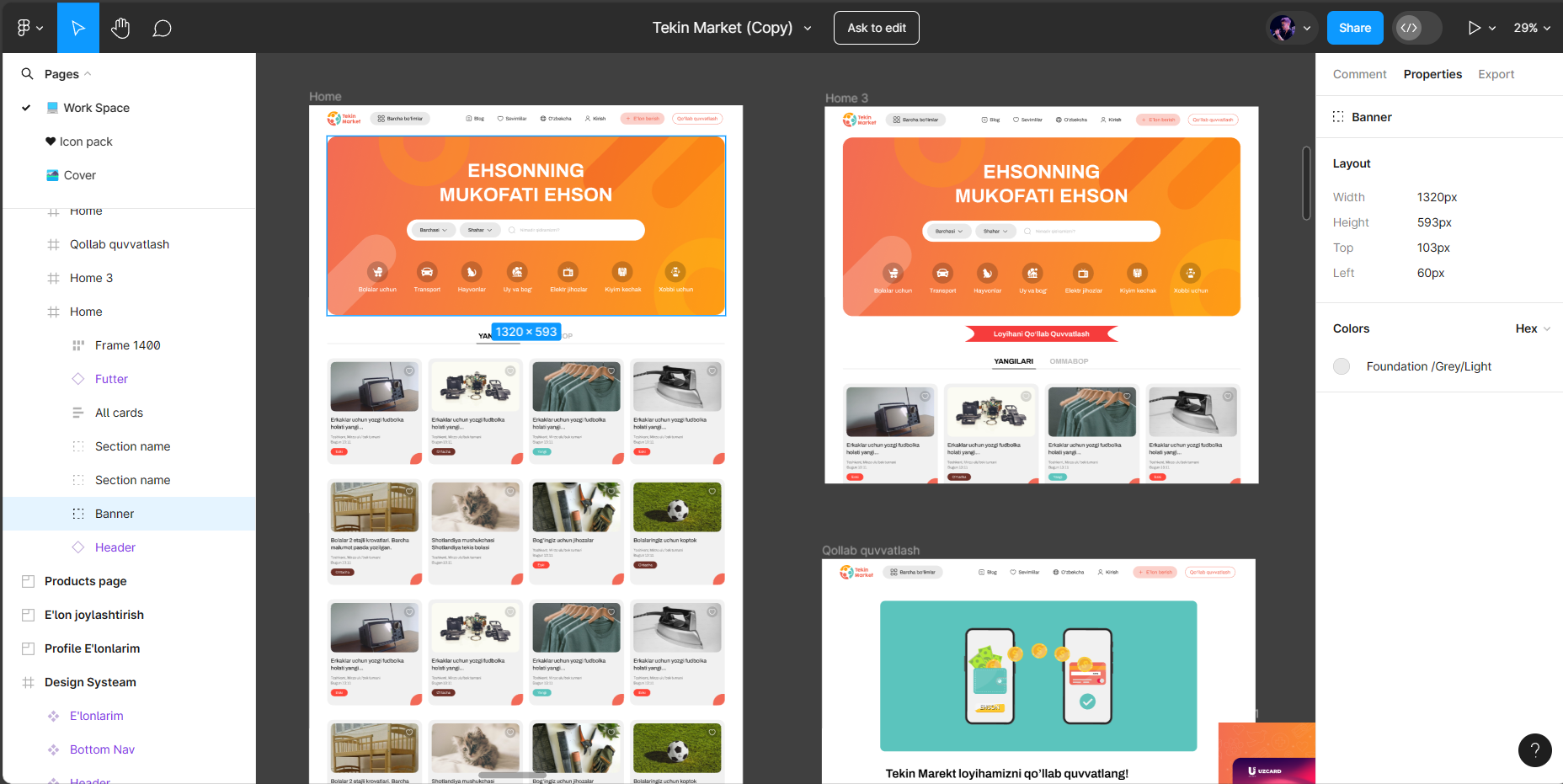


**1-rasm . Home page qismi**



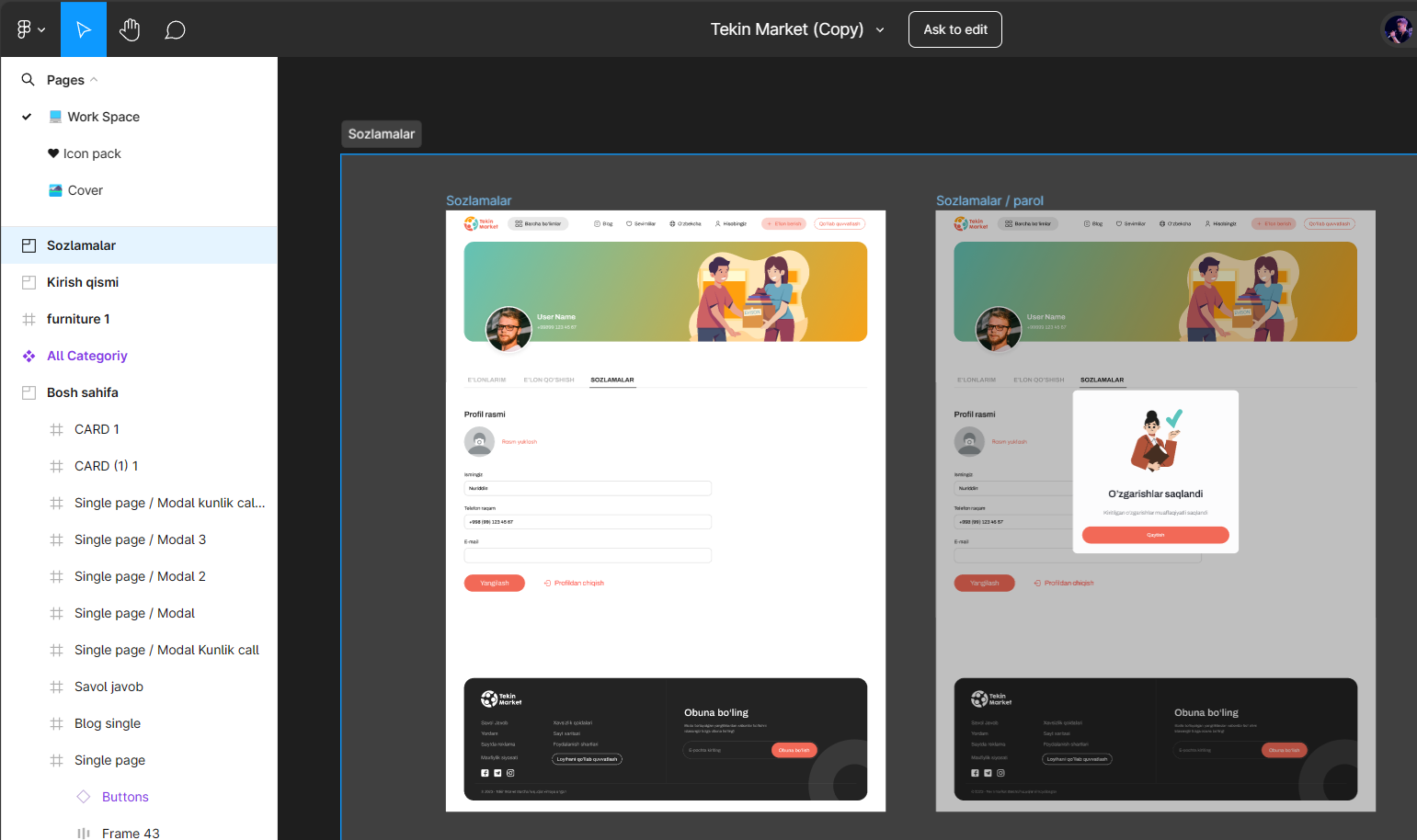
2 – rasm . Web sahifaning All category qismining figmadagi ko’rinishi

Tekin market web sahifasining All category qismining figmadagi ko’rinishini keltirdim. Bu yerda barcha mahsulot toifalari bir ustun ko’rinishida joylashtirilgan. Ekran ikki qismga bo’linga o’ng tomonida mahsulot kategoriyalarining turlari keltirilgan. Bu toifalarga logolar yaratilgan va ularning chap tomoniga joylashtirilgan.



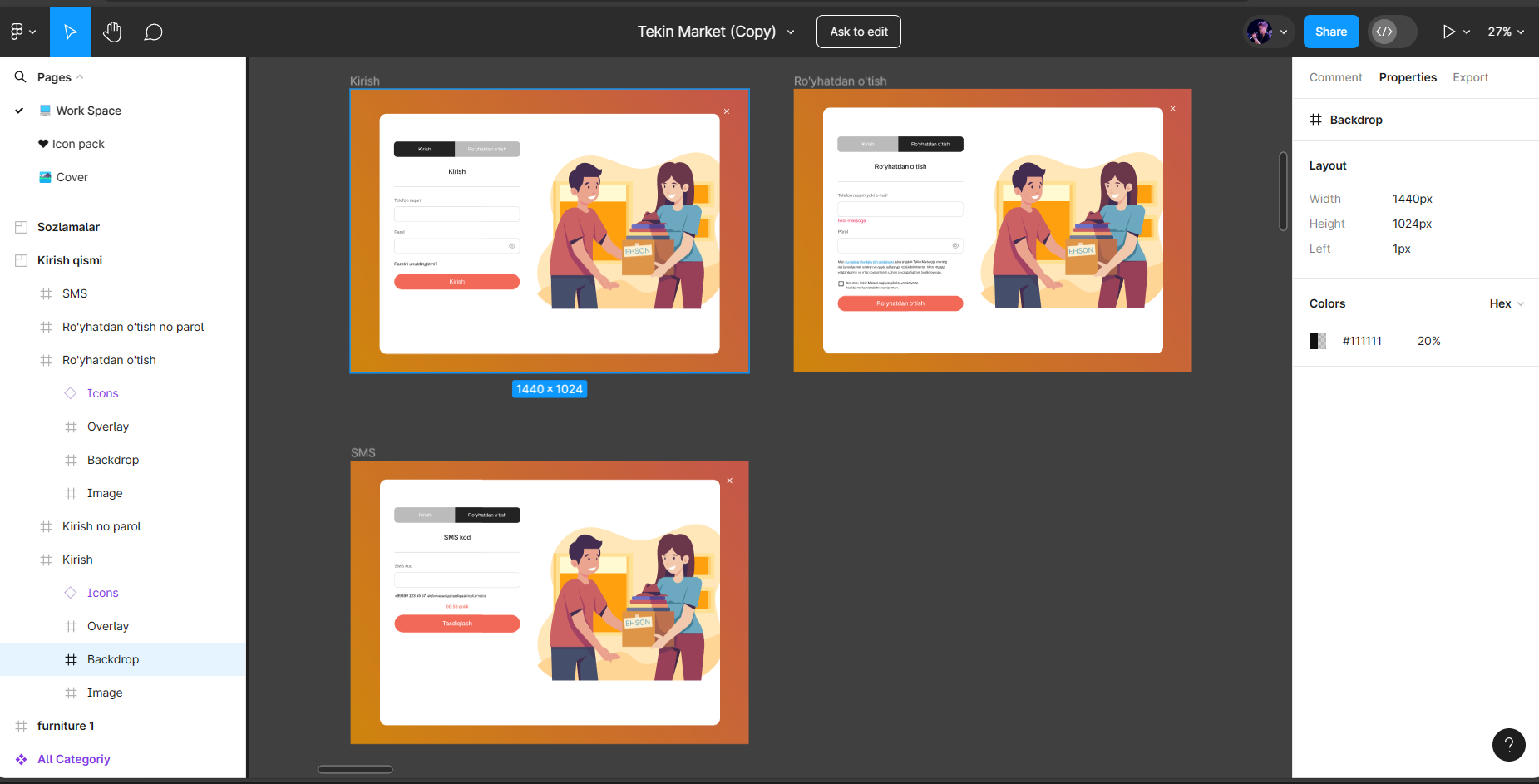
3-rasm. Home page qismining figmasi keltirilgan

Home page turli xil dizaynlar orqali yaratilgan va obyektlarning joylashuvi aniq keltirilgan. Bu yerda elonlarning rasmini va tagida uning tasnifi yoritilgan bo’ladi. Undan tashqari qo’llab quvatlash qismi ham keltirilgan.



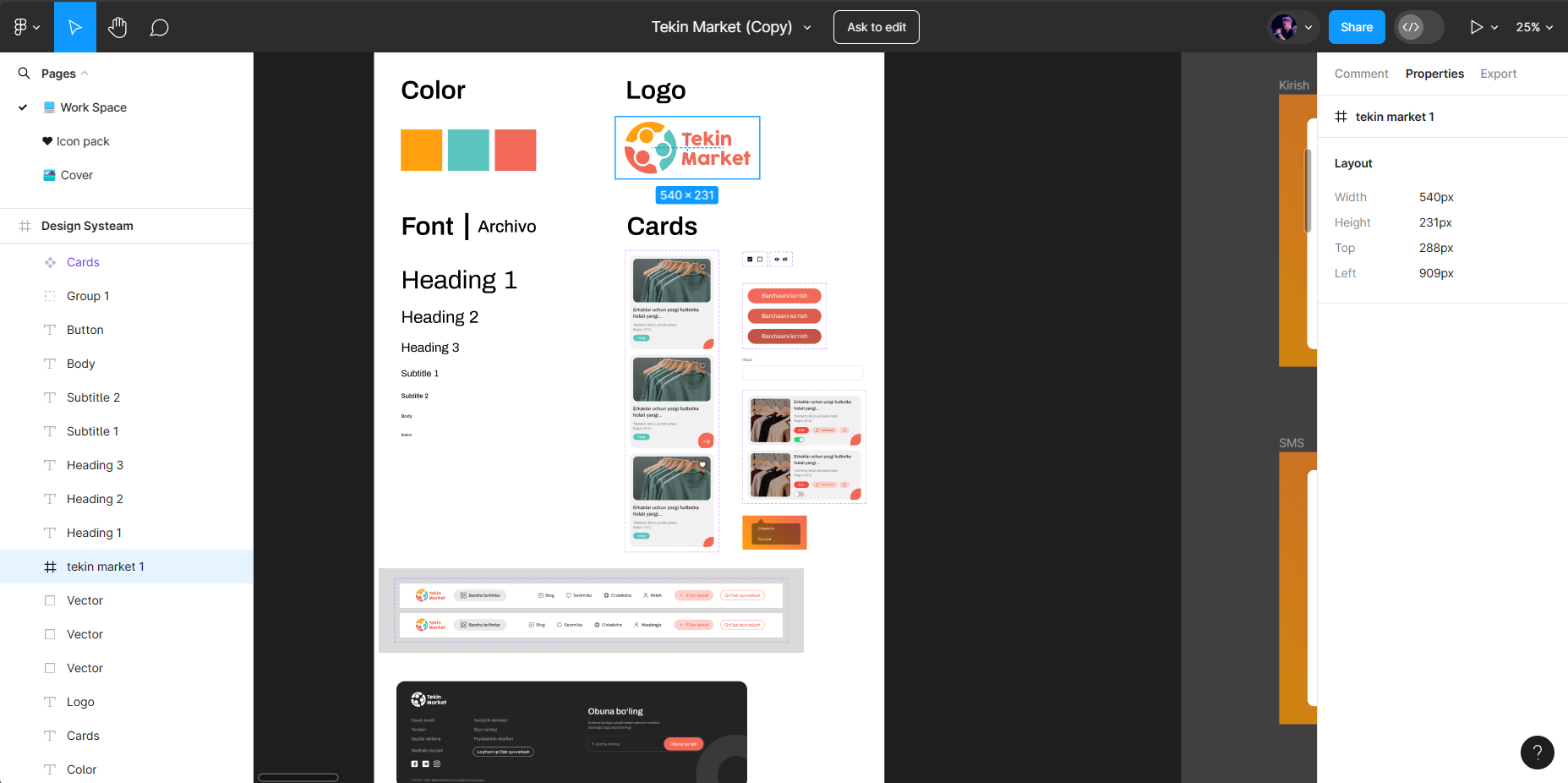
4-rasm. Profile sozlamalari keltirilgan.

Bu yerda profile ma’lumotlari to’laligicha keltirilgan. Profile fotosi va bazi malumotlar kiritiladi. Nomer va ism albatta kirirtish shart. Bu yerga ma’lumotlarni o’zgartirish button ni ham joylashtirilgan.



5-rasm. Bu web sahifaga kirish va ro’yxatdan o’tish qismi keltirilgan

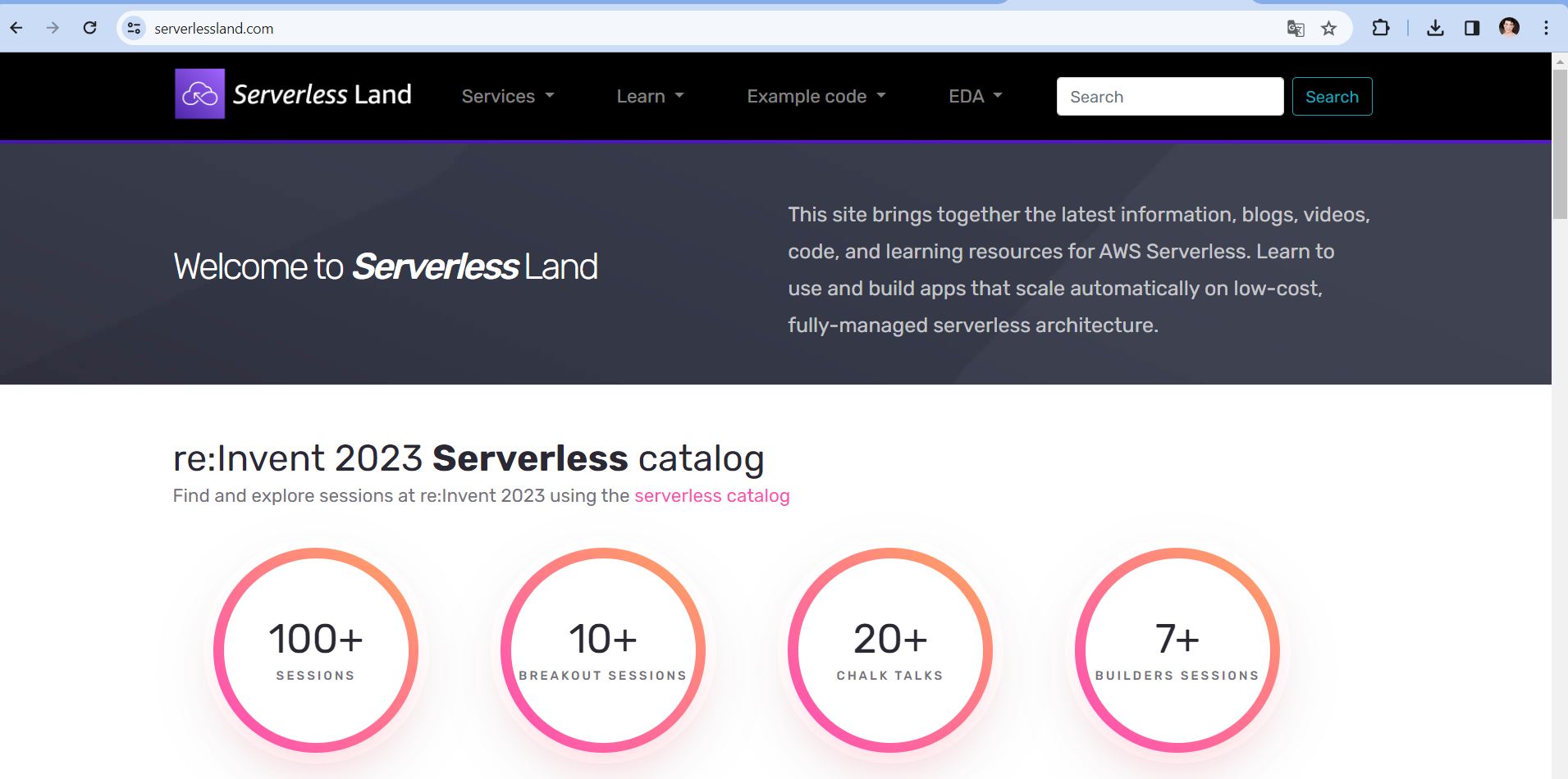
Birinchi novbatda bu yerda ro’yxatdan o’tish qismi mavjud. U yerga telefon orqali ro’yxatdan o’tish tizimi qo’llaniladi. Agar ro’yxatdan o’tgan bo’lsa kirish orqali nomer bilan web sahifaga kirishni amalga oshiradi.



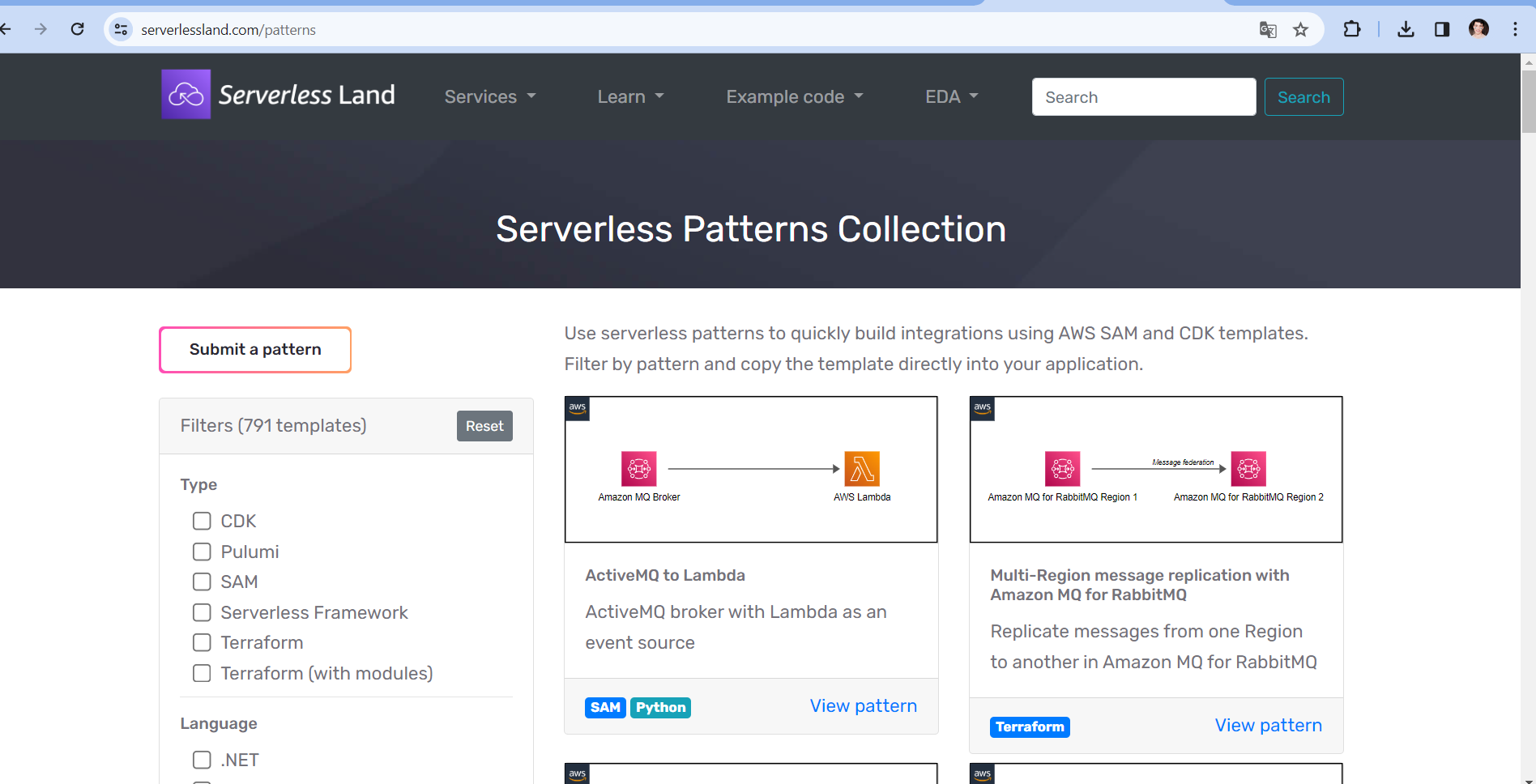
6-rasm. Bu yerda dizayn elementlari keltirilgan.

Bu yerda web sahifa ranglari , logotipi, shrift o’lchamlari, va elonlar ko’rinishi yaratilgan.

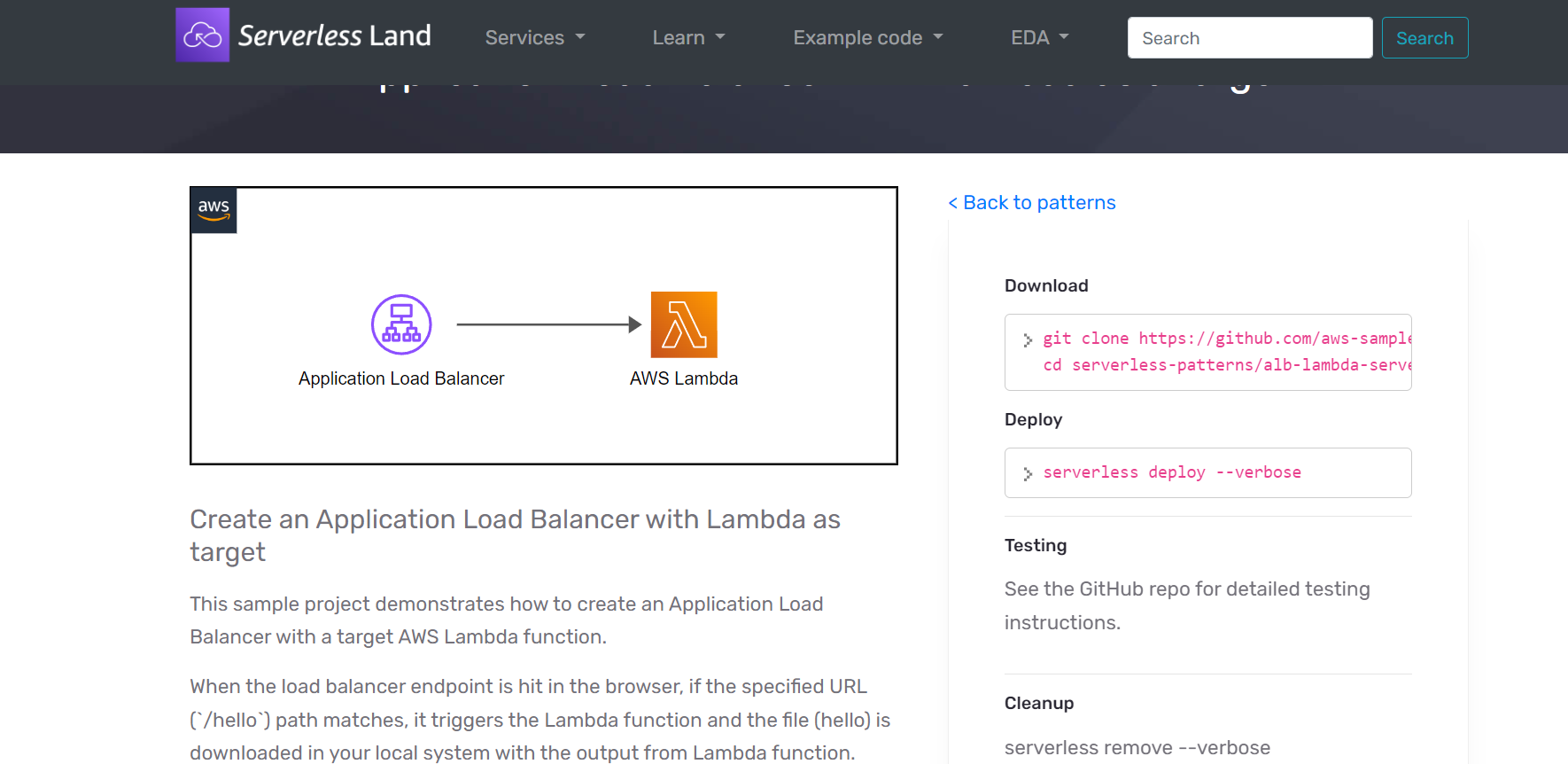
Tekin market web sahifasining arxitekturasini tanlash



Bu serverless Land tarmog’i bo’lib bu web sahifada barcha server arxitekturalariga misollar keltirilgan va ular uchun eng yaxshi kodlarga misollar yaratilgan. Bu saytda biz albatta turli xil arxitekturalarni tanlab undan foydalanishimiz mumkin.



Bu rasmda ham turli xil turdagi server arxitekturalari mavjuda ularni eng yaxshisini tanlab olish mumkin.



Bu yerda men foydalangan server arxitekturasi bo’lib Ushbu namunaviy loyiha maqsadli AWS Lambda funksiyasi bilan Application Load Balancerni qanday yaratishni ko'rsatadi.

Brauzerda yuk balansi so'nggi nuqtasiga urilganda, belgilangan URL (`/salom`) yo'liga mos kelsa, u Lambda funksiyasini ishga tushiradi va fayl (salom) Lambda funksiyasidan chiqqan holda mahalliy tizimingizga yuklab olinadi.

service: alb-lambda-serverless

frameworkVersion: '^3.28.1'

useDotenv: true

provider:

name: aws

runtime: nodejs14.x

memorySize: 256

timeout: 30

# override the default stage (dev) to be `prod`, or you can use the `--stage` CLI option

stage: ${opt:stage, "prod"}

# specify the region using --region flag while deploying the stack, else the default region ap-south-1 will be chosen

region: ${opt:region, "ap-south-1"}

iamRoleStatements:

- Effect: Allow

Action: s3:\*

Resource: '\*'

- Effect: Allow

Action:

- lambda:InvokeFunction

Resource: '\*'

functions:

# lambda function with the alb event configured in it.

functionOne:

handler: handlers/alb-lambda-handler.handler

events:

- alb:

listenerArn: { Ref: HTTPListener }

priority: 1

conditions:

path: /hello

method: GET

resources:

Resources:

# creates a VPC with 2 Public subnets within the VPC

VPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: 12.2.1.0/24

EnableDnsHostnames: true

EnableDnsSupport: true

SubnetOne:

Type: AWS::EC2::Subnet

Properties:

VpcId: { Ref: VPC }

CidrBlock: 12.2.1.0/25

MapPublicIpOnLaunch: true

AvailabilityZone: !Select

- 0

- Fn::GetAZs: {Ref: 'AWS::Region'}

SubnetTwo:

Type: AWS::EC2::Subnet

Properties:

VpcId: { Ref: VPC }

MapPublicIpOnLaunch: true

CidrBlock: 12.2.1.128/25

AvailabilityZone: !Select

- 1

- Fn::GetAZs: { Ref: 'AWS::Region'}

# security group that allows all the http/s traffic to the load balancer

SecurityGroupALB:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: SG created via serverless framework

VpcId: { Ref: VPC }

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: 80

ToPort: 80

CidrIp: 0.0.0.0/0

- IpProtocol: tcp

FromPort: 443

ToPort: 443

CidrIp: 0.0.0.0/0

# internet gateway that connects the vpc to the internet

InternetGateway:

Type: AWS::EC2::InternetGateway

AttachGateway:

Type: AWS::EC2::VPCGatewayAttachment

Properties:

VpcId: { Ref: VPC }

InternetGatewayId: { Ref: InternetGateway }

PublicSubnetRouteTable:

Type: AWS::EC2::RouteTable

Properties:

VpcId: { Ref: VPC }

# creates public routes for the subnets

PublicSubnetRoute:

Type: AWS::EC2::Route

DependsOn: AttachGateway

Properties:

RouteTableId: { Ref: PublicSubnetRouteTable }

DestinationCidrBlock: 0.0.0.0/0

GatewayId: { Ref: InternetGateway }

PublicSubnetRouteTableAssociationOne:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: { Ref: PublicSubnetRouteTable }

SubnetId: { Ref: SubnetOne }

PublicSubnetRouteTableAssociationTwo:

Type: AWS::EC2::SubnetRouteTableAssociation

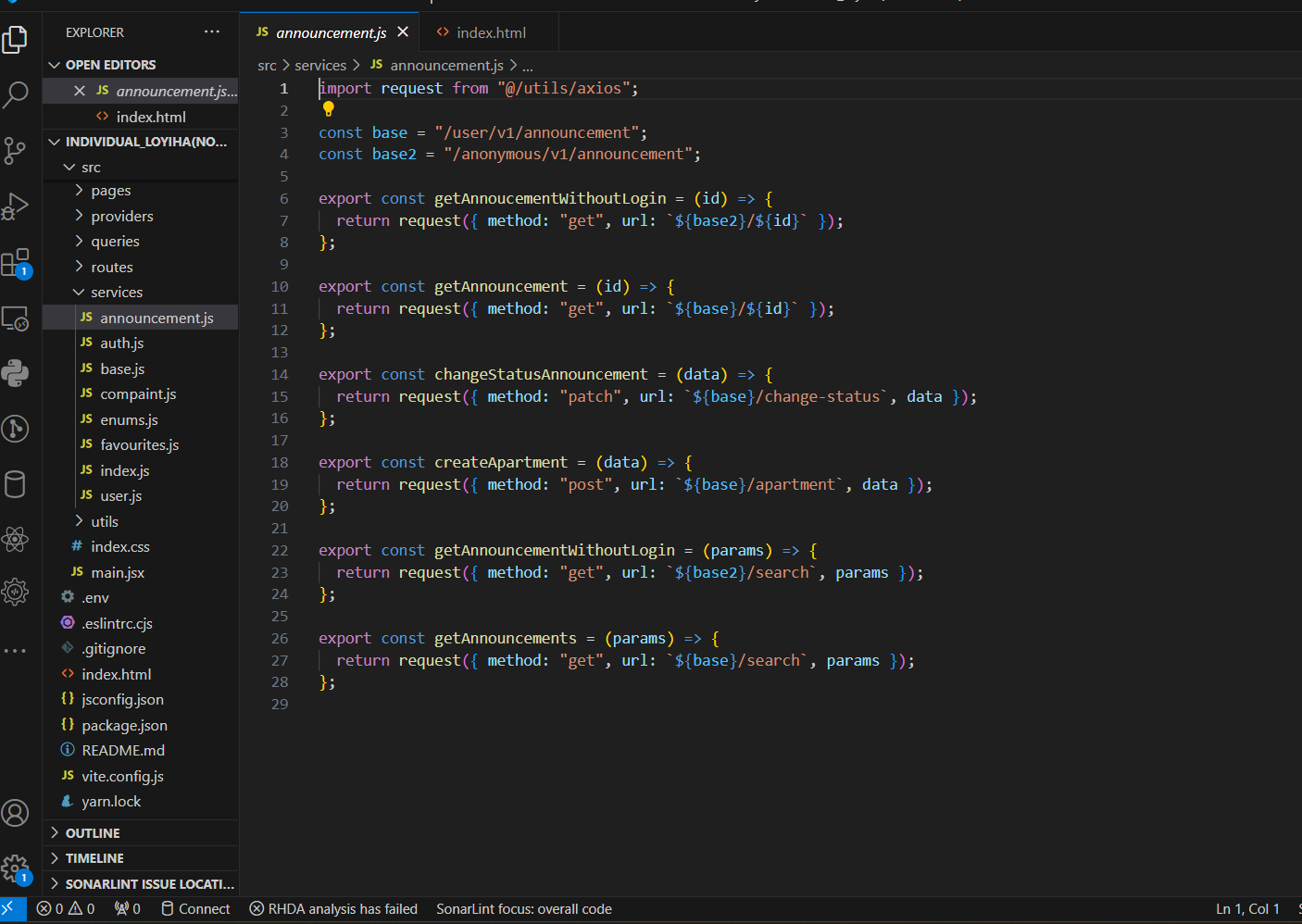
Properties:

RouteTableId: { Ref: PublicSubnetRouteTable }

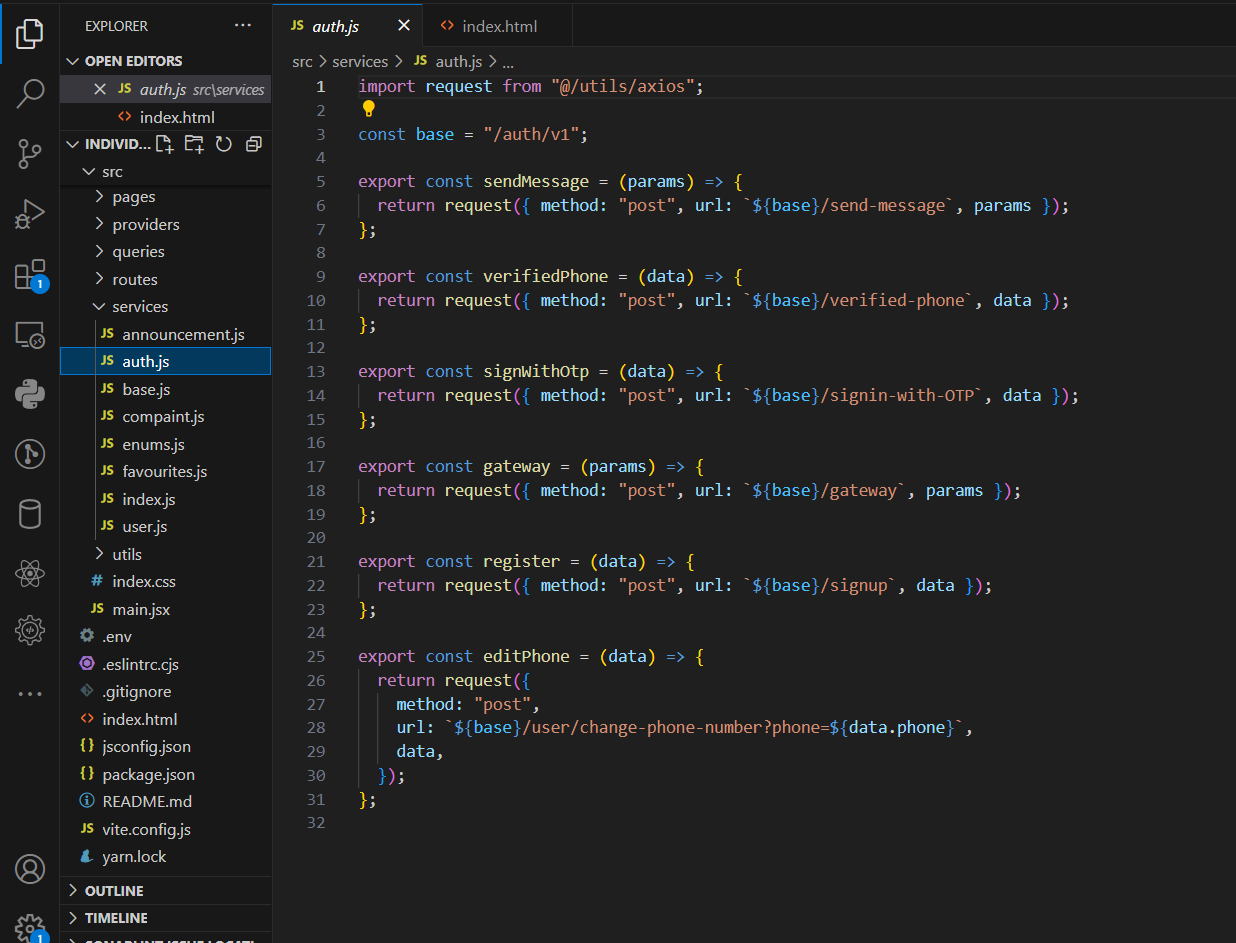
SubnetId: { Ref: SubnetTwo }

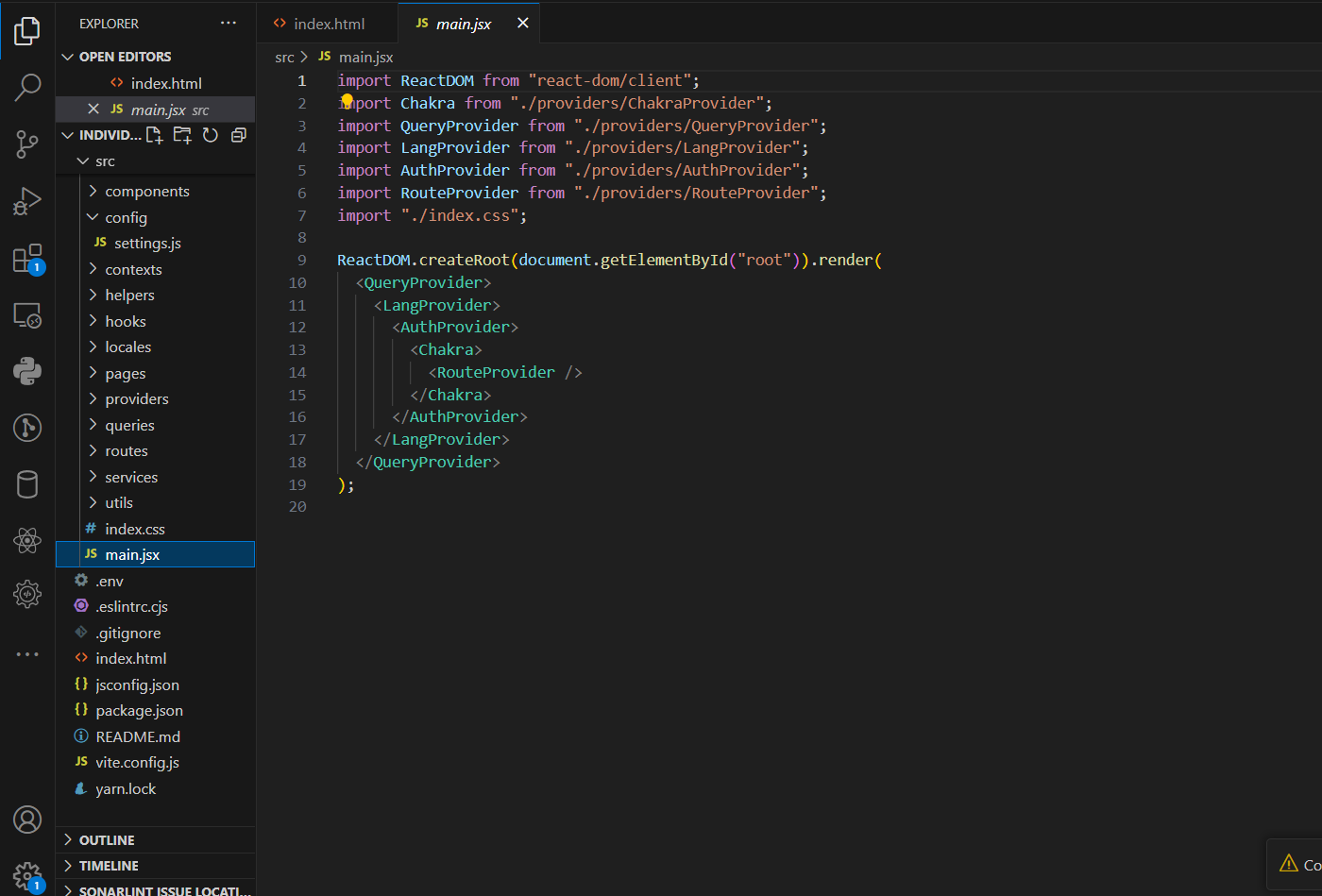
# creates a internet-facing load balancer

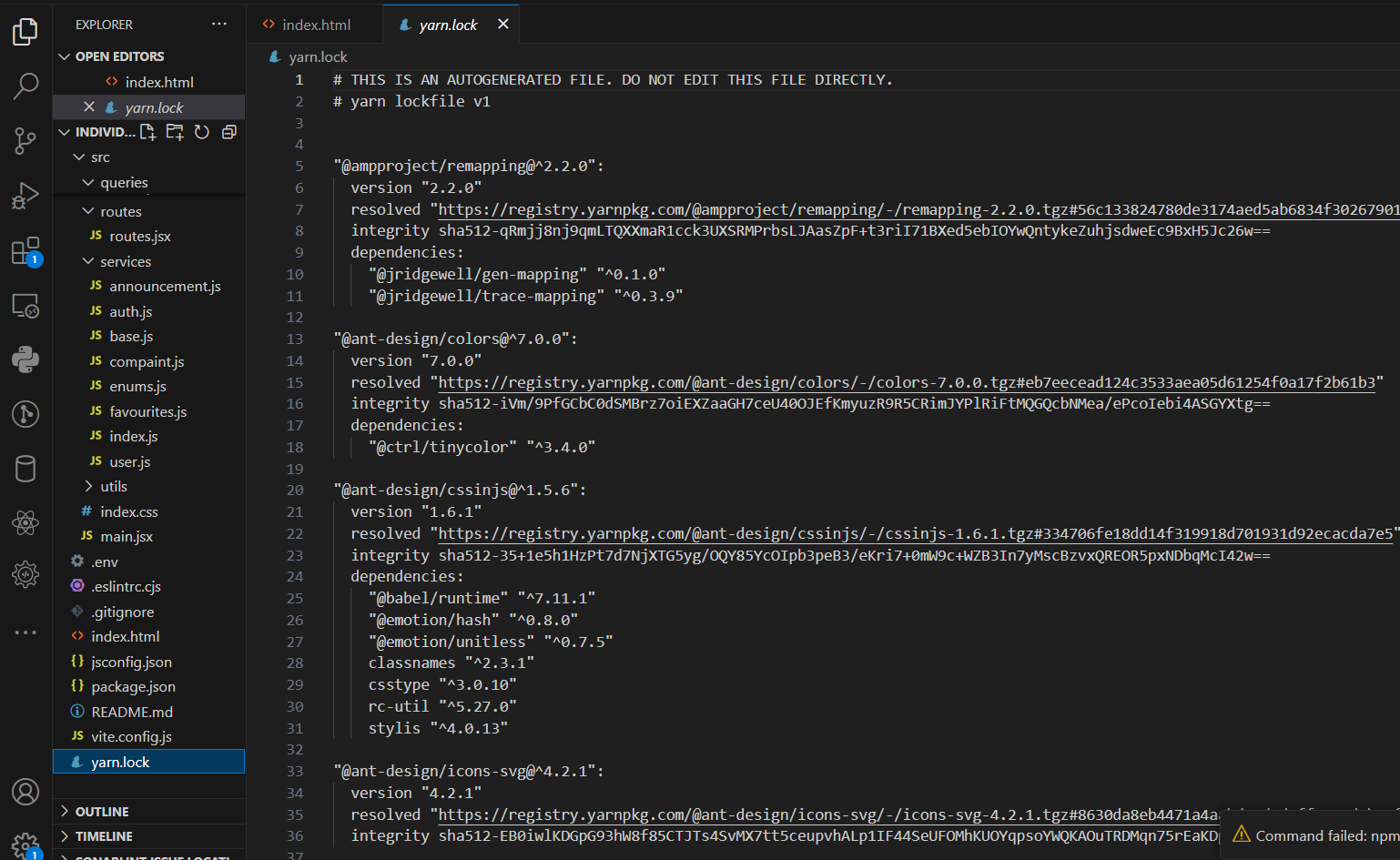
LoadBalancer:



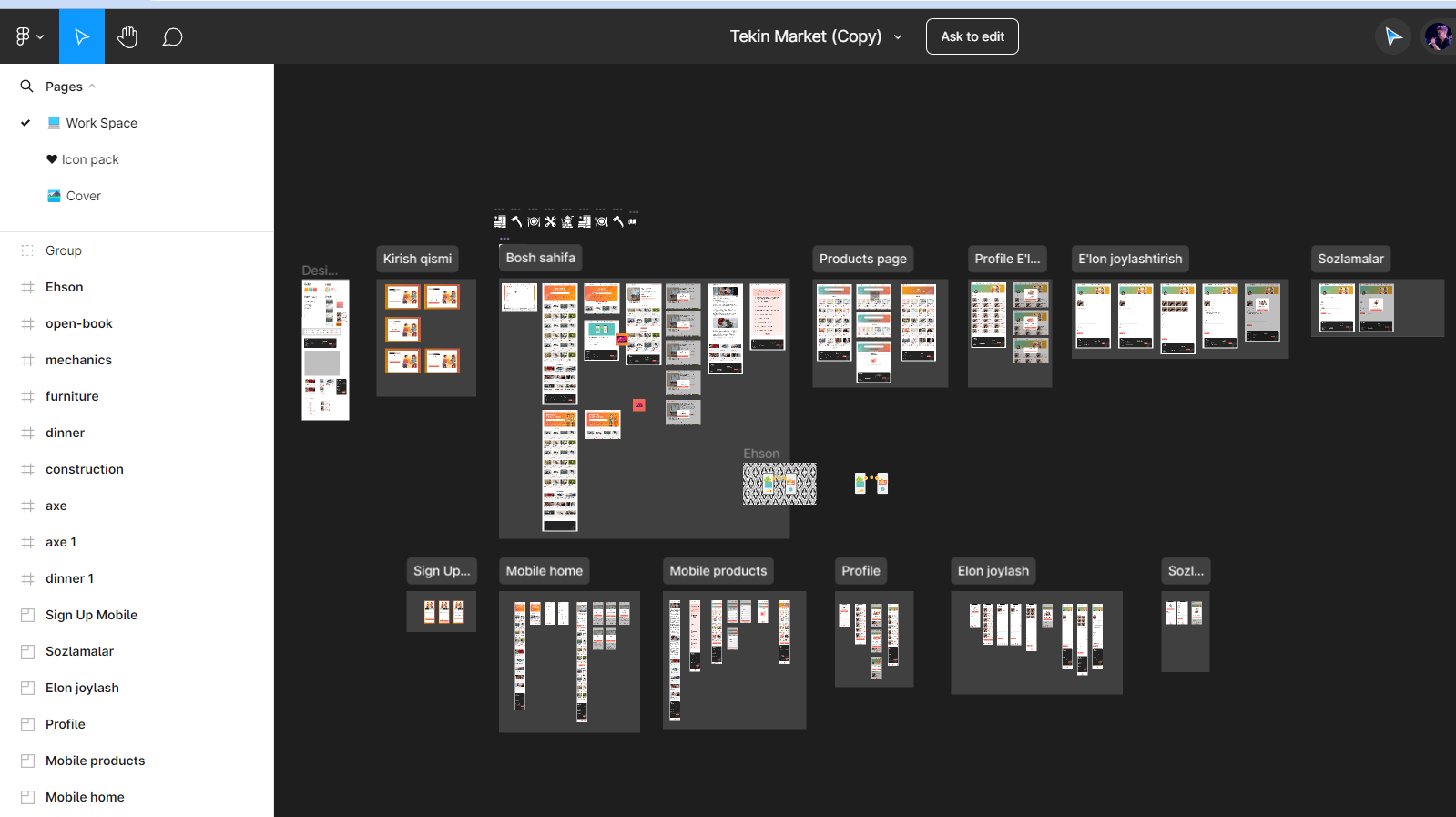
Ushbu loyiha Tekin market loyihasining umumiy kodi bu yerda barcha server va servis kodlarini keltirganman bu server orqali turli xil qidirish yoki turli xil so’rovlarni server orqali bajarishni taminlaydi.







Yuqorida barcha kodlar taqdim etildi u yerda serverlar bilan bog’lanish mavjud va har doim serverdan ma’lumotlarni yangilab turadi. Undan tashqari hech kim ma’lumotlarni serverdan to’g’ridan to’g’ri o’zgartira olmaydi ya’ni bu web sahifa arxitekturasi faqat ruhsat berilganlarga bu imkonni beradi. Serverless websahifasida juda ko’plab server arxitekturalari keltirilgan ularni o’rganish va foydalanish oson qilib yaratilgan.

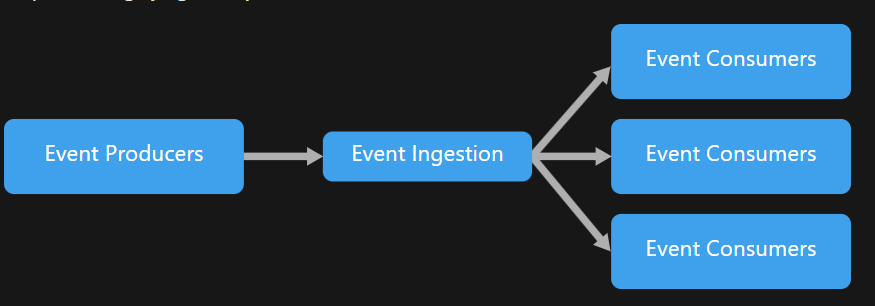


Bu Tekinmarket saytining to’liq figma ko’rinishi

Even drive arxitekturasini loyihalash

**Event-driven architecture**

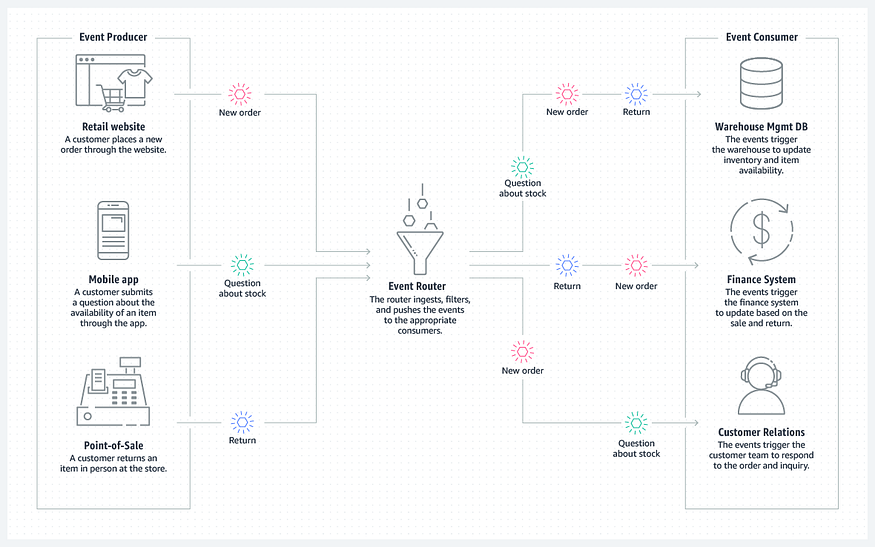
Voqealarga asoslangan arxitektura voqealar oqimini yaratadigan **voqea ishlab chiqaruvchilardan** va voqealarni tinglaydigan **voqea iste'molchilaridan iborat.**



Voqealar deyarli real vaqtda yetkazib beriladi, shuning uchun iste'molchilar sodir bo'lgan voqealarga darhol javob berishlari mumkin. Ishlab chiqaruvchilar iste'molchilardan ajratilgan - ishlab chiqaruvchi qaysi iste'molchilar tinglayotganini bilmaydi. Iste'molchilar ham bir-biridan ajralib turadi va har bir iste'molchi barcha voqealarni ko'radi. Bu [Raqobatdosh iste'molchilar](https://learn.microsoft.com/en-us/azure/architecture/patterns/competing-consumers) namunasidan farq qiladi, bunda iste'molchilar navbatdan xabar olishadi va xabar faqat bir marta qayta ishlanadi (xatolar bo'lmasa). IoT kabi ba'zi tizimlarda voqealar juda yuqori hajmda yutilishi kerak.

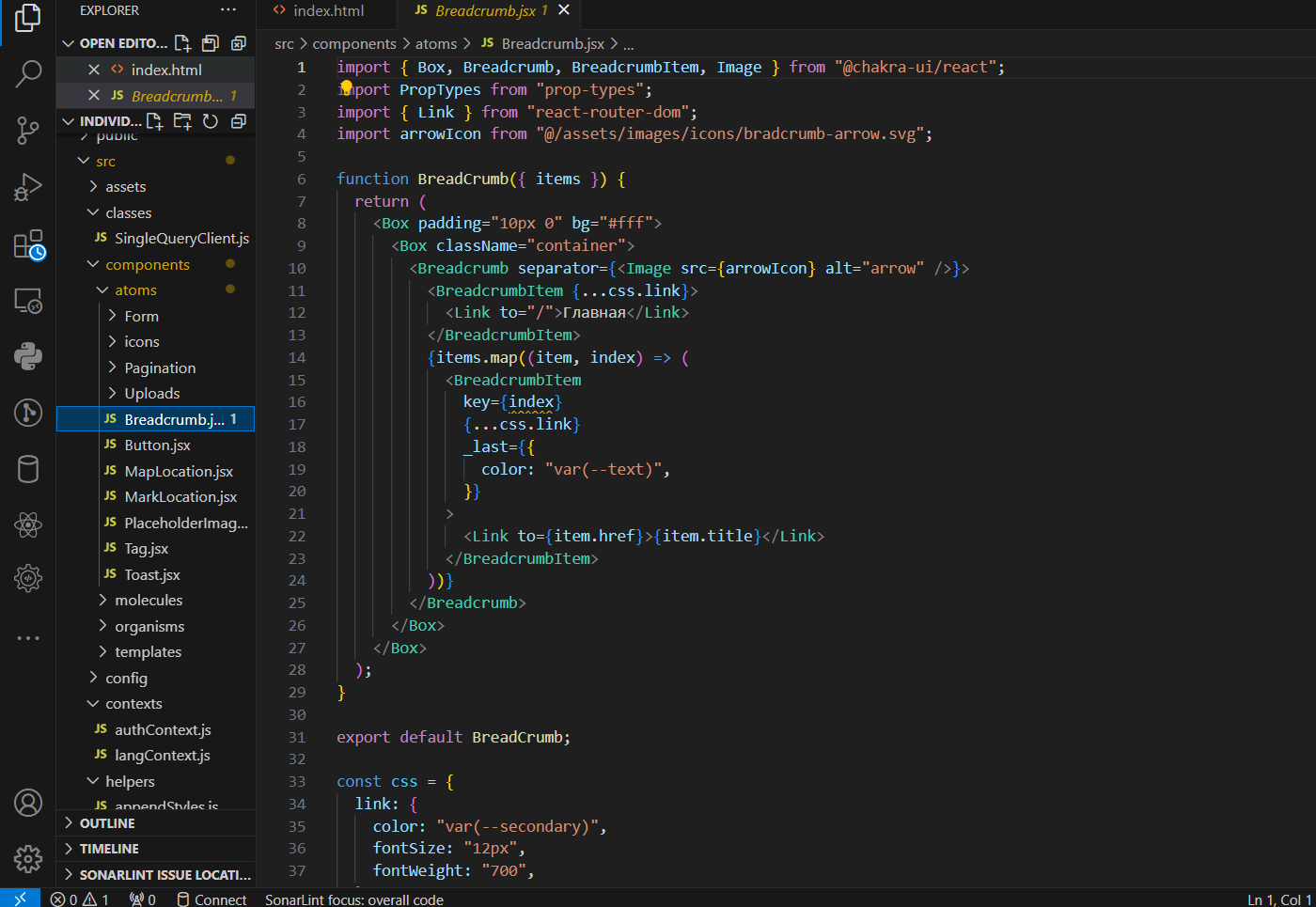
[Voqealarga asoslangan arxitektura nashr qilish/obuna boʻlish](https://learn.microsoft.com/en-us/azure/architecture/patterns/publisher-subscriber) (shuningdek, pub/sub deb ham ataladi ) modeli yoki voqealar oqimi modelidan foydalanishi mumkin .

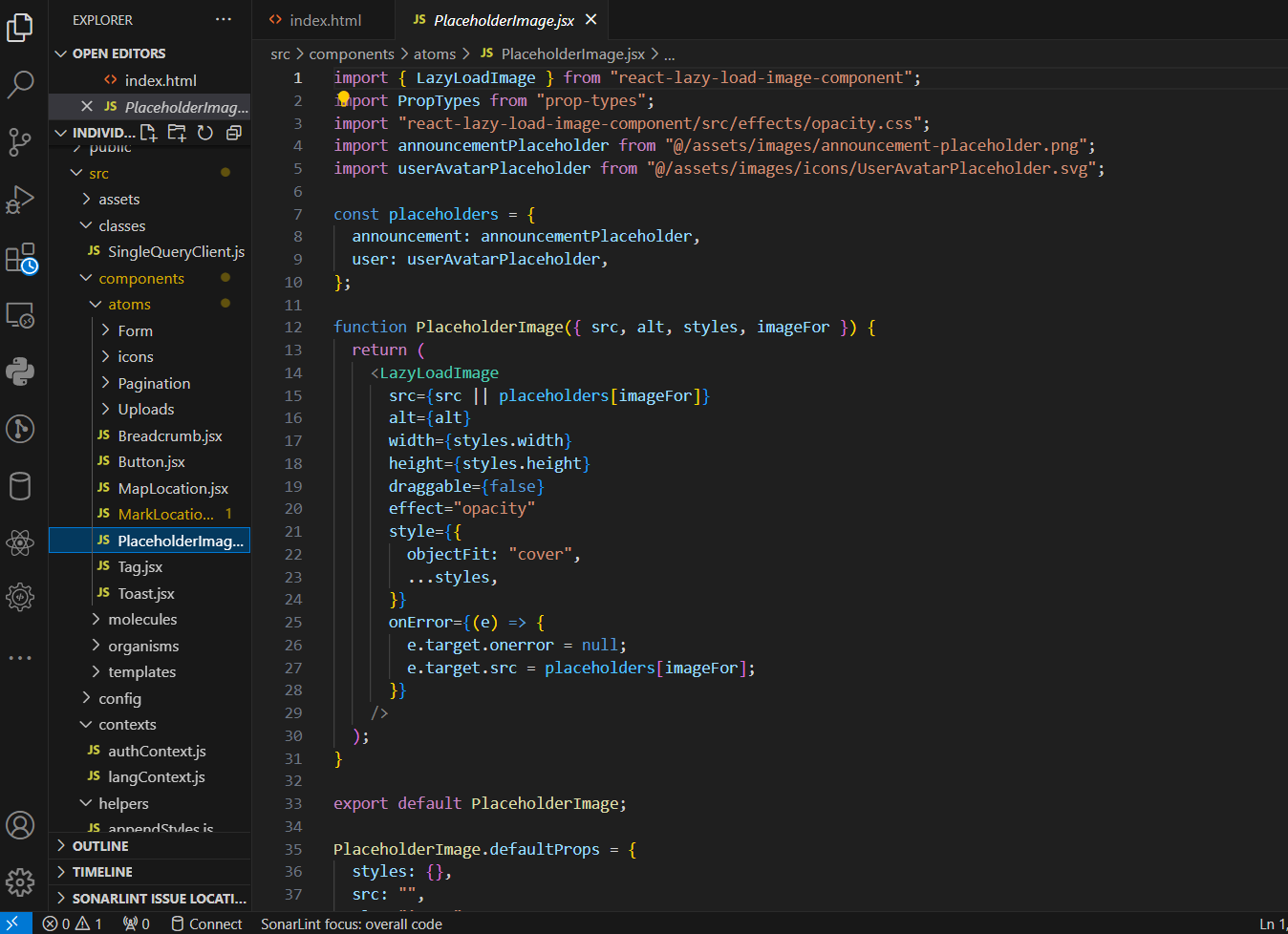
**Pub/sub** : xabar almashish infratuzilmasi obunalarni kuzatib boradi. Tadbir nashr etilganda, u voqeani har bir obunachiga yuboradi. Hodisa qabul qilingandan keyin uni qayta ko‘rib bo‘lmaydi va yangi obunachilar voqeani ko‘rmaydilar.

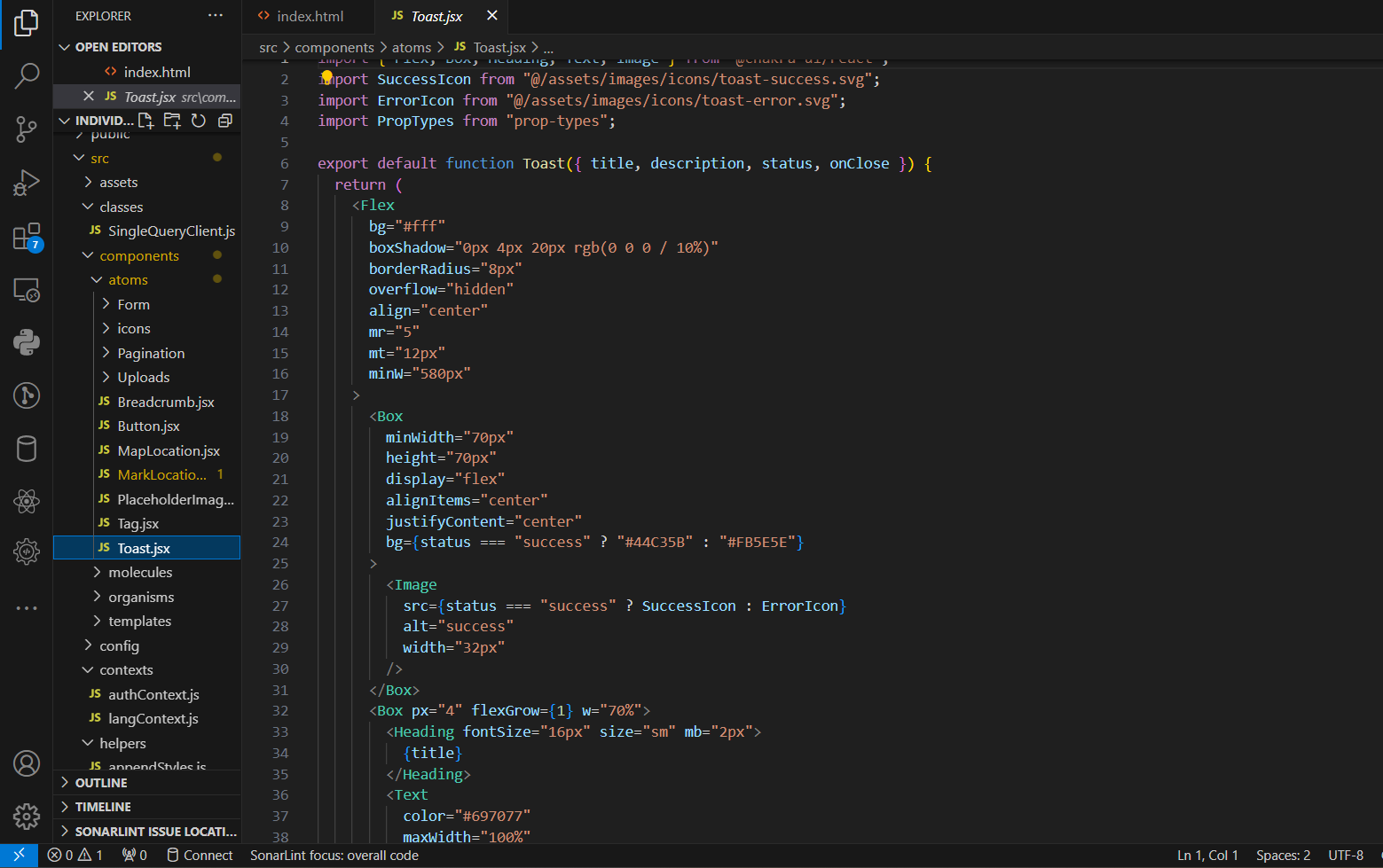


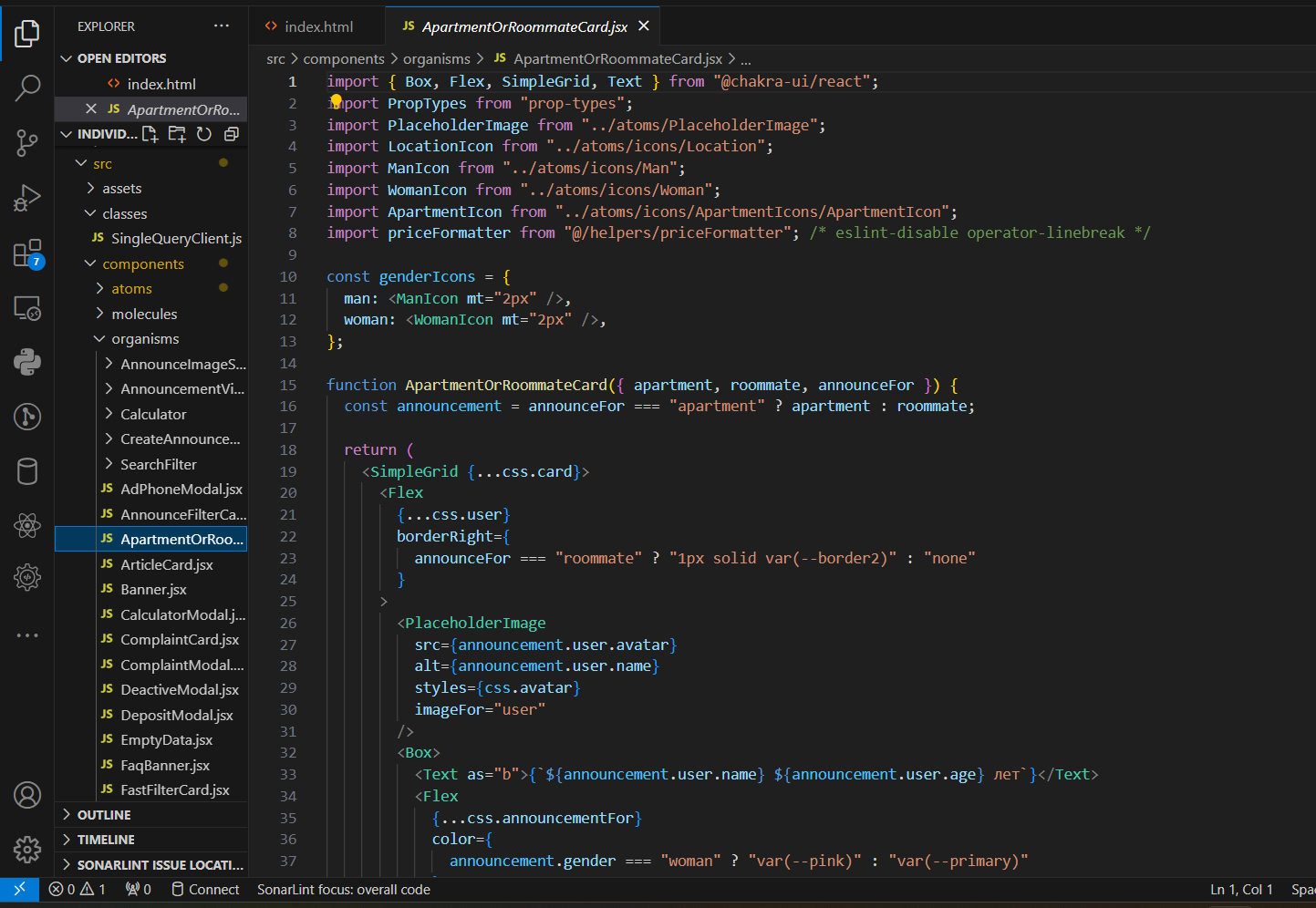
2-rasm. **eCommerce Event-Driven Architecture. Even- driven electron xarid arxitekturasi**

**Voqealar oqimi** : Voqealar jurnalga yoziladi. Voqealar qat'iy buyurtma qilingan (bo'lim ichida) va bardoshli. Mijozlar oqimga obuna bo‘lmaydi, aksincha, mijoz oqimning istalgan qismidan o‘qishi mumkin. Mijoz oqimdagi o'z pozitsiyasini oshirish uchun javobgardir. Bu shuni anglatadiki, mijoz istalgan vaqtda qo'shilishi va voqealarni takrorlashi mumkin.





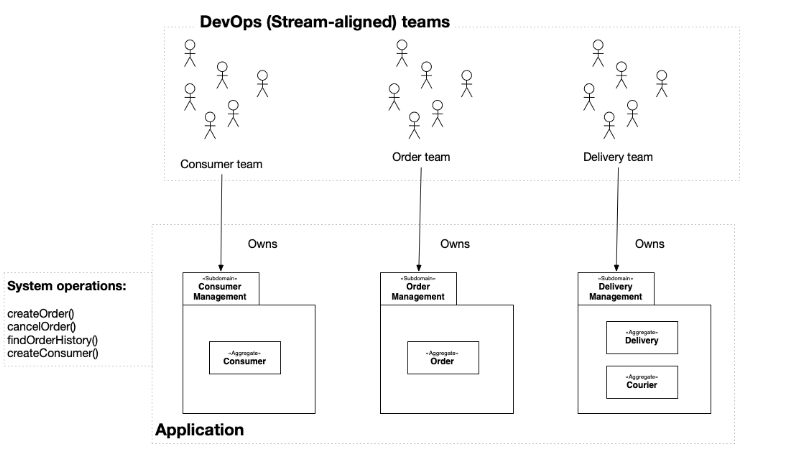




Web sahifa uchun Microservis arxitekturasni qo’llash

# **Mikroservis arxitekturasi**

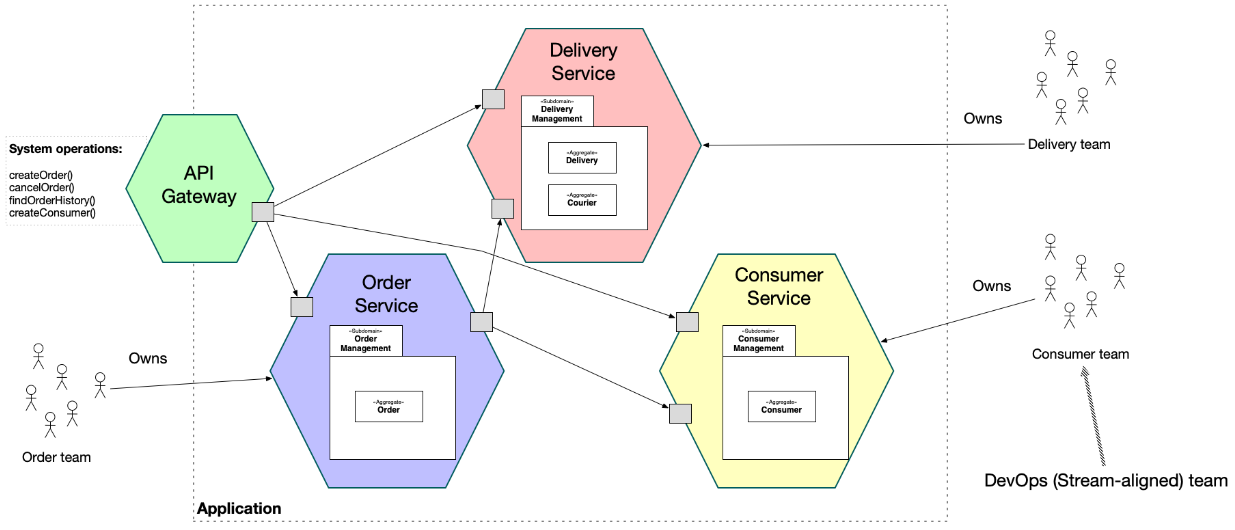
Siz biznes uchun muhim korxona ilovasini ishlab chiqmoqdasiz. Sizning biznesingiz bugungi o'zgaruvchan, noaniq, murakkab va noaniq dunyoda gullab-yashnashi uchun [DORA ko'rsatkichlari](https://microservices.io/articles/glossary#dora-metrics) bilan o'lchanadigan o'zgarishlarni tez, tez-tez va ishonchli tarzda amalga oshirishingiz kerak . Shunday qilib, sizning muhandislik tashkilotingiz [Team Topologies](https://microservices.io/tags/team%20topologies) tomonidan tavsiflanganidek, kichik, erkin bog'langan, o'zaro faoliyat funktsional guruhlarga tashkil etilgan . Har bir jamoa [DevOps qo'llanmasida](https://microservices.io/tags/devops) belgilangan DevOps amaliyotlaridan foydalangan holda dasturiy ta'minotni yetkazib beradi . Xususan, u doimiy joylashtirishni amalda qo'llaydi. Jamoa kichik, tez-tez o'zgarishlar oqimini taqdim etadi, ular avtomatlashtirilgan tarqatish quvuri orqali sinovdan o'tkaziladi va ishlab chiqarishga joylashtiriladi.



Jamoa bir yoki bir nechta subdomenlar uchun javobgardir. Subdomain - bu biznes imkoniyatlarining bir qismi, ya'ni biznes qobiliyatining amalga oshirilishi mumkin bo'lgan modeli. U biznes qoidalarini amalga oshiradigan biznes sub'ektlari (aka DDD agregatlari) va tashqi dunyo bilan aloqa qiladigan adapterlardan iborat biznes mantig'idan iborat. Masalan, Java-ga asoslangan subdomain JAR fayliga kompilyatsiya qilingan paketlarga ajratilgan sinflardan iborat.

Subdomenlar (tizim) operatsiyalar to'plamidan iborat bo'lgan ilovaning xatti-harakatlarini amalga oshiradi. Operatsiya uchta usuldan biri bilan chaqiriladi: mijozlarning sinxron va asinxron so'rovlari; boshqa ilovalar va xizmatlar tomonidan nashr etilgan voqealar; va vaqt o'tishi. U mutatsiyaga uchragan va bir yoki bir nechta subdomendagi tadbirkorlik sub'ektlarini so'raydi.

Mustaqil ravishda joylashtirilishi uchun har bir xizmat odatda o'zining manba kodlari omboriga va xizmatni quradigan, sinovdan o'tkazadigan va tarqatadigan o'z tarqatish quvuriga ega.

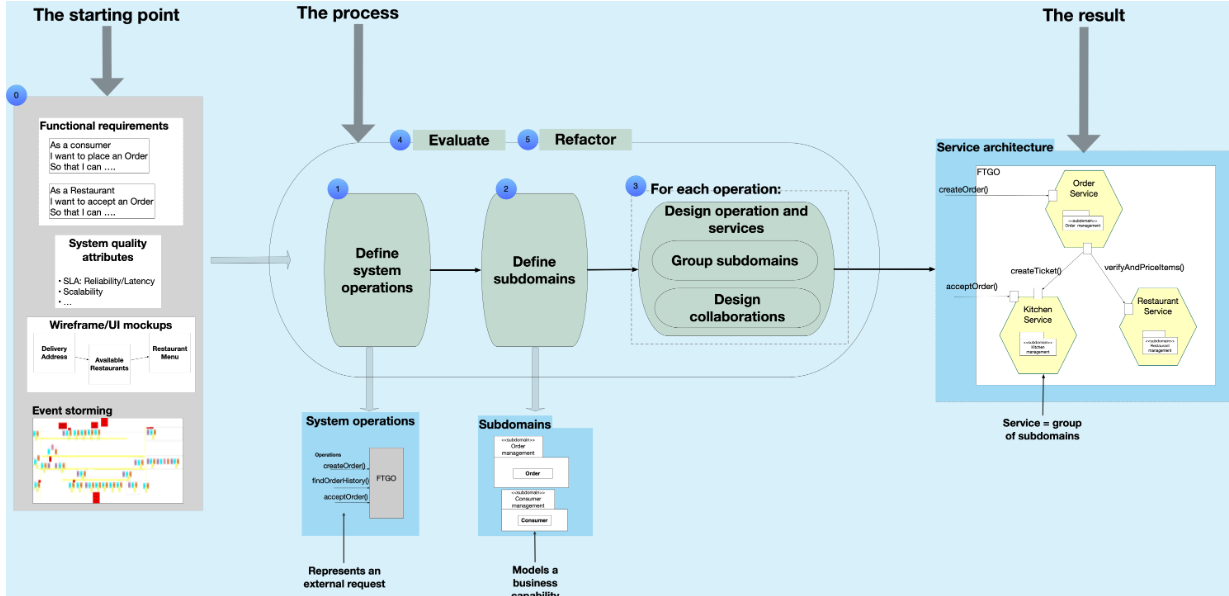


### **Muammolar**

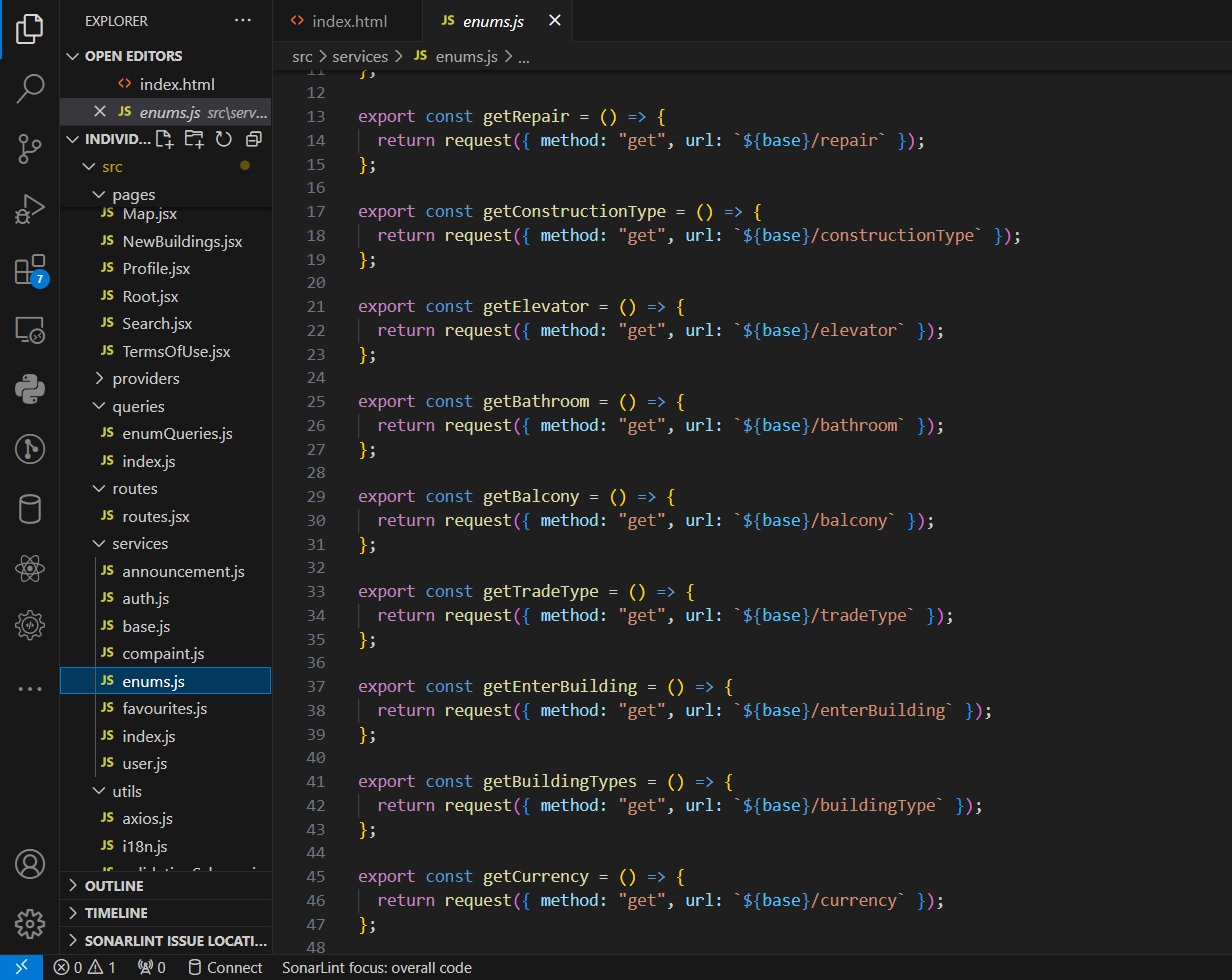
Arxitekturani loyihalashda siz hal qilishingiz kerak bo'lgan ko'plab muammolar mavjud.

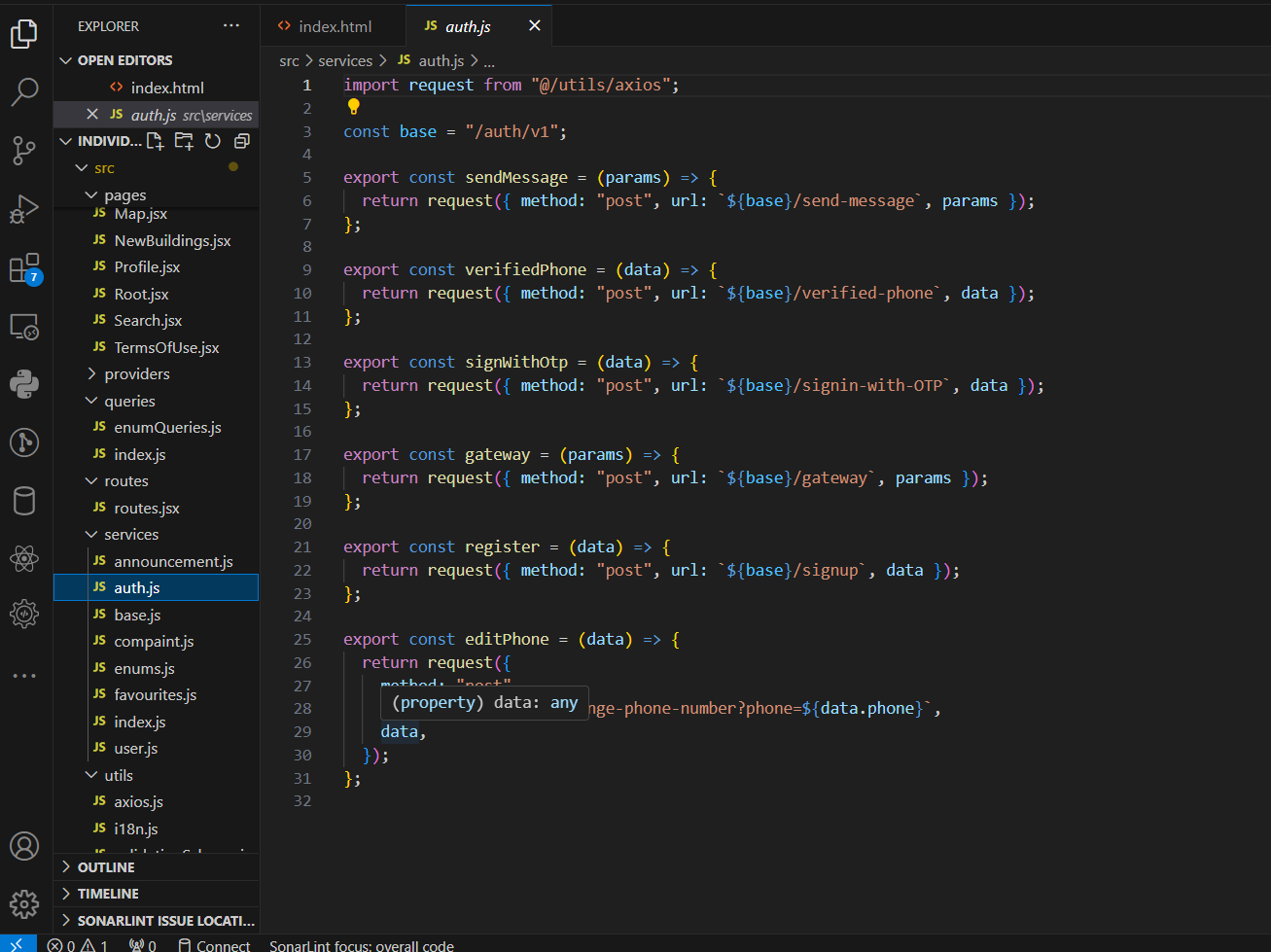
#### **Assemblage bilan yaxshi (monolitik yoki mikroservis) arxitekturasini loyihalash**

Siz hal qilishingiz kerak bo'lgan ikkita asosiy muammo mavjud. Birinchi masala - monolit yoki mikroservis arxitekturasidan foydalanish. Va keyin, agar siz mikroservis arxitekturasidan foydalanishni tanlasangiz, keyingi asosiy vazifa yaxshi [xizmat arxitekturasini](https://microservices.io/post/architecture/2023/09/19/assemblage-part-3-whats-a-service-architecture.html) aniqlashdir . Siz yuzaga kelishi mumkin bo'lgan kamchiliklardan qochishingiz (yoki hech bo'lmaganda minimallashtirishingiz) kerak: murakkab, samarasiz shovqinlar; murakkab yakuniy izchil operatsiyalar; va qattiq ish vaqtini ulash.



Mikroservislar mustaqil ravishda joylashtirilganligi sababli, xatolarni tuzatish va funksiya relizlarini boshqarish osonroq. Siz butun ilovani qayta joylashtirmasdan xizmatni yangilashingiz va biror narsa noto'g'ri bo'lsa, yangilanishni orqaga qaytarishingiz mumkin. Ko'pgina an'anaviy ilovalarda, agar dasturning bir qismida xato topilsa, u butun chiqarish jarayonini bloklashi mumkin. Xatolarni tuzatish integratsiyalashuvi, sinovdan o'tkazilishi va nashr etilishi uchun yangi xususiyatlar to'xtatilishi mumkin.





**P2P (Peer-to-Peer Process)**

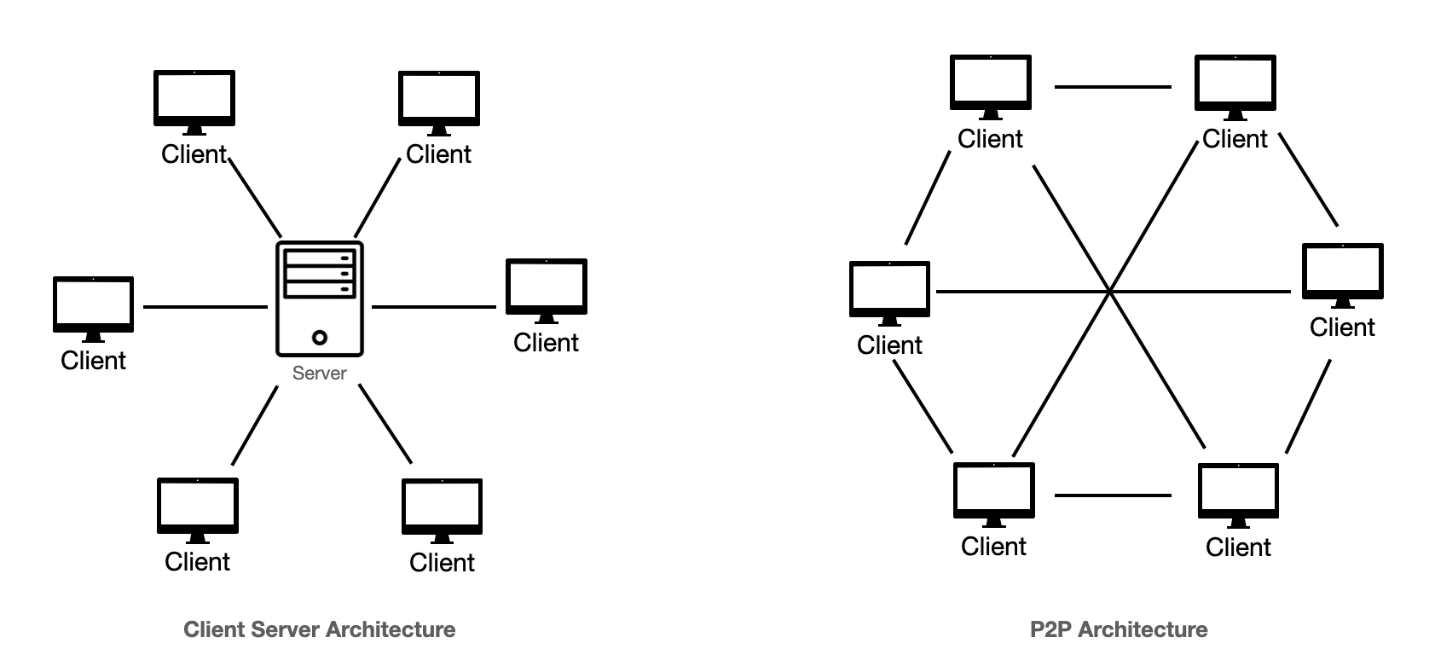
Peer-to-peer tarmog'i - bu oddiy kompyuterlar tarmog'i. U birinchi marta 1970-yillarning oxirida paydo bo'lgan. Bu erda har bir kompyuter shakllangan tarmoq ichida fayllarni almashish uchun tugun vazifasini bajaradi. Bu erda har bir tugun server vazifasini bajaradi va shuning uchun tarmoqda markaziy server mavjud emas. Bu katta hajmdagi ma'lumotlarni almashish imkonini beradi. Vazifalar tugunlar o'rtasida teng taqsimlanadi. Tarmoqqa ulangan har bir tugun teng ish yukini taqsimlaydi. Tarmoq ishlamay qolishi uchun barcha tugunlar alohida ishlamay qolishi kerak. Buning sababi shundaki, har bir tugun mustaqil ishlaydi.

**P2P tarmoqlari tarixi**

P2P rivojlanishidan oldin, USENET 1979 yilda paydo bo'lgan. Tarmoq foydalanuvchilarga xabarlarni o'qish va joylashtirish imkonini berdi. Bugungi kunda biz foydalanadigan forumlardan farqli o'laroq, unda markaziy server yo'q edi. U yangi xabarlarni tugunning barcha serverlariga nusxalash uchun ishlatiladi.

1980-yillarda P2P tarmoqlaridan birinchi marta foydalanish shaxsiy kompyuterlar paydo bo'lgandan keyin sodir bo'ldi.

1988 yil avgust oyida internet-relay suhbati matn va chat almashish uchun qurilgan birinchi P2P tarmog'i edi.



1999 yil iyun oyida fayl almashish uchun P2P dasturi bo'lgan Napster ishlab chiqildi. U audio fayllarni almashish uchun ham ishlatilishi mumkin. Ushbu dasturiy ta'minot fayllarni noqonuniy almashish tufayli yopildi. Ammo tarmoqni almashish, ya'ni P2P tushunchasi mashhur bo'ldi.

2000 yil iyun oyida Gnutella birinchi markazlashtirilmagan P2P fayl almashish tarmog'i edi. Bu foydalanuvchilarga boshqa foydalanuvchilarning kompyuterlaridagi fayllarga belgilangan papka orqali kirish imkonini berdi.

**P2P tarmoq arxitekturasi**

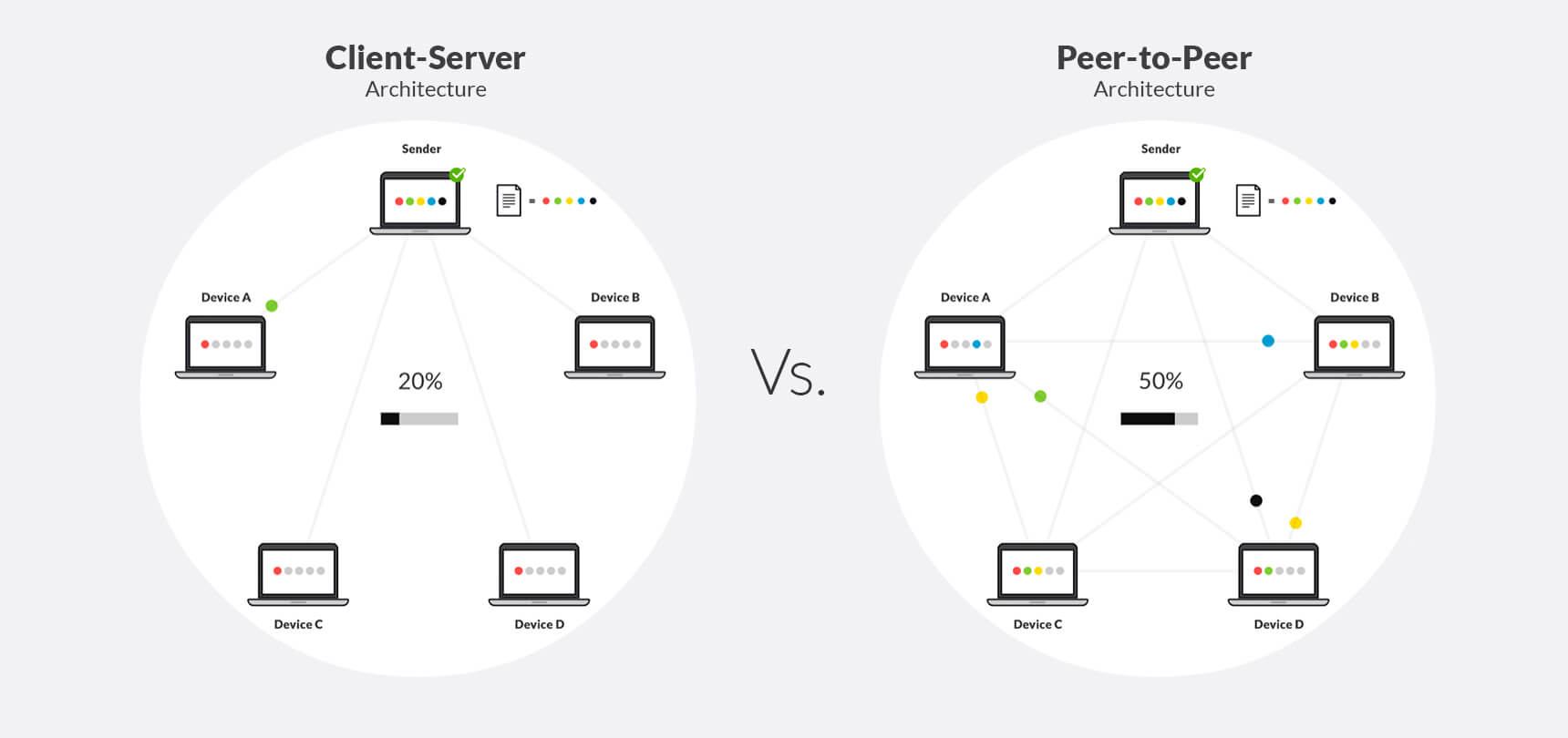
P2P tarmoq arxitekturasida kompyuterlar fayllarni almashish, internet va printerlarga kirish uchun ishchi guruhda bir-biri bilan bog'lanadi.

Tarmoqdagi har bir kompyuter bir xil mas'uliyat va imkoniyatlarga ega.

Tarmoqdagi har bir qurilma mijoz va server vazifasini bajaradi.

Arxitektura turar-joylarda, kichik ofislarda yoki kichik kompaniyalarda foydalidir, bu erda har bir kompyuter mustaqil ish stantsiyasi sifatida ishlaydi va ma'lumotlarni qattiq diskida saqlaydi.

Tarmoqdagi har bir kompyuter tarmoqdagi boshqa kompyuterlar bilan ma'lumotlarni almashish imkoniyatiga ega. Arxitektura odatda 12 yoki undan ortiq kompyuterdan iborat ishchi guruhlardan iborat.



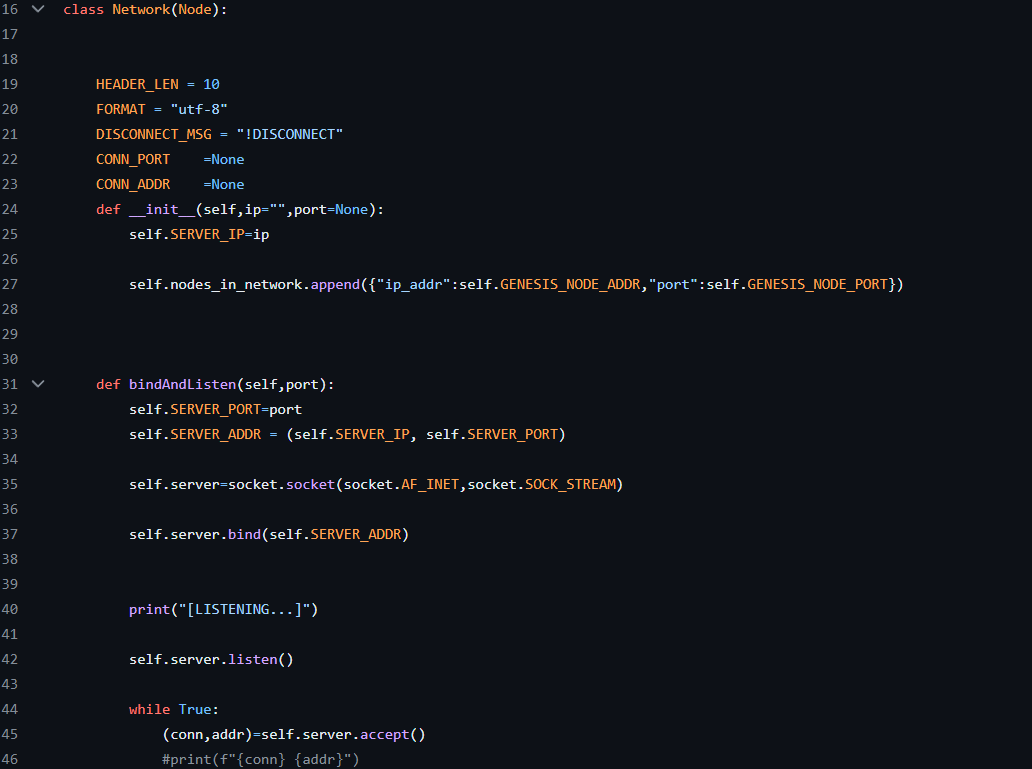
P2P ilovalari tarmoqning betarafligi haqidagi bahs-munozaralarning asosiy masalalaridan biri bo'lib, bu tamoyil Internet mazmuni, formati, texnologiyalari, uskunalari yoki aloqa usullariga hech qanday cheklovlar qo'yilmasligini qo'llab-quvvatlaydi. P2P tarafdorlarining ta'kidlashicha, hukumatlar va yirik Internet-provayderlar tarmoq strukturasini mijoz/server arxitekturasiga yo'naltirish orqali Internetdan foydalanish va kontentni nazorat qilishga harakat qilmoqda. Bu Internetga kirishni xohlaydigan jismoniy shaxslar va kichik nashriyotlar uchun moliyaviy to'siqlarni o'rnatadi va katta fayllarni almashishda samarasizlikni keltirib chiqaradi.

Tarkibi bo'lmagan P2P kompyuter tarmog'i arxitekturasining uchta modeli mavjud:

* Sof P2P
* Gibrid P2P
* Markazlashtirilgan P2P

Strukturaviy P2P kompyuter tarmog'i arxitekturasida ish stantsiyalari (tengdoshlar) va ba'zan resurslar ham ma'lum mezonlar va algoritmlarga muvofiq tashkil etilgan. Bu o'ziga xos topologiyalar va xususiyatlarga ega qoplamalarga olib keladi.

P2P tarmog'ining ba'zi afzalliklari va zaif tomonlarini baholash mijoz/server arxitekturasi bilan taqqoslashni o'z ichiga oladi. P2P tarmoqlari o'tkazish qobiliyati, saqlash maydoni va ishlov berish kuchi kabi resurslarga ega mijozlarga ega. Har bir tugun orqali tizimga ko'proq talab qo'yilsa, butun tizimning imkoniyatlari ortadi. (Bu tizim xavfsizligi va fayllarni tekshirish mexanizmlarining ulkan o'sishini hisobga olib, aksariyat P2P tarmoqlarini deyarli har qanday hujumga chidamli qiladi.) Taqqoslash uchun, odatiy mijoz/server tarmog'i talablarni baham ko'radi, lekin resurslarni emas. Tizimga qo'shimcha mijozlar qo'shilsa, har biriga kamroq resurslar mavjud bo'ladi.





Loyihani yaratishda turli xil arxitekturalardan foydalanish

**Model-View-Controller (MVC) arxitekturasi**

So'nggi bir necha yil ichida veb-saytlar biroz CSS-ga ega bo'lgan oddiy HTML-sahifalardan bir vaqtning o'zida minglab ishlab chiquvchilar ishlayotgan nihoyatda murakkab ilovalarga o'tdi. Ushbu murakkab veb-ilovalar bilan ishlash uchun ishlab chiquvchilar o'zlarining loyihalarini ishlab chiqish uchun turli dizayn naqshlaridan foydalanadilar, kodni kamroq murakkab va ishlashni osonlashtiradi. Ushbu naqshlarning eng mashhuri MVC, shuningdek **, Model View Controller** sifatida ham tanilgan .

**MVC HAQIDA TUSHUNCHA**

Model **-View-Controller (MVC)** ramkasi dasturni uchta asosiy mantiqiy komponentlarg ajratuvchi arxitektura/dizayn naqshidir **Model** , **Ko'rinish** va **Controller** . Har bir arxitektura komponenti dasturning o'ziga xos rivojlanish jihatlarini boshqarish uchun qurilgan. U biznes mantig'i va taqdimot qatlamini bir-biridan ajratib turadi. U an'anaviy ravishda ish stoli **grafik foydalanuvchi interfeyslari (GUI)** uchun ishlatilgan . Hozirgi vaqtda MVC kengaytiriladigan va kengaytiriladigan loyihalarni yaratish uchun eng ko'p ishlatiladigan sanoat standartidagi veb-ishlab chiqish ramkalaridan biridir. Bundan tashqari, u mobil ilovalarni loyihalash uchun ishlatiladi.

**MVC Trygve Reenskaug** tomonidan yaratilgan . Ushbu dizayn naqshining asosiy maqsadi foydalanuvchilarning katta va murakkab ma'lumotlar to'plamini boshqarish muammosini hal qilish edi, bu katta dasturni o'z maqsadiga ega bo'lgan muayyan bo'limlarga bo'lish edi.

**MVC xususiyatlari:**

**Bu biznes mantig'i, UI mantig'i va kirish mantig'ini** aniq ajratishni ta'minlaydi .

U HTML va URL manzillaringizni to'liq nazorat qilishni taklif qiladi, bu esa veb-ilovalar arxitekturasini loyihalashni osonlashtiradi.

Bu kuchli URL xaritalash komponenti bo'lib, uning yordamida biz tushunarli va qidiriladigan URL manzillariga ega ilovalarni yaratishimiz mumkin.

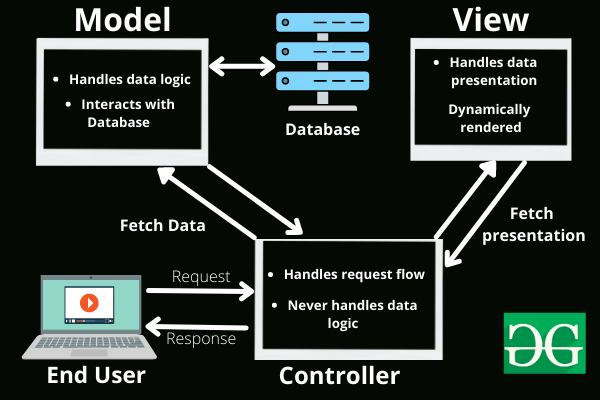
U **Test Driven Development (TDD) ni qo'llab-quvvatlaydi.**

**MVC komponentlari:**

Nazoratchi

Model

Ko'rinish

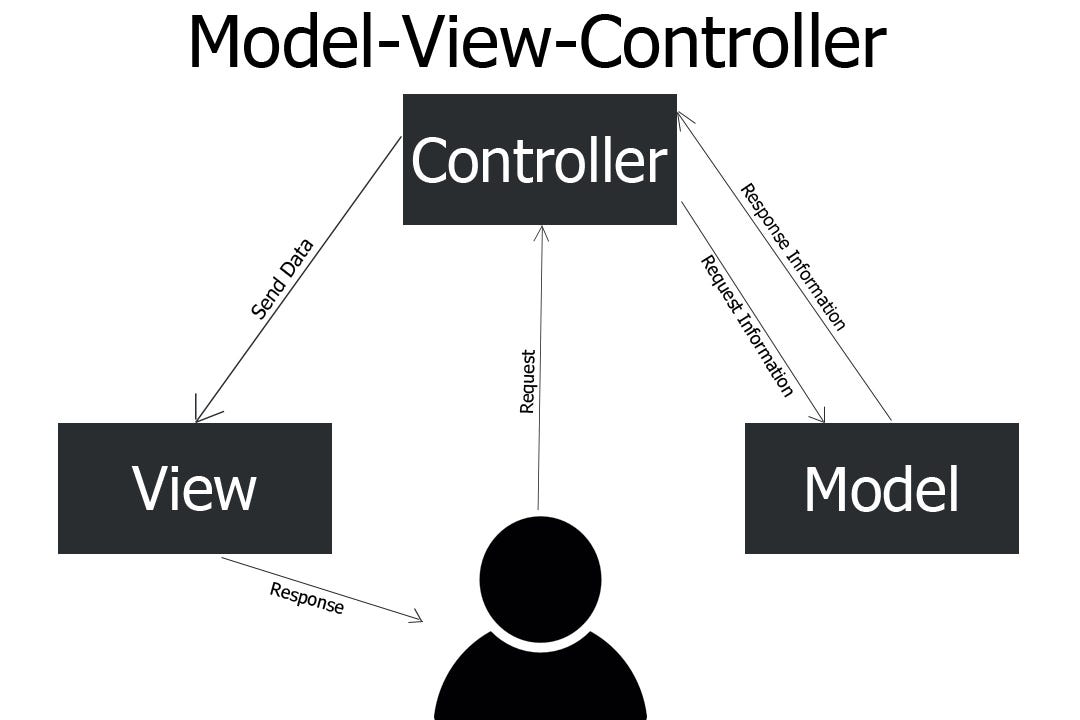


Nazoratchi ko'rinishlar va model o'rtasidagi o'zaro bog'lanishni ta'minlaydigan komponent bo'lib, u vositachi sifatida ishlaydi. Tekshiruvchi ma'lumotlar mantig'i bilan ishlash haqida tashvishlanmasligi kerak, u faqat modelga nima qilish kerakligini aytadi. U barcha biznes mantiqiy va kiruvchi so'rovlarni qayta ishlaydi,  komponenti yordamida ma'lumotlarni boshqaradi va yakuniy natijani ko'rsatish uchun

View ilovaning barcha UI mantiqlari uchun ishlatiladi. U foydalanuvchi uchun foydalanuvchi interfeysini yaratadi. Ko'rinishlar model komponenti tomonidan to'plangan ma'lumotlar orqali yaratiladi, lekin bu ma'lumotlar to'g'ridan-to'g'ri emas, balki kontroller orqali olinadi. U faqat boshqaruvchi bilan o'zaro ta'sir qiladi.

Model komponenti foydalanuvchi ishlaydigan barcha ma'lumotlar bilan bog'liq mantiqqa mos keladi **.** Bu View va Controller komponentlari oʻrtasida uzatilayotgan maʼlumotlarni yoki biznes mantigʻiga oid boshqa maʼlumotlarni koʻrsatishi mumkin. U ma'lumotlar bazasidan ma'lumotlarni qo'shishi yoki olishi mumkin. U boshqaruvchining so'roviga javob beradi, chunki boshqaruvchi ma'lumotlar bazasi bilan o'zi o'zaro ta'sir qila olmaydi. Model ma'lumotlar bazasi bilan o'zaro ta'sir qiladi va kerakli ma'lumotlarni tekshirgichga qaytaradi.

Tasavvur qilaylik, oxirgi foydalanuvchi sinfda o'qiyotgan talabalar ro'yxatini olish uchun serverga so'rov yuboradi. Keyin server ushbu so'rovni talabalar bilan ishlaydigan o'sha nazoratchiga yuboradi. Keyin bu nazoratchi talabalarni boshqaradigan modeldan sinfda o'qiyotgan barcha talabalar ro'yxatini qaytarishni so'raydi.



Model barcha talabalar ro'yxati uchun ma'lumotlar bazasini so'raydi va keyin bu ro'yxatni tekshirgichga qaytaradi. Agar modeldan javob muvaffaqiyatli bo'lsa, boshqaruvchi talabalar bilan bog'liq ko'rinishdan talabalar ro'yxati taqdimotini qaytarishni so'raydi. Ushbu ko'rinish talabalar ro'yxatini tekshirgichdan oladi va ro'yxatni brauzer tomonidan ishlatilishi mumkin bo'lgan HTMLga beradi.

Keyin boshqaruvchi ushbu taqdimotni oladi va uni foydalanuvchiga qaytaradi. Shunday qilib, so'rov tugaydi. Agar ilgari model xatolikni qaytargan bo'lsa, boshqaruvchi xatolarni ko'rib chiqadigan ko'rinishdan ushbu xato uchun taqdimotni ko'rsatishni so'rash orqali ushbu xatoni hal qiladi. Keyin o'sha xato taqdimoti talabalar ro'yxati taqdimoti o'rniga foydalanuvchiga qaytariladi.

Yuqoridagi misoldan ko'rinib turibdiki, model barcha ma'lumotlarni qayta ishlaydi. Ko'rinish barcha taqdimotlarni boshqaradi va boshqaruvchi faqat model va ko'rinishni nima qilish kerakligini aytadi. Bu MVC ramkasining asosiy arxitekturasi va ishlashi.



Model barcha talabalar ro'yxati uchun ma'lumotlar bazasini so'raydi va keyin bu ro'yxatni tekshirgichga qaytaradi. Agar modeldan javob muvaffaqiyatli bo'lsa, boshqaruvchi talabalar bilan bog'liq ko'rinishdan talabalar ro'yxati taqdimotini qaytarishni so'raydi. Ushbu ko'rinish talabalar ro'yxatini tekshirgichdan oladi va ro'yxatni brauzer tomonidan ishlatilishi mumkin bo'lgan HTMLga beradi.

Keyin boshqaruvchi ushbu taqdimotni oladi va uni foydalanuvchiga qaytaradi. Shunday qilib, so'rov tugaydi. Agar ilgari model xatolikni qaytargan bo'lsa, boshqaruvchi xatolarni ko'rib chiqadigan ko'rinishdan ushbu xato uchun taqdimotni ko'rsatishni so'rash orqali ushbu xatoni hal qiladi. Keyin o'sha xato taqdimoti talabalar ro'yxati taqdimoti o'rniga foydalanuvchiga qaytariladi.

Yuqoridagi misoldan ko'rinib turibdiki, model barcha ma'lumotlarni qayta ishlaydi. Ko'rinish barcha taqdimotlarni boshqaradi va boshqaruvchi faqat model va ko'rinishni nima qilish kerakligini aytadi. Bu MVC ramkasining asosiy arxitekturasi va ishlashi.

MVC arxitektura namunasi bizga quyidagi dizayn tamoyillariga amal qilish imkonini beradi:  
1. **Bo'lin va zabt et.** Uch komponent bir oz mustaqil ravishda ishlab chiqilishi mumkin.  
2. **Birdamlikni oshirish.** Ko'rinish va boshqaruvchi bitta UI qatlamida birga bo'lganidan ko'ra, komponentlar qatlamning mustahkam uyg'unligiga ega.  
3. **Ulanishni kamaytiring.** Uch komponent o'rtasidagi aloqa kanallari minimal va topish oson.  
4. **Qayta foydalanishni oshirish.** Ko'rinish va kontroller odatda har xil turdagi UI boshqaruvlari uchun qayta foydalanish mumkin bo'lgan komponentlardan keng foydalanadi. Biroq, UI ilovaga xos bo'ladi, shuning uchun uni qayta ishlatish oson bo'lmaydi.  
5. **Moslashuvchanlik uchun dizayn.** Ko'rinishni, boshqaruvchini yoki ikkalasini o'zgartirish orqali foydalanuvchi interfeysini o'zgartirish odatda juda oson.

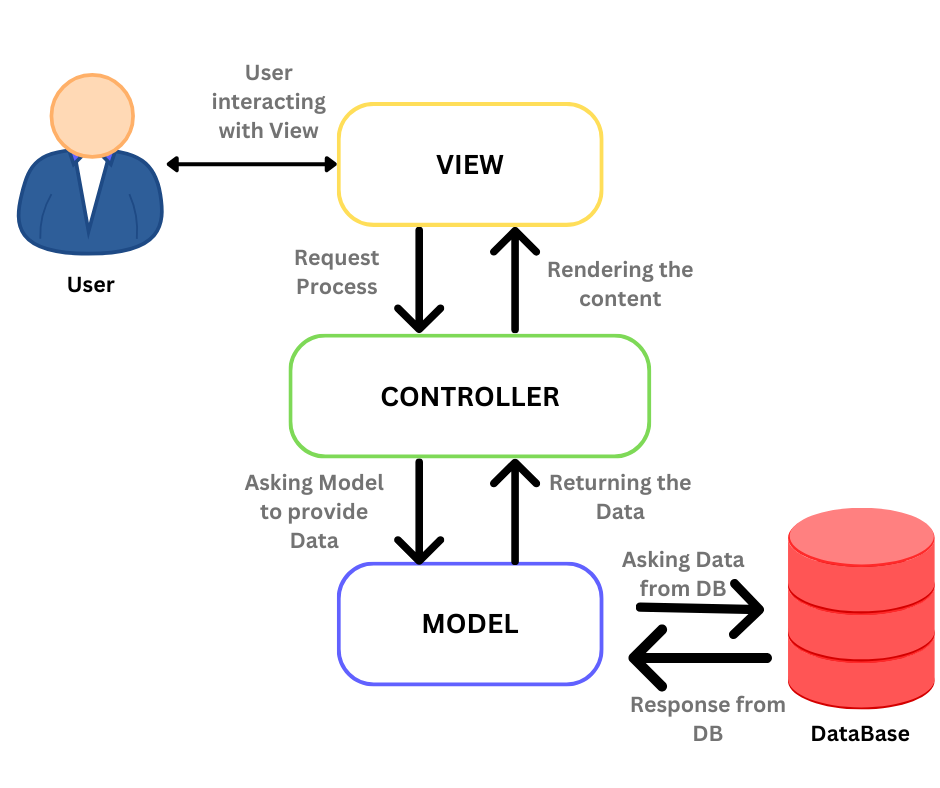
**MVC ning afzalliklari:**

Kodlarni saqlash oson va ularni osongina uzaytirish mumkin.

MVC **modeli** komponenti alohida tekshirilishi mumkin.

MVC komponentlari bir vaqtning o'zida ishlab chiqilishi mumkin.

Ilovani uchta birlikka bo'lish orqali murakkablikni kamaytiradi. **Model, ko'rinish va boshqaruvchi.**



U **Test Driven Development (TDD) ni qo'llab-quvvatlaydi.**

U veb-dizaynerlar va ishlab chiquvchilarning katta guruhlari tomonidan qo'llab-quvvatlanadigan veb-ilovalar uchun yaxshi ishlaydi.

Ushbu arxitektura komponentlarni mustaqil ravishda sinab ko'rishga yordam beradi, chunki barcha sinflar va ob'ektlar bir-biridan mustaqildir

**Qidiruv tizimini optimallashtirish (SEO)** qulay.

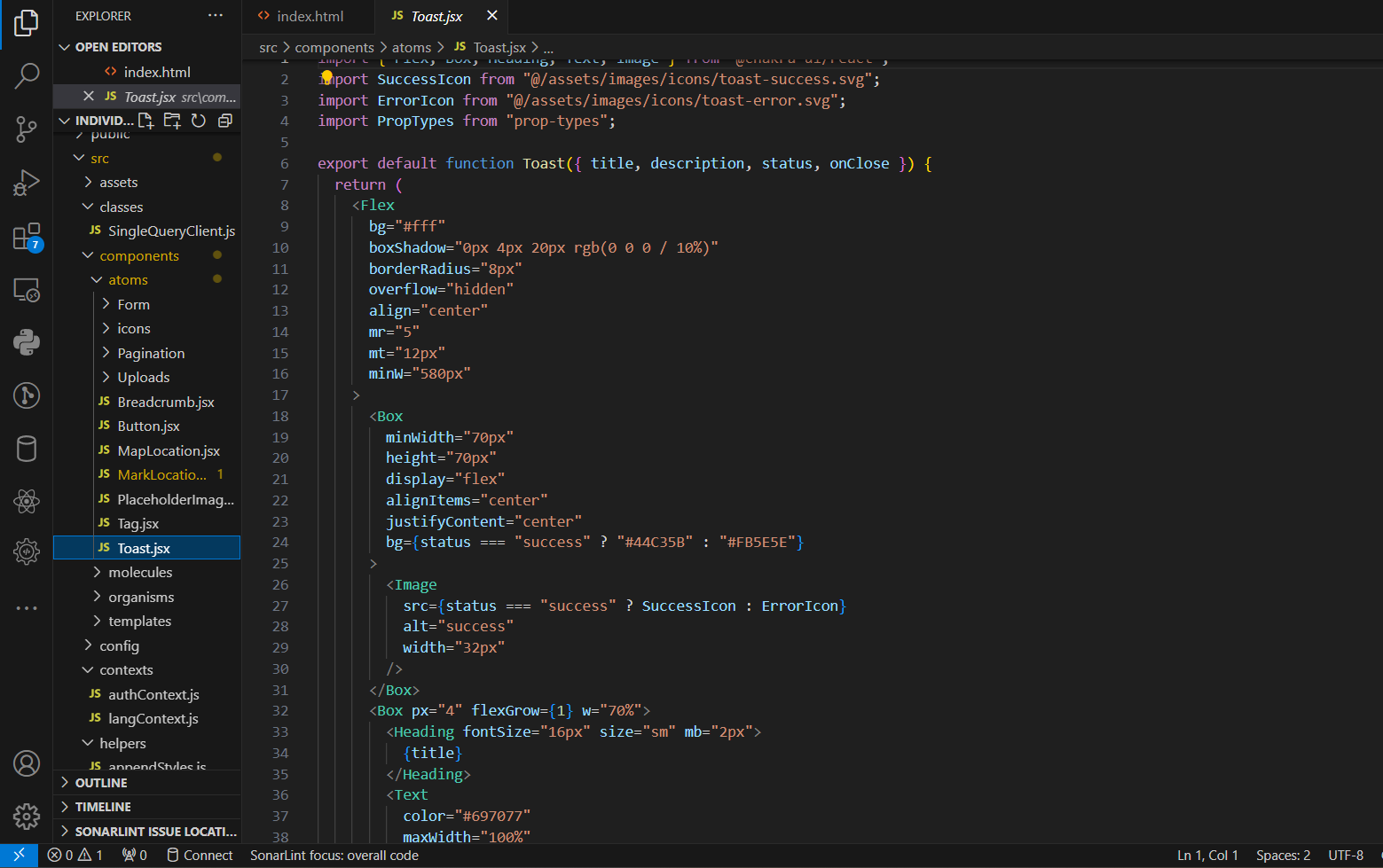
**MVC ning kamchiliklari:**

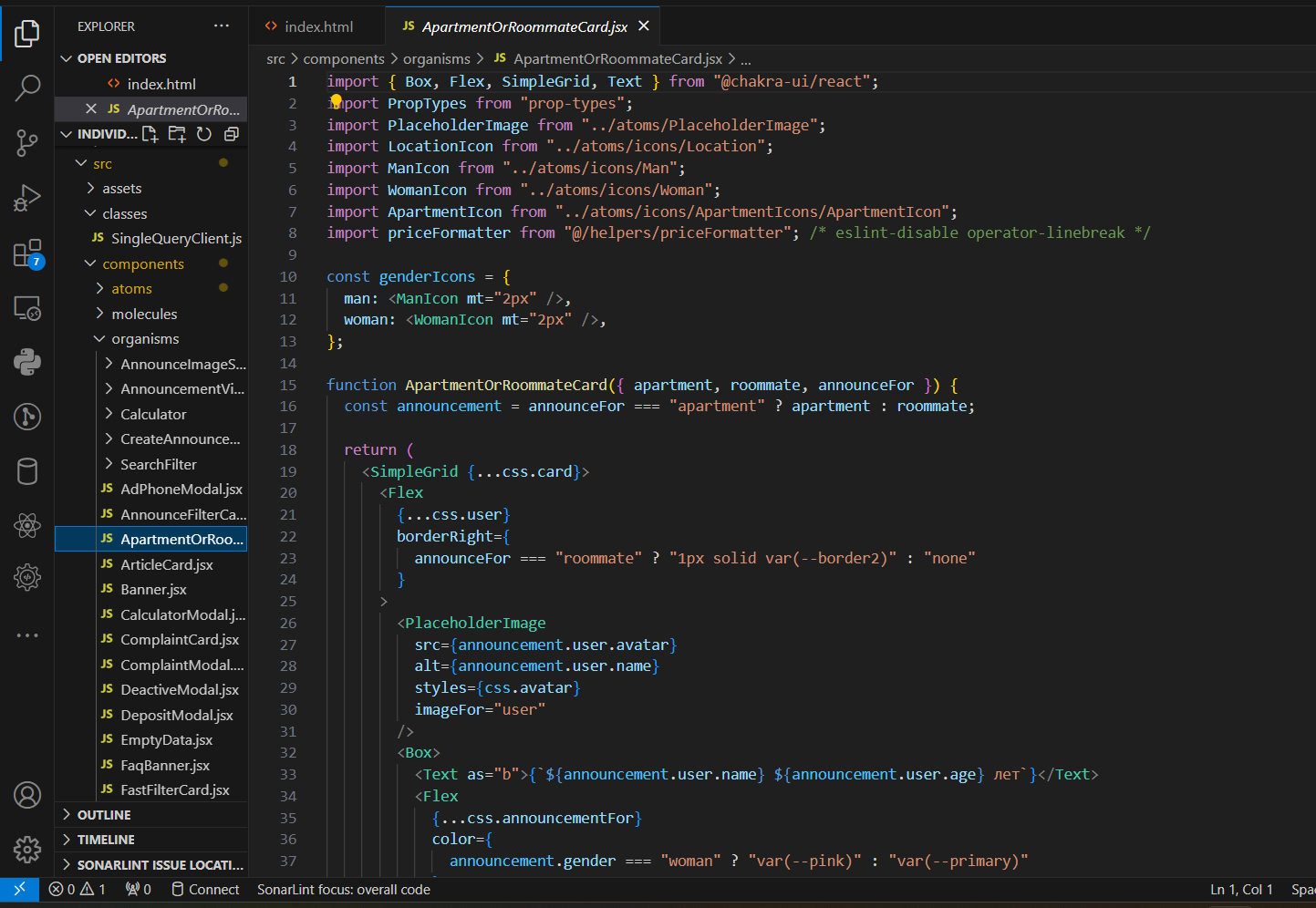
Ushbu modelni o'qish, o'zgartirish, sinab ko'rish va qayta ishlatish qiyin

Kichik ilovalarni yaratish uchun mos emas.

Ko'rinishda ma'lumotlarga kirishning samarasizligi.

Ramka navigatsiyasi murakkab bo'lishi mumkin, chunki u foydalanuvchilardan MVC parchalanish mezonlariga moslashishni talab qiladigan yangi abstraksiya qatlamlarini taqdim etadi.





## Mashhur MVC Frameworks

## Ruby on Rails

[Ruby on Rails](https://rubyonrails.org/) - bu Devid Xaynemeyer Hansson tomonidan yaratilgan Ruby dasturlash tilida yozilgan ochiq manbali veb-ishlab chiqish tizimi. U murakkab veb-ilovalarni ishlab chiqish va saqlashni osonlashtiradigan bir qator xususiyatlarni taqdim etadi. U ob'ektga yo'naltirilgan dasturlash tili Ruby ustiga qurilgan va model-view-controller (MVC) arxitektura naqshiga amal qiladi.

Raillar MVC-ga asoslangan, ya'ni u barcha UI komponentlarini uch xil qismga ajratadi: Model, Ko'rinish va Controller.

**Django**

[Django](https://www.djangoproject.com/) - bu Python-da yozilgan bepul va ochiq manbali veb-ilovalar tizimi. U MVC arxitekturasiga amal qiladi va murakkab, ma'lumotlar bazasiga asoslangan veb-saytlarni jadal rivojlantirish uchun birlashtirilgan komponentlar to'plamini taqdim etadi. Django Django Software Foundation (DSF) tomonidan ishlab chiqilgan va u Apache litsenziyasi, 2.0 versiyasi ostida litsenziyalangan.

**Laravel**

[Laravel](https://laravel.com/) veb-ishlab chiqish uchun bepul, ochiq manbali PHP tizimidir. U Teylor Otvell tomonidan yaratilgan va asosan PHP dasturlash tilidan foydalangan holda model-view-controller (MVC) arxitektura naqshiga asosan veb-ilovalarni ishlab chiqish va loyihalash uchun mo'ljallangan. Laravel ishlab chiqish jarayonini undan foydalanadigan ishlab chiquvchilar uchun yoqimli holga keltirishni maqsad qilgan.

**Bahor MVC**

Spring [MVC](https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html) ramkasi - bu korporativ darajadagi MVC ilovalarini ishlab chiqish uchun mustahkam xususiyatlar to'plamini ta'minlovchi ochiq manbali Java veb-ilovalar tizimi. U ishlatish uchun qulay, yuqori darajada sinovdan o'tkazilishi va kengaytirilishi uchun mo'ljallangan.

Spring MVC sizga Java-da o'rta darajani (nazoratchi) tashkil etuvchi sanitariya-tesisat kodi haqida tashvishlanmasdan veb-ilovalarni yaratishga imkon beradi. Bularning barchasini siz uchun hal qiladi va sizga ilova mantig'iga e'tiboringizni erkin qo'yadi. Shuningdek, u xalqarolashtirish (i18n) va mahalliylashtirishni (l10n) qo'llab-quvvatlaydi, shuning uchun kontentni bir nechta tillarda osongina ko'rsatishingiz mumkin.

Xulosa

Tekin Market, O'zbekistonda joylashgan onlayn supermarket bo'lib, xaridorlarga keng turdagi mahsulotlarni uyidan chiqmasdan xarid qilish imkonini beradi. Ularning veb-sayti toza dizayni, qulay navigatsiyasi va xarid qilish jarayonini osonlashtiradigan turli funksiyalari bilan ajralib turadi.

**Veb-saytning ijobiy jihatlari:**

* **Keng turdagi mahsulotlar:** Oziq-ovqat, ichimliklar, go'sht va baliq mahsulotlari, sut mahsulotlari, non va pishiriqlar, uy-ro'zg'or buyumlari va boshqa ko'plab mahsulotlarni topish mumkin.
* **Qulay navigatsiya:** Mahsulotlar kategoriyalar bo'yicha guruhlangan, bu esa kerakli mahsulotni tezda topishga yordam beradi. Shuningdek, qidiruv tizimi orqali ham mahsulotni qidirish mumkin.
* **Oson xarid qilish jarayoni:** Xarid qilish jarayoni bir necha bosqichdan iborat bo'lib, sodda va tushunarli.
* **Yetkazib berish xizmati:** Tekin Market Toshkent shahri bo'ylab yetkazib berish xizmatini taklif etadi.

**Veb-saytni yaxshilash bo'yicha takliflar:**

* **Mahsulotlar haqida batafsil ma'lumot:** Ba'zi mahsulotlar haqida yetarli ma'lumot berilmagan. Masalan, tarkibi, ozuqaviy qiymati, ishlab chiqaruvchisi va boshqa ma'lumotlarni qo'shish mumkin.
* **Suratlar sifatini yaxshilash:** Ba'zi mahsulotlarning suratlari yetarlicha sifatli emas.
* **To'lov usullarini ko'paytirish:** Hozirda faqat naqd pul va plastik karta orqali to'lov qilish mumkin. Boshqa to'lov usullarini qo'shish (masalan, elektron hamyonlar) xaridorlar uchun qulaylik yaratishi mumkin.

Tekin Market veb-sayti xaridorlar uchun qulay va foydali xizmat ko'rsatishga qaratilgan. Veb-saytning yuqorida qayd etilgan kamchiliklarini bartaraf etish orqali uning yanada samaraliroq ishlashiga erishish mumkin.

# **Foydalanilgan adabiyotlar va web sahifalar**

1. <https://www.wix.com/>
2. <https://www.simplilearn.com/best-javascript-books-to-read-article>
3. <https://devopscube.com/>
4. <https://www.guru99.com/>
5. <https://bard.google.com/>
6. <https://skillcrush.com/blog/>
7. <https://www.reactlibraries.com/>
8. <https://www.w3schools.com/js/>