|  |
| --- |
| Programming |
| Sanjarbek Matlabov |

**Programming**

**O‘qituvchi:** Xoshimov Abdulaziz

**Guruh identifikatori: 23-404**

**Talaba ID: 230592**

**Taqdim etilgan sana:** 18.01.2025 yil

**BTEC o'quvchilar topshiriqlarini baholash va deklaratsiya**

Baholash uchun ishlarni taqdim etganda, har bir o'quvchi ish o'ziniki ekanligini tasdiqlovchi deklaratsiyani imzolashi kerak.

|  |  |
| --- | --- |
| **O‘quvchi (talaba) identifikatori:** | 230592 |
| **Baholovchi nomi:** | Xoshimov Abdulaziz |
| **Birlik yoki komponent raqami va nomi:** | Programming |
| **Topshiriq nomi:** | Internal Assignment |
| **Topshiriq topshirilgan sana:** | 18.01.2025yil |

Iltimos, har bir topshiriq uchun berilgan ishlarni sanab o'ting. Ishlarni topish mumkin bo'lgan sahifa raqamlarini ko'rsating yoki ishlarning mohiyatini tavsiflang (masalan, grafik, rasm).

|  |  |  |
| --- | --- | --- |
| **Topshiriq vazifasi ma'lumoti** | **Ishlar taqdim etildi** | **Sahifa** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |
| --- |
| **O'quvchi deklaratsiyasi**  Men ushbu topshiriq uchun taqdim etilgan ish meniki ekanligini tasdiqlayman. Ishda foydalanilgan manbalarga aniq havola qildim. Men noto'g'ri deklaratsiya noto'g'ri ishlashning bir shakli ekanligini tushunaman.  **O‘quvchi imzosi: C:\Users\qwer\Pictures\qol.png Sana: 18** .01.2025 yil |

Mundarija

[KIRISH 3](#_Toc188099697)

[A.P1: 3](#_Toc188099698)

[**A.P2:** 4](#_Toc188099699)

[A.M1: 6](#_Toc188099700)

[B.P3: 7](#_Toc188099701)

[**B.P4** 11](#_Toc188099702)

[B.M2: 13](#_Toc188099703)

[B.D1**:** 15](#_Toc188099704)

[**C.P5:** 18](#_Toc188099705)

[**C.D2:** 18](#_Toc188099706)

[**D.M4:** 19](#_Toc188099707)

[D.D3: 20](#_Toc188099708)

[D.D4: 21](#_Toc188099709)

[**Xulosa** 22](#_Toc188099710)

[adabiyotlar ro'yxati 23](#_Toc188099711)

KIRISH

Menga yuklatilgan vazifa:

Algoritm nima ekanligini va dasturni yaratish bosqichlarini birma bir tavsiflab berish. OOP yani obyektga yonaltirilgan dasturlash umumiy tavsifini berib otaman va menga berilgan senario bilan yani chakana savdo uchun api yarataman unda qollagan kodlarimni tavsiflab beraman. Yana yozilgan kodni qanday testdan otkazishlarni sinashni birma bir sinab korsatib otaman.

## A.P1:

Algoritm – bu muammoni yechish yoki belgilangan maqsadga erishish uchun ketma-ket bajarilishi kerak bo‘lgan aniq va cheklangan qadamlar to‘plamidir. Algoritmlar dasturlashning asosiy elementi bo‘lib, ular har qanday dastur yoki tizimning ishlash mexanizmini belgilaydi.

Algoritm quyidagi xususiyatlarga ega bo‘lishi kerak:

1. **Aniqlik** – har bir qadam tushunarli va aniq bo‘lishi lozim.
2. **Cheklanganlik** – algoritm biror vaqt ichida yakunlanishi kerak.
3. **Kirish ma’lumotlari** – algoritm boshlanishida kerakli ma’lumotlar kiritilishi kerak.

**Chiqish ma’lumotlari** – algoritm yakunida natija olinishi kerak.

**Qaytaruvchanlik** – algoritmni bir nechta marta qo‘llaganda natijasi bir xil bo‘lishi kerak.

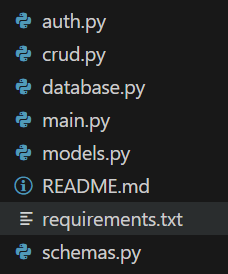
Ilova Yaratish Jarayoni

Ilova yaratish dasturlash va texnologiyalar sohasida keng qamrovli jarayon bo‘lib, bir necha bosqichlarni o‘z ichiga oladi. Quyida ilova yaratish jarayoni bosqichma-bosqich tushuntiriladi:

1. Talablarni aniqlash

Menga berilgan vazifada asosiy talab autentisifikatsiya bolishi kerak. Malumotlar bazasi, admin va user hususiyatlari va xavfsizlik uchu parollarni xeshlash kerak.

1. Ilovani umumiy dizaynini tayyorlash yani qanday fayllarda qanday kodlar yozilishi kerak



Auth.py bu faylda token asosida mijoz va admin rolini boshqarish kodi yozilishi kerak.

Crud.py bu faylda umumiy admin va userning xususiyatlarni boshqarish uchun funksiyalar yozilishi kerak.

Database.py database yaratish jarayoni yozilishi kerak.

Main.py bu faylda fastapi api endpointlari yozili kerak.

Models.py umumiy modellar yaratiladigan fayl.

Schemas.py validatsiya uchun ochilgan fayl.

3. Dasturlash

Bu bosqichda dasturchilar algoritmlarni dastur kodiga aylantiradi:

**Til tanlash**: Bizga berilgan talabdan foydalanib python fastapi kutubxonasidan foydalanaman.

4.Testlash

**Funktsional testlash**: Har bir funksiyaning to‘g‘ri ishlashini tekshirish.

**Qayta testlash**: Ilova qayta-qayta sinovdan o‘tkaziladi va xatolar tuzatiladi.

**Ishlash testlari**: Ilova katta hajmdagi ma’lumotlar bilan ishlaganda qanday ishlashini sinash.

**A.P2:**

Protsessual, ob'ektga yo'naltirilgan va hodisalarga asoslangan dasturlash paradigmalari

Protsessual dasturlasj

Fokus: dasturni bosqichlar yoki protseduralar (funktsiyalar) ketma-ketligiga bo'lish.

Xususiyatlari:

Algoritmlar va bosqichma-bosqich bajarishga urg'u berish.

Ma'lumotlar va funktsiyalar alohida ob'ektlardir.

Yuqoridan pastga yondashuv: muammolarni kichikroq, boshqariladigan kichik muammolarga ajratish.

Misollar: C, Paskal, FORTRAN

Ob'ektga yo'naltirilgan dasturlash (OOP)

Fokus: Haqiqiy ob'ektlarni xususiyatlari (ma'lumotlari) va xatti-harakatlari (usullari) bo'lgan ob'ektlar sifatida modellashtirish.

Xususiyatlari:

Inkapsulyatsiya: ob'ekt ichidagi ma'lumotlarda ishlaydigan ma'lumotlar va usullarni birlashtirish.

Meros: mavjud ob'ektlardan yangi ob'ektlar (sinflar) yaratish, ularning xususiyatlari va xatti-harakatlarini meros qilib olish.

Polimorfizm: ob'ektlarning ko'p shakllarni olish qobiliyati, moslashuvchanlik va kodni qayta ishlatish imkonini beradi.

Abstraktsiya: ob'ektning ichki tafsilotlarini yashirish, faqat muhim xususiyatlarni ochib berish.

Misollar: Java, C++, Python, C#

Voqealarga asoslangan dasturlash

Diqqat: ko'rsatmalarning qat'iy ketma-ketligiga rioya qilish o'rniga hodisalarga javob berish (foydalanuvchilarning o'zaro ta'siri, tizim signallari).

Xususiyatlari:

Voqealar ishlovchilari: Muayyan hodisalar sodir bo'lganda ishga tushadigan funktsiyalar.

Asinxron bajarilish: Hodisalar istalgan vaqtda ro'y berishi mumkin, bu esa chiziqli bo'lmagan dastur oqimiga olib keladi.

Odatda grafik foydalanuvchi interfeyslari (GUI) uchun ishlatiladi.

Misollar: JavaScript, Tkinter yoki PyQt kabi kutubxonalarga ega Python

Paradigmalar o'rtasidagi munosabatlar

Protsessual va OOP: OOPni protsessual dasturlashning kengaytmasi sifatida ko'rish mumkin, bu erda protseduralar ob'ektlar ichida inkapsullanadi.(Neubauer, T. (2023)),

OOP va hodisalarga asoslangan: inkapsulyatsiya va meros kabi OOP tushunchalari hodisalarni qayta ishlash vositalarini tashkil qilish va hodisalarga javoban ob'ektlar holatini boshqarish uchun ishlatilishi mumkin.

Gibrid yondashuvlar: Ko'pgina zamonaviy dasturlash tillari va ramkalar ushbu paradigmalarning elementlarini birlashtiradi. Masalan, GUI ilovasi voqealarga asoslangan yondashuv yordamida tuzilishi mumkin, asosiy mantiq esa OOP tamoyillari yordamida amalga oshiriladi.

Xuloas

Protsessual: protseduralar va algoritmlarga e'tibor beradi.

OOP: ob'ektlarga, ularning xususiyatlariga va ularning o'zaro ta'siriga e'tibor qaratadi.

Voqealarga asoslangan: voqealarga javob berishga qaratilgan.

Ushbu paradigmalar dasturiy ta'minot haqida fikr yuritish va tizimlashtirishning turli usullarini ta'minlaydi, ularning har biri o'zining kuchli va zaif tomonlariga ega. Paradigmani tanlash hal qilinayotgan muayyan muammoga va dasturiy ta'minotning kerakli xususiyatlariga bog'liq.(Burgess, M. (2020)).

## A.M1:

1. Talablarni aniqlash

Dasturiy ta'minotning maqsadi va funksionalligi aniqlanadi.

Texnik va biznes talablari belgilanadi.

2. Rejalashtirish va dizayn

Arxitektura dizayni: Dasturning umumiy tuzilishi belgilanadi (modullar, sinflar, interfeyslar).

Algoritmlar va yechimlar: Muammoning algoritmik yechimi ishlab chiqiladi.

Maketlar: Grafik interfeysga ega dasturlar uchun foydalanuvchi interfeysining maketi yaratiladi.

3. Kod yozish

Tanlangan dasturlash tili: Kod tanlangan dasturlash tilida yoziladi.

Modulyar kodlash: Kodni boshqarish oson bo'lishi uchun u kichik qismlarga bo‘linadi.

Kod standartlari: Kodingizning tushunarli va o‘qilishi oson bo‘lishini ta'minlash uchun standartlarga rioya qilinadi.

4. Versiya nazorati

Kod versiyalarini boshqarish uchun Git, SVN kabi tizimlardan foydalaniladi.

Kodning o‘zgarishlari kuzatiladi va boshqariladi.

5. Kompilyatsiya yoki interpretatsiya

Kompilyatsiya: Kod mashina tiliga o‘tkaziladi (masalan, C++ yoki Java).

Interpretatsiya: Kod qadam-baqadam bajariladi (masalan, Python yoki JavaScript).

6. Testlash

Birlik testlari: Har bir modulning alohida ishlashi sinovdan o‘tkaziladi.

Integratsiya testlari: Modullar birgalikda qanday ishlashi tekshiriladi.

Funktsional testlar: Dasturning foydalanuvchi talablariga mos kelishi sinovdan o‘tkaziladi.

Xatolarni aniqlash va tuzatish: Topilgan muammolar hal qilinadi.

7. Qo‘lda yoki avtomatlashtirilgan sinovdan o‘tkazish

Qo‘lda sinov o‘tkazish yoki CI/CD (uzluksiz integratsiya va yetkazib berish) tizimlari yordamida avtomatlashtirilgan jarayonlardan foydalaniladi.

8. Yig‘ish va paketlash

Dasturni build jarayonida bitta ijro fayli yoki paketga o‘raladi.

Masalan, Java dasturlari uchun .jar yoki .war fayllari yaratiladi.

9. Qo‘llab-quvvatlash muhitida ishlatish (Deployment)

Dastur serverga o‘rnatiladi yoki foydalanuvchilarga yetkaziladi.

Cloud, on-premises yoki mobil platformalarda tarqatiladi.

10. Ijro etish (Execution)

Dasturni foydalanuvchi ishga tushiradi.

Foydalanuvchi dastur bilan ishlaydi va dastur real vaqt rejimida buyruqlarni bajaradi.

11. Monitoring va tahlil

Dasturning ishlash samaradorligi kuzatiladi.

Loglar va xatolarni yozish orqali kelgusidagi xatolar oldini olish tahlil qilinadi.

B.P3:  
Algoritm bajaradigan dastur yozish uchun biz uchun berilgan “Chakana savdoga buyurtma berish” tizimini api ni yaratamiz. Bu api ni yaratishni oldindan tuzib berilgan senariy orqali yozamiz bu senariyda nimalar keltirilgan. “newera cash & carry” bu Toshkentdagi qaysidir manzilda joylashgan 5000 mijozga hayoliy chakana savdo dokoni. U xaridorlarga turli xil oziq ovqat mahsulotlarni sotishadi. Hozirda dokon kundalik operatsiyalarini qogoz bilan qol bilan olib bormoqda. Bu esa dokonda noaniqlik va samarasiz ishni kopaytirib yubormoqda. Dokonni esa barcha trankzatsiyalarini oop bilan databasega saqlaydigan qilib ularga dastur tuzib berish kerak.   
Tizimning asosiy maqsadi:  
Chakana savdo do'konining kundalik operatsiyalarini boshqarish uchun avtomatlashtirilgan buyurtma tizimini joriy qilish. Tizim oziq-ovqat mahsulotlari buyurtmalarini boshqarish, mijozlar va administratorlar uchun autentifikatsiya va ma'lumotlarning xavfsizligini ta'minlashi kerak.  
Talablar:  
1. Foydalanuvchi rollari:   
 **Administrator**:  
Mahsulotlarni boshqaradi (qo'shish, yangilash, o'chirish).  
Buyurtmalarni boshqaradi (barcha buyurtmalarni ko'rish, buyurtma tafsilotlarini ko'rish).

 **Mijoz**:  
Mahsulotlar ro'yxatini ko'rish.  
Buyurtma yaratish.  
Buyurtmalar holatini kuzatish.

2. Tizim funksiyalari:

Autentifikatsiya:

Mijozlar va administratorlar uchun tizimga kirish

Parollarni xavfsiz xeshlash

3. Malumotlar bazasi:   
Jadvallar:

Users: ID, Username, Password, role

Products: ID, name, description, price

Orders: ID, customerID, status

Orderdetails: ID, orderid, productID,quantity.

Api end-points:

Admin uchun:

Mahsulotlarni boshqarish:

* Yangi mahsulot qo'shish.
* Mahsulot tafsilotlarini olish.
* Mahsulotni yangilash.
* Mahsulotni o'chirish.

Buyurtmalarni boshqarish:

* Barcha buyurtmalarni olish.
* Aniq buyurtma tafsilotlarini olish.

**Mijozlar uchun**:

Mahsulotlarni ko'rish.

Yangi buyurtma yaratish.

Mijozga xos buyurtmalar ro'yxatini olish.

Buyurtma holatini tekshirish.

**API hujjatlari**:

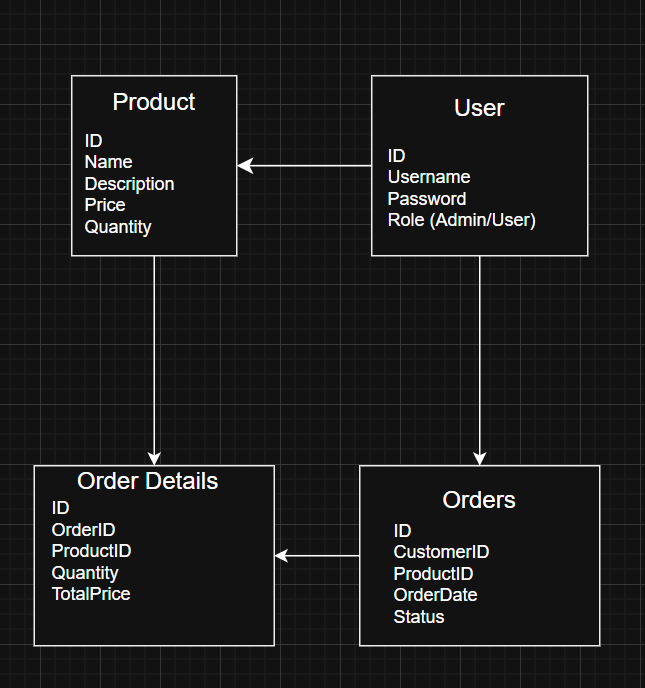
FastAPI-ning OpenAPI hujjatlari (/docs orqali).

**Xavfsizlik**:

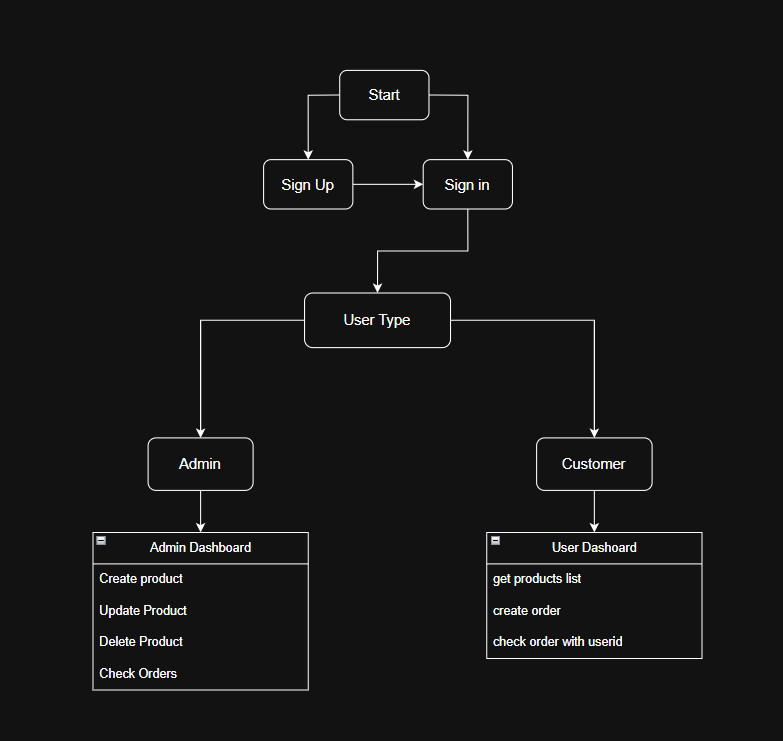
OAuth2 va JWT asosida autentifikatsiya.

Parolni xeshlash va xavfsiz saqlash.

Umumiy dastur algoritmini chizish:  
Umumiy dastur algoritmini senariya asoslanib chizib oldim bu algoritmni chizishda draw.io dan foydalandim.



Tepada ko’rib turganingizdek u yerda umumiy dalgoritmini tuzib oldim yani Product oz ichiga nimalar olishi,User, Order details, Orders bolimlari oz ichiga nimalar olishlarini yozib oldim. Endi dasturning ishlash tartibiga ham alohida osonlashtirish uchun strukturasini ketma ketligini chizib olaman.

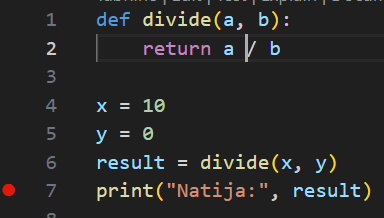


Bu flowchartda umumiy dastur qanday ishlashi tushuntirilgan.

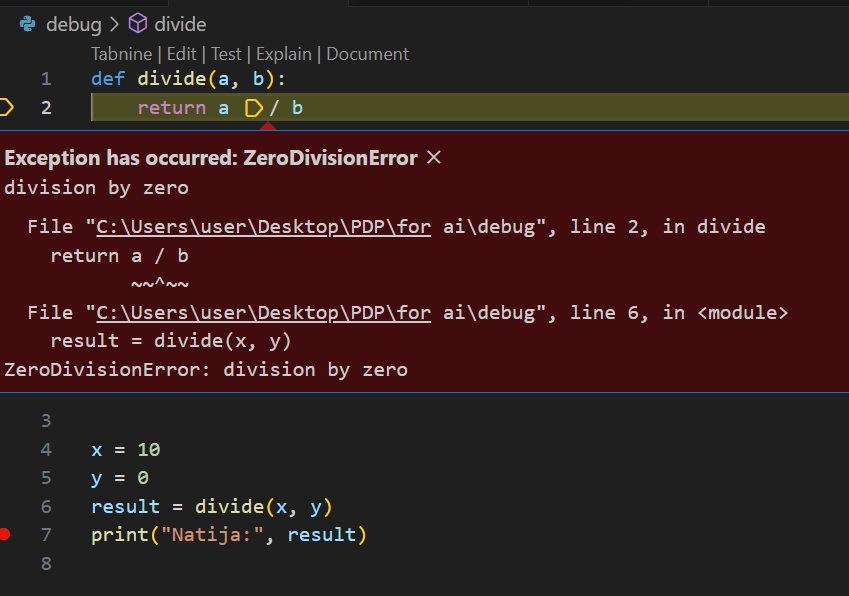
**B.P4**

IDE da turli nosozliklarni aniqlash vositalari bor. Debugger, Unittest kabi hozir debuggerni qanday ishlatishni tushuntirib ketaman.

Debugger — u dastur kodini xatoliklar yoki muammolarni topish va tuzatish uchun ishlatiladi.

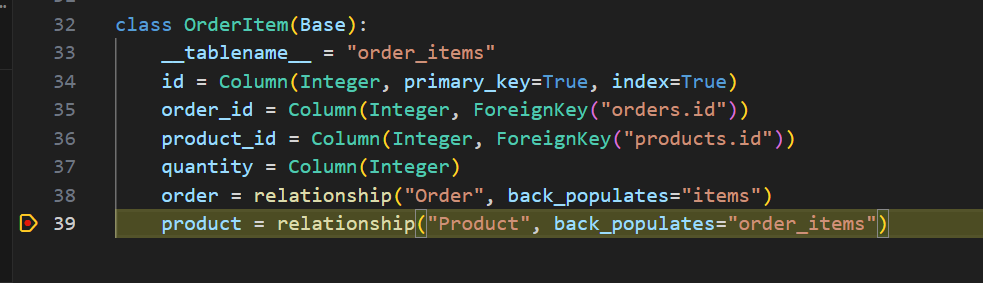


Bu yerda oddiy kod misolida debugging qanday ishlashini tushuntiryabman. Breakpoint qoyiladi qayergacha tekshirmoqchi bolsangiz.



Bu yerda debug qilingandan keyin testdan mhuvaffaqiyatsiz otgan kod yani error bor kod misoli.

Menga berilgan topshiriqni kodida qanday debug qilganimni ko’rsatib otaman har bir kod ishlashini tekshirib ko’raman.



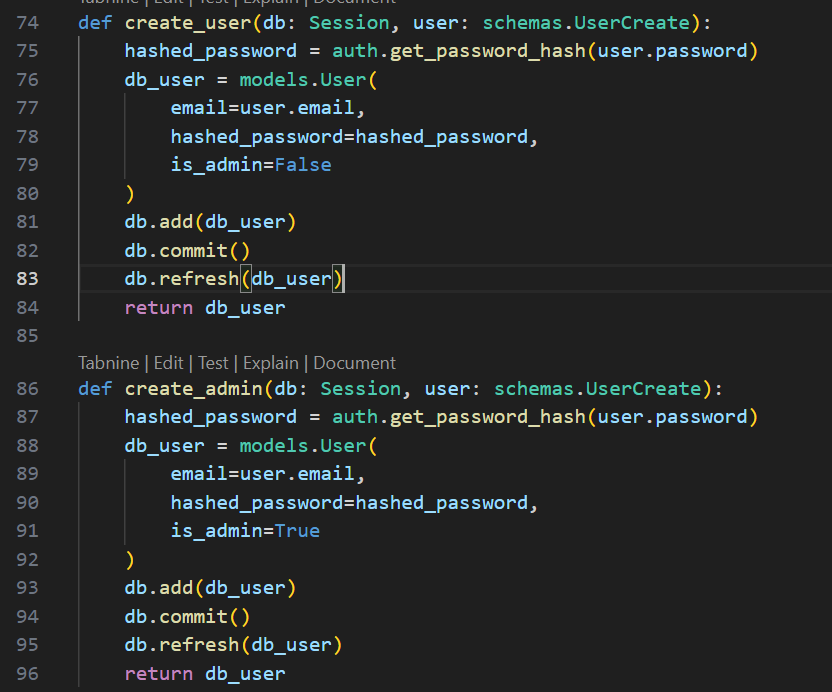
Bu kodda sqlalchemy da database table yartish yozilgan. Bu koddni tekshirishimdan maqsad relationshiplari torg’ri ulanganmi yoki yoq shuni tekshirdim.

## B.M2:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Xususiyat** |  |  | | --- | |  |  |  | | --- | |  | | |  | | --- | | **Protsessual** | | **Obyektga yo‘naltirilgan** | **Hodisaga asoslangan** |
| Ma’lumotlarni boshqarish | Funksiyalarda ma’lumotlar qayta ishlanadi | Ma’lumotlar sinflar va obyektlar orqali boshqariladi | Ma’lumotlar hodisalarga mos ravishda boshqariladi |
| Modullik va qayta ishlatish | Har bir funksiya mustaqil, lekin kod takrorlanishi yuqori | Sinflar va obyektlar kodni modulli va qayta ishlatishga mos qiladi | Har bir funksiya ma’lum hodisaga javob beradi, modullik yaxshi |
| Ish oqimi boshqaruvi | Funksiyalarni chaqirish tartibi aniq belgilanishi kerak | Ob’ektlar va ular o‘zaro munosabatlari ish oqimini boshqaradi | Hodisalar (masalan, HTTP so‘rovlar) asosida boshqariladi |
| Afzalliklari | Oddiy jarayonlarni dasturlashda qulay | Katta tizimlar uchun kodni strukturaviy boshqarish imkonini beradi | Interaktiv va hodisaga asoslangan dasturlar uchun samarali |
| Kamchiliklari | |  | | --- | |  |  |  | | --- | | Katta tizimlarda kod boshqaruvi qiyinlashishi mumkin | | Kod ko‘proq bo‘lishi va nisbatan murakkabroq bo‘lishi mumkin | Hodisalar o‘rtasidagi bog‘liqlikni tushunish qiyin bo‘lishi mumkin |

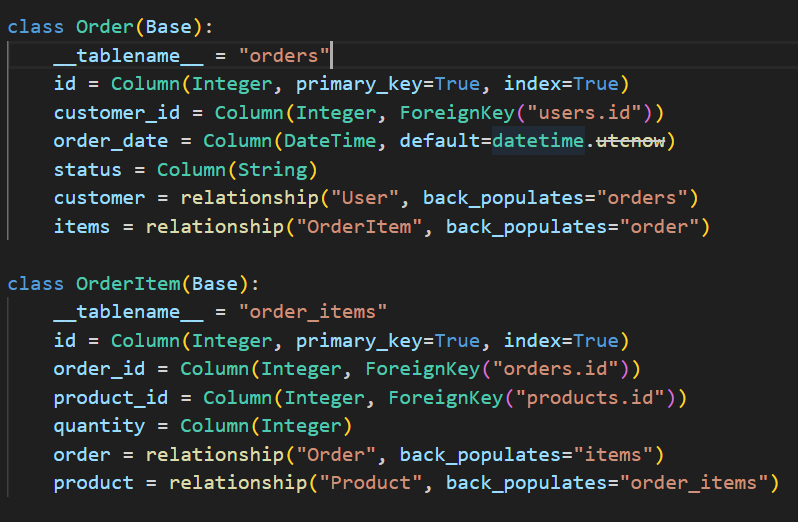
Ushbu usullarni oz joyida qollash muhim hisoblanadi bu sizni ishingizni va vaqtingizni kamaytiradi.

* 1. Protsessual dasturlash:



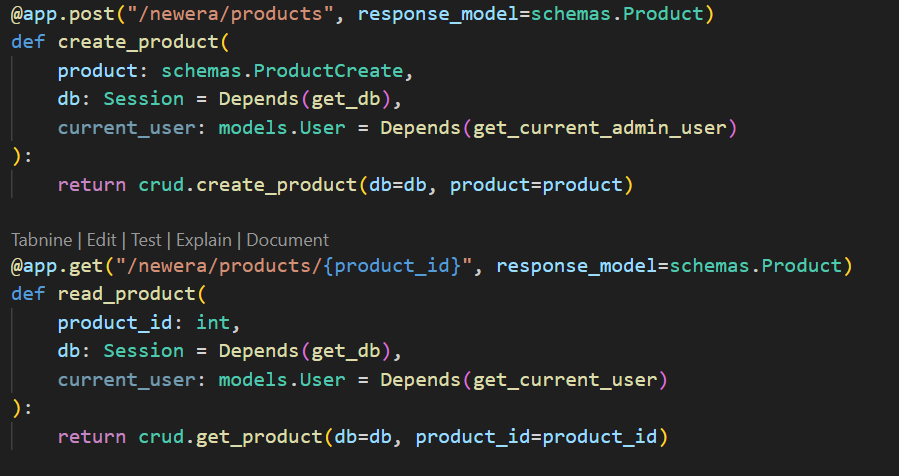
**Bu kodda admin va user signUP tizimi yozilgan funksiya. Bu funksiyani qachon chaqilirganda ishlaydi va protsessual dasturlashga misol boladi.**

* 1. Obyektga yonaltirilgan dasturlash

****

**OOP ga misol boladigan bu kodda umumiy shablon tayorlangan yani sqlalchemyga table yaratish va unga malumotlarni saqlash uchun ozgaruvchilar bilan yozilgan.**

* 1. Hodisaga asoslangan dasturlash

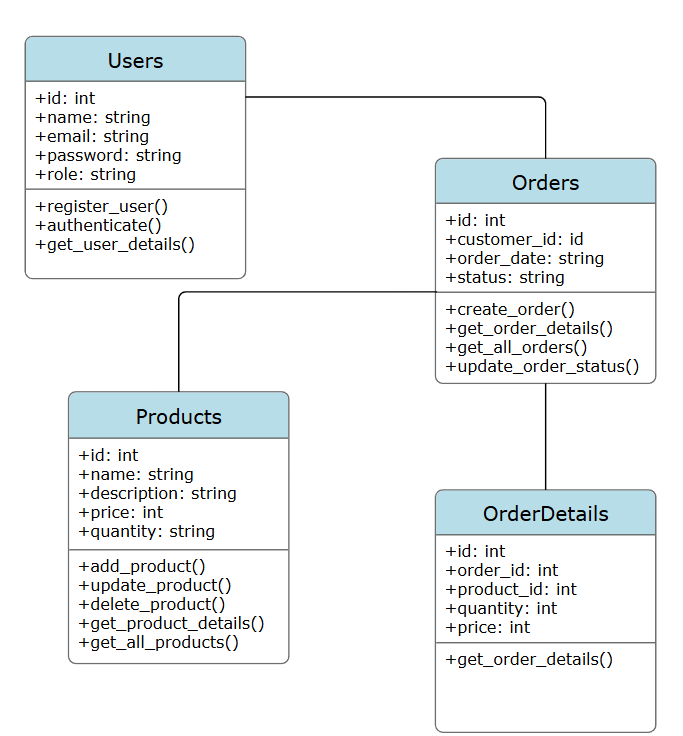
****

**App.post admin uchun yozilgan bu requestda admin qachonkiproduct yaratganida ishlaydi shuning uchun hodisaga asoslangan dasturlashga misol bo’ladi.**

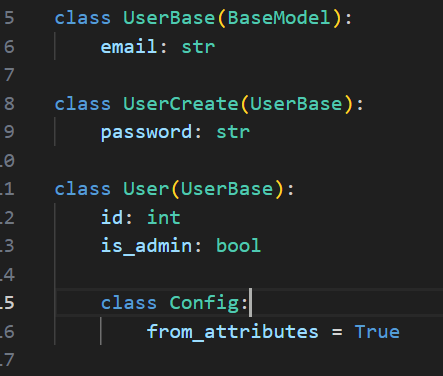
## B.D1**:**

Dasturni yaxshiroq tushunish uchun UML diagrammalarini tarkibiga kiruvchi class diagrammasidan foydalandim.

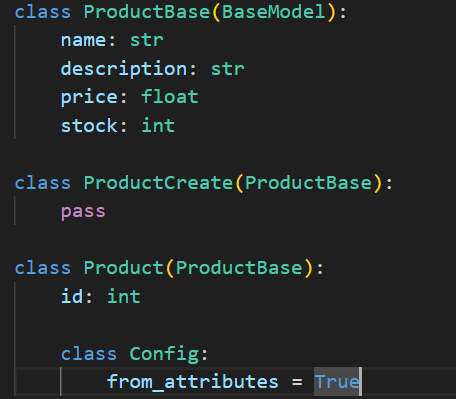
UML (Unified Modeling Language) diagrammalari dasturiy ta'minotni loyihalash, tahlil qilish va hujjatlashtirish jarayonida tizimni vizual tasvirlash uchun ishlatiladigan grafik vositalar majmuasidir.



Bu diagrammada classlar oz ichiga nimalar olishi va bu classlar yordamida qanday amallar bajarishi yozilgan va buni kod bila yozilganda qanday korinadi qanday ishlashini koramiz.



Bu rasmda user modeli tasvirlangan lekin men bu modelni uchga bolib yozdim umumiy userbase modelida faqat username bor lekin usercreate boganida parollarni xeshlashi uchun usercreate degranh class ochib userBasedan voris oldim sababi signup qilgan vaqtda parol xeshlashni oson qilish va cleancode tamoyiliga amal qildim. Config classi esa pythonni atributlarini olishga yordm berishi uhchun yaratilgan.



Bu kodda esa product uchun yaratilgan sinflar tasvirlangan. Clean kodga amal qilgan holda yaratiladigan product uchun productcreate classi alohida ochilgan va pass berib otilgan.

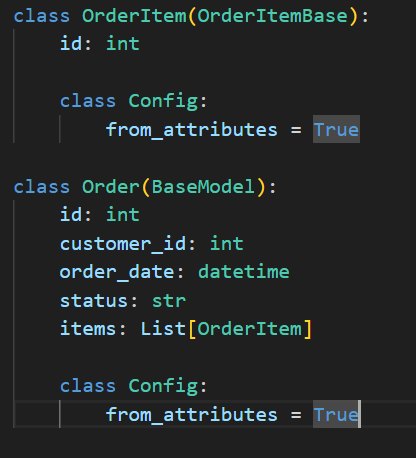
**C.P5:**

Kod yozishda kodlassh standartiga amal qilish muhimdir. Keyingi dasturchi sizning yozgan kodingizni oson tushinishi uchun clean codega amal qilish kerak.

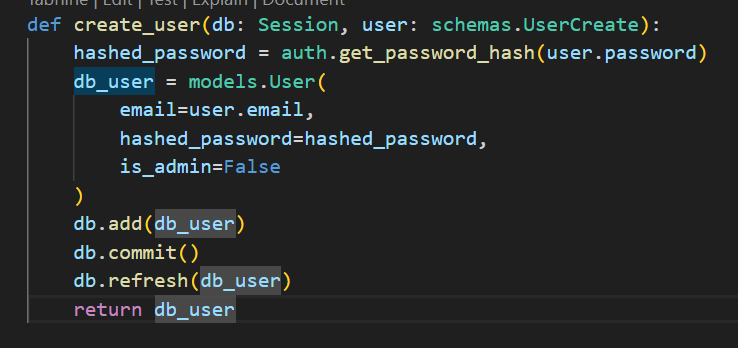
Kodlash jarayonida har bir fayylga bitta vazifani bajaradigan funksiya yozish kerak va bu bir xil maqsadga xizmat qilishi kerak. Kodlash standartlari ishlab chiquvchilar kod yozishda amal qiladigan ko'rsatmalar va eng yaxshi amaliyotlar to'plamidir. Ushbu standartlar nomlash konventsiyalari, kodni tashkil etish, chekinish, sharhlash, xatolarni qayta ishlash va boshqalar kabi turli jihatlarni qamrab oladi. Kodlash standartlarini minimal xato bilan toza, o'qilishi mumkin bo'lgan va samaraliroq kodni ishlab chiqish uchun qoidalar, texnikalar va eng yaxshi amaliyotlar sifatida ko'rib chiqing.( Bose, S. (2023))

**C.D2:**

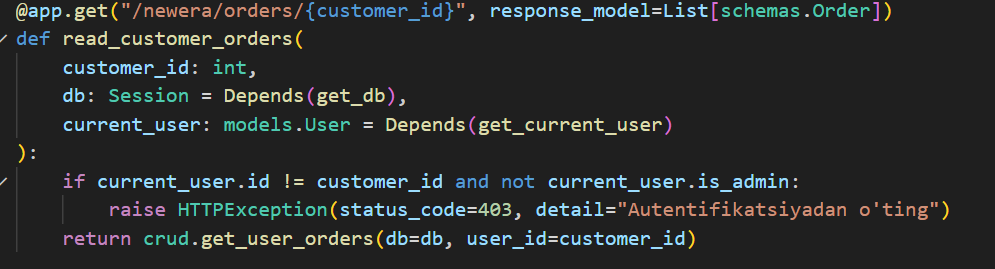
Men yozgan chakana savdo dokoni uchun apida ushbu uch turdagi paradigmalarni yozishni ko’rish mumkin.



Bu kod OOP yani obyektga yonaltirilgan dasturlash bilan yozilgan.



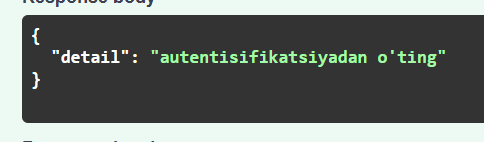
Bu kod esa protsessual dasturlashda misol boladi sababi bu aniq bir funksiyadir yani create\_user bu esa oz ichiga yaratgan modeldan kelib chiqib user create qiladi.



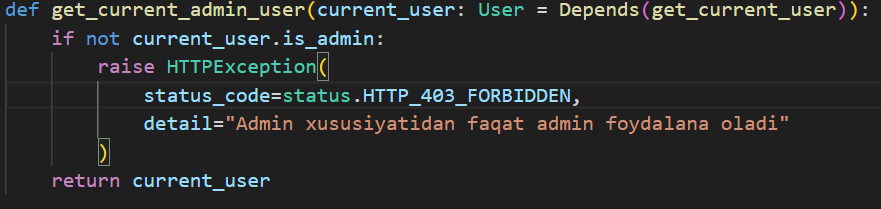
Ushbu kod hodisaga asoslangan dasturlashga misol boladi. Bu koda qachonki admin mijozlarni buyurtmalarini oqishga harakat qilganda ishlaydi.

**D.M4:**

Ilovani ishlash jarayonida turli xatolar kuzatildi masalan baglar jid yozish sintaksisidagi xatolar kod yakunlanayotgan qilingan test tufayli bu xatolar yuzaga keldi bu xatolarni korishingiz mumkin. Ilovni mijoz sifatida ishga tushirdim va men adminning xususiyatlaridan foydalanmoqchi boldim lekin u mendan autentisifikatsiyadan oting dedi



Lekin men autentisifikatsiyadan o’tganman ammo yana soradi bu yerga admin xususiyatini mijozlar foydalana olmasligini bildiruvchi xabar yozish kerak.



Bu kodni shunday korinishga yozib oldim va bagni tog’irladim.

D.D3:  
Agar siz yangi boshlovchi, talaba yoki dasturlashni endi boshlagan bo'lsangiz, mavjud vositalarni tushunish biroz qiyin bo'lishi mumkin. Ehtiyojlaringizga eng mosini tanlashga yordam berish uchun keling, ushbu ikkita muhim vositani - kod muharrirlari va IDElarni ajratib ko'rsatamiz.

Ikkala vosita ham kod yozish, tahrirlash va boshqarish uchun platforma sifatida xizmat qiladi, biroq ular xususiyatlari va murakkabligi bilan farqlanadi.

Kod muharriri - bu kod yozish va tahrirlash uchun engil vosita bo'lib, muhim sintaksisni ajratib ko'rsatish va ba'zan asosiy nosozliklarni tuzatishni taklif qiladi.

Integratsiyalashgan rivojlanish muhiti (IDE), ammo muharrir, tuzatuvchi, kompilyator va boshqalar kabi xususiyatlarga ega bo'lgan dasturiy ta'minotni ishlab chiqish uchun to'liq to'plamdir.(GeeksforGeeks (2024))

## D.D4:

Quyidagilar kodlash standartlariga ega bo'lishdan maqsad: Kodlash standarti turli muhandislar tomonidan yozilgan kodlarga bir xil ko'rinish beradi.Bu kodning o'qilishi va barqarorligini yaxshilaydi va murakkablikni kamaytiradi.Bu kodni qayta ishlatishga yordam beradi va xatolarni osongina aniqlashga yordam beradi.Bu dasturlashning to'g'ri amaliyotini targ'ib qiladi va dasturchilarning samaradorligini oshiradi.  
Kodlash standartlari va ko'rsatmalari izchil, o'qilishi mumkin bo'lgan va barqaror kodni ta'minlaydi, samarali ishlab chiqish va xatolarni aniqlashga yordam beradi. Ular nomlash, cheklash va hujjatlashtirish amaliyotlarini standartlashtiradi, murakkablikni kamaytiradi va kodni qayta ishlatishni osonlashtiradi. Ushbu amaliyotlarga rioya qilish umumiy dasturiy ta'minot sifati va ishlab chiqish samaradorligini oshiradi.( GeeksforGeeks (2019))

# **Xulosa**

Api yaratilishidagi kod standartlarini qollanilishini batafsil tushuntirish berildi va debug jarayonini ham korib chiqdik ilova yaratilishidagi kamchiliklarni tog’irlab ilovani soz holatga olib keldim. IDE ning qanchalik dasturchi kundalik hayotia muhim ekanligini shu dasturni yozishda yanada yaxshiroq tushunib yetdim.

adabiyotlar ro'yxati

1. Neubauer, T. (2023). *The what, why and how of event-driven programming*. [online] Quix.io. Available at: <https://quix.io/blog/what-why-how-of-event-driven-programming>.
2. Burgess, M. (2020). *Programming Paradigms — Procedural, Object Oriented, and Functional Programming*. [online] Medium. Available at: <https://mattburgess.medium.com/programming-paradigms-procedural-object-oriented-and-functional-programming-23a587c6a537>.
3. GeeksforGeeks (2024). *Difference between IDE and Code editor*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/ide-vs-code-editor/?ref=gcse\_outind [Accessed 13 Jan. 2025].
4. GeeksforGeeks (2019). *Coding Standards and Guidelines*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/coding-standards-and-guidelines/#conclusion [Accessed 14 Jan. 2025].
5. Bose, S. (2023). *Coding Standards and Best Practices To Follow*. [online] BrowserStack. Available at: https://www.browserstack.com/guide/coding-standards-best-practices.