|  |
| --- |
| Programming |
| Sagdullayev Azizxon |

**4-qism: Programming Level 3**

**O’qituvchi:** Abdumajidov Dostonbek

**Guruh IDsi:** 24-201

**Talaba IDsi: 240146**

**Topshirish sanasi:** 18.01.2025

**BTEC Learner Assessment Submission and Declaration**

Har bir o’quvchi vazifani baholash uchun taqdim etganda, yozilgan deklaratsiya o’ziniki ekanligini tasdiqlovchi imzo qoldirishi kerak.

|  |  |
| --- | --- |
| **O’quvchi (Talaba) IDsi:** | 240146 |
| **Baholovchining ismi:** | Abdumajidov Dostonbek |
| **BTEC Dasturining mavzusi:** | Pearson BTEC 3-darajali Axborot texnologiyalari tizimlari |
| **Qism yoki Komponent raqami va Mavzu:** | 4-qism: Programming  Avtomabil raqamlarini sotish dasturi |
| **Vazifa mavzusi:** | Internal Assignment |
| **Vazifani topshirish sanasi:** | 18.01.2025 |

Iltimos, har bir topshiriq uchun taqdim etilgan dalillarni sanab o'ting. Dalillarni topish mumkin bo'lgan sahifa raqamlarini ko'rsating yoki dalillarning mohiyatini tavsiflang (masalan, video, rasm).

|  |  |
| --- | --- |
| **Topshiriq manbalari** | **Taqdim etilgan dalillar** |
|  |  |
|  |  |
|  |  |
|  |  |

|  |
| --- |
| **O’quvchi deklaratsiyasi**  Ushbu vazifa uchun topshirilgan ish meniki ekanligini tasdiqlayman. Ishda foydalanilgan manbalarga aniqlik kiritdim va ko’rsatib o’tdim. Noto’g’ri deklaratsiya noto’g’ri ishlashning bir shakli ekanligini tushunaman.  **Talabaning imzosi: Sana:** 18.01.2025 |

Mundarija

Contents

# 1.0. Kirish

# **1.1. Topshiriq haqida qisqacha ma`lumot**

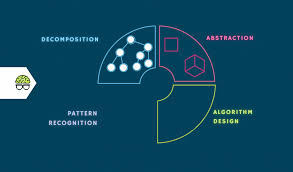
**Avtomobil raqamlarini boshqarish va sotishni avtomatlashtirish dasturi loyihasi**

Avtomobil raqamlarini saqlash, boshqarish va sotish jarayonlarini avtomatlashtirish hozirgi zamon talabidan kelib chiqadigan, o‘ta muhim masalalardan biri bo‘lib, ushbu yo‘nalishda dastur yaratish – oddiy texnik ish emas, balki unda inson intilishlari, ijodiy yondashuvi va hissiyati mujassam bo‘lishi kerak. Men aynan shunday dastur ustida ishlayapman va bu nafaqat texnik jarayon, balki ijodiy mehnat bilan uyg‘unlashgan bir faoliyat sifatida qiziqarli va mas’uliyatli tajriba hisoblanadi. Dastur yaratish jarayonida muvaffaqiyatning kaliti – to‘g‘ri rejalash va dasturiy tamoyillarga amal qilishdir. Ishning boshlanishidan boshlab, har bir qadamda tanqidiy fikrlash, mantiqiy tahlil va hisoblashga oid fikrlash (Computational Thinking) tamoyillari muhim rol o‘ynaydi. Bu tamoyillarni to‘g‘ri qo‘llash, har bir qarorni asoslash va yakuniy mahsulotning sifatli bo‘lishini ta’minlaydi. Avvalo, loyiha mijozning talablariga mos ravishda rejalashtiriladi. Bu jarayonda har bir mayda detal diqqat bilan ko‘rib chiqiladi, chunki mijozlarning har biri o‘z ehtiyojlari bilan o‘ziga xosdir. Ularning talablarini to‘liq tushunish va bu talablarni qondiradigan mahsulot yaratish – bu nafaqat texnik vazifa va insoniy mas’uliyatni ham anglatadi. Bu jarayonda mijoz bilan yaqin muloqotda bo‘lish va ularning fikrlarini hisobga olish, mahsulotga bo‘lgan ishonchni oshiradi. Loyihaning dastlabki rejasi taqdim etilgach, kamchiliklarni aniqlash va ularni bartaraf etish bo‘yicha jiddiy ish olib boriladi. Bu jarayon – faqat texnik ish emas, balki ijodiy tahlil va yondashuvni ham talab qiladi. Kamchiliklarni ko‘rish va ularga yechim topish – dasturchining nafaqat bilim, balki tajriba va ijodkorlik qobiliyatini ham namoyon etadi. Kod yozish bosqichiga o‘tishda esa ishning murakkabligi oshadi. Har bir kod satri – bu nafaqat texnik buyruq, balki ijodiy yechimdir. Ba’zan murakkab vaziyatlar yuzaga keladi va bu muammolarni hal qilish uchun noodatiy yondashuvlar talab qilinadi. Dasturchi uchun bunday qiyinchiliklarni yengish va innovatsion yechimlarni topish – haqiqiy mahoratning namunasidir. Kod yozilib bo‘lgach, dastur sinovdan o‘tkaziladi. Bu bosqichda dasturdagi har qanday kichik muammo ham topiladi va tuzatiladi. Sinov jarayoni nafaqat dastur barqarorligini ta’minlash, balki mijozning ishonchini oqlash uchun ham juda muhimdir. Dastur mijozga topshirilgach, uning talablariga qanchalik mos kelgani baholanadi va dastur yanada mukammallashtiriladi. Loyiha davomida duch kelgan qiyinchiliklar va ularni hal qilish yo‘llari – bu faqat texnik tajriba emas, balki insoniy saboqlarning bir qismidir. Har bir bosqichda o‘rganilgan yangi bilimlar, yechilgan murakkab vazifalar va o‘rganilgan saboqlar kelajakdagi loyihalar uchun muhim zamin yaratadi.

# 2.0.Tadqiqotim dasturlashda muhim bo‘lgan hisoblashga asoslangan fikrlash ko‘nikmalari va tamoyillari haqida.

Muammolarni aniqlash, ularni kichik qismlarga ajratish va algoritmik usullar yordamida samarali yechim topish ko‘nikmasidir. Ushbu qobiliyat dasturiy ta'minot ishlab chiqishda asosiy ahamiyatga ega, chunki u jarayonlarni optimallashtirish, kodni soddalashtirish va muammolarni aniq hal qilish imkonini beradi.

# 2.1. A.P1



1-rasm. Decomposition, Abstraction, Pattern Recognition, Algoritmm Design

1. **Decomposition –** bu murakkab muammoni kichikroq va boshqarilishi oson qismlarga ajratish usuli. Ba’zida katta bir muammoni birdaniga hal qilish qiyin bo‘lib tuyulishi mumkin, lekin uni mayda bo‘laklarga bo‘lib, har birini alohida yechish osonlashadi. Keyinchalik, shu kichik qismlarning yechimlari bir joyga jamlanadi va umumiy muammo hal qilinadi. Bu usul, ayniqsa, dastur ishlab chiqishda juda foydali bo‘lib, jarayonni tartibli va qulay qiladi. Masalan biz avtomabil dasturini yasamoqchi bo`lsak uni dekompazitsiya qilishimiz quyidagicha bo`ladi:

* Avtomabil ma`lumotlarini qismlarga bo`lish
* Malumotlarni qayta ishlash va tahlil qilish
* Jiddiy yoki zararli avtomabil prognozlari haqida xabar berish uchun bildirishnomalarni amalga oshirish.

Xulosa: Dekompozitsiya murakkab muammolarni boshqarilishi oson bo‘lgan kichik qismlarga ajratish usuli bo‘lib, har bir qismini alohida hal qilish orqali umumiy muammoni yechishni osonlashtiradi. Bu yondashuv dastur ishlab chiqishda tartib va samaradorlikni ta’minlaydi. Masalan, avtomobil dasturini yaratishda, avval avtomobil ma’lumotlarini qismlarga ajratish, ularni qayta ishlash va tahlil qilish, so‘ngra bildirishnomalar yordamida muhim xabarlarni yetkazishni rejalashtirish mumkin. Bu bosqichlar orqali murakkab vazifani qulay tarzda bajarish mumkin bo‘ladi.

1. **Camputational Thinking** - bu muammolarga tizimli va mantiqiy yondashish usuli bo‘lib, u turli sohalarda, nafaqat dasturlashda qo‘llaniladi. Ushbu fikrlash turiga ega bo‘lgan odamlar miqdoriy yoki ma’lumotlarga asoslangan masalalarni hal qilish uchun samarali yechimlarni yaratishi mumkin. Hisoblash fikrlashning asosiy tushunchalari – modellashtirish, abstraksiya, dekompozitsiya, algoritmik va mantiqiy fikrlashdir. Bu fikrlash har bir insonda turlicha namoyon bo‘ladi: masalan, olim uni eksperimental protsedura sifatida ko‘rishi, kompyuter olimi esa algoritmlar va ularning ilovalarini o‘rganish deb tushunishi mumkin. Misol sifatida, ob-havo va iqlimni bashorat qilish uchun kompyuter modellari ishlatilishini keltirish mumkin.
2. **Pattern Recognition -** bu 0 va 1 dunyosidagi hamma narsa. Masalan Avtomatik elektron pochta filtra. Email xizmati spam xabarlarni aniqlash uchun namunalarni tanib olishdan foydalanadi. Masalan, ko‘pincha spam xatlarda shubhali kalit so‘zlar ya`ni("tezroq boy bo‘ling", "to‘liq bepul") yoki noodatiy manzillar bor. Tizim bu namunalarni aniqlab, kerakli xabarlarni alohida qismga ajratadi. Yana bir misol: Yuzni aniqlash texnologiyasi. Telefon yoki kamera yuzni tanib olish uchun ko‘z, burun, og‘iz joylashuvi kabi ma’lum namunalarni aniqlaydi. Tizim bu namunalarni avvaldan saqlangan yuz ma’lumotlari bilan solishtirib, mos kelishini tekshiradi va mos kelsa telefon ochiladi yoki mos kelmasa telefon qulflanadi.
3. **Abstraction -** bu kodni soddalashtirish va boshqarishni osonlashtirish uchun ishlatiladigan usul. U murakkab jarayonlarni oddiy ko‘rinishda ifodalashga yordam beradi. Abstraksiya yordamida biz bir nechta sinflar uchun umumiy bo‘lgan atributlar va xatti-harakatlarni belgilab, ularni boshqa sinflar uchun tayanch qilib ishlatamiz. Masalan, ota-sinf (mavhum sinf) yaratamiz va unda umumiy xususiyatlar yoki metodlarni belgilab qo‘yamiz. Bu ota-sinfdan vorislik orqali boshqa sinflar “farzand sinflar” o‘ziga kerakli xususiyatlarni va metodlarni oladi. Bu yondashuv orqali dasturni modulli qiladi va kodni qayta ishlatish imkonini beradi. Abstraksiya orqali yuqori darajadagi funksiyalarni tushunish osonlashadi va dastur murakkab qismi bilan ishlash jarayoni yanada sodda bo‘ladi. Masalan, biz transport vositalarini ifodalash uchun "Transport" deb nomlangan mavhum sinf yaratishimiz va undan "Avtomobil", "Velosiped", "Samolyot" kabi sinflar uchun asos sifatida foydalanishimiz mumkin. Bu barcha sinflarni birlashtirish va ularga umumiy ishlov berishni osonlashtiradi.

Masalan bunga Bankomatni misol qilib keltirishim mumkin:

**Bankomat (ATM):** Bankomatdan pul yechish jarayonida siz faqat tugmalarni bosasiz va natijani ko'rasiz (pul beriladi). Siz ichki jarayonlarni (hisobingizdagi mablag‘ni tekshirish, bank tarmog‘i bilan aloqa qilish) bilishingiz shart emas. Bankomatning murakkab funksiyalari foydalanuvchidan yashiriladi, bu esa abstraksiyaning real hayotdagi ko‘rinishi bo`ladi.

1. **Algorithm Design -** Algoritmlar dasturiy ta'minotning asosi bo‘lib, ular muammolarni mantiqiy ketma-ketlikda hal qilishni ta'minlaydi. Yaxshi ishlab chiqilgan algoritmlar dasturlarni tezkor, samarali va ishonchli qiladi. Masalan, navigatsiya ilovasi yoki ro‘yxatni tartiblash kabi kundalik vazifalar algoritmlar yordamida aniq va samarali bajariladi. Shu bois, algoritmlarni ishlab chiqish va ulardan to‘g‘ri foydalanish dasturlashning muhim qismi hisoblanadi.

Bunga hayotiy misol keltiraman: Masalan, siz biror joydan ikkinchi joyga borishni rejalashtiryapsiz va eng qisqa yo‘lni izlayapsiz. Bunda sizning maqsadingiz – yo‘lning uzunligini kamaytirish va vaqtni tejash. Shunday qilib, siz navigatsiya ilovasini ochasiz va u sizga eng qisqa yo‘llarni topishga yordam beradi.

Umumiy qilib misol keltiraman: Masalan ijaraga berish.

**Dekompozitsiya**:

Avtomobilni ijaraga olish jarayonini osonlashtirish uchun, uni kichik qismalarga ajratishimiz kerak:

* Avtomobil tanlash: Yil, model, yoqilg‘i turi, narx kabi parametrlarni hisobga olish.
* Avtomobilni olish: Arizani to‘ldirish, haydovchilik guvohnomasini tekshirish.
* To‘lov qilish: Kredit kartasi orqali to‘lovni amalga oshirish.
* Ijara muddati: Ijara muddati va to‘lovlarni boshqarish.

Har bir qism alohida muammo sifatida ko‘riladi va osonroq yechiladi.

**Pattern Recognition**:

Avtomobil tanlashda, biz ma’lum bir patternni aniqlashimiz mumkin. Misol uchun, agar mijoz oldin biror modelni ijaraga olgan bo‘lsa, u yana o‘sha modelni tanlash ehtimoli yuqori. Shuningdek, avtomobillarni ijaraga olishda narxlar orasidagi o‘xshashliklarni ko‘rish, mijozga eng yaxshi variantni taklif qilish uchun pattern recognition ishlatiladi.

**Abstraksiya**:

Avtomobilni ijaraga olish jarayonida ko‘plab tafsilotlarni ko‘rib chiqish zarur, masalan, avtomobilning texnik holati yoki ijaraga olishning maxsus shartlari. Biroq, mijoz uchun bu tafsilotlar muhim emas, ular faqatgina "Avtomobilni ijaraga olish" jarayonining umumiy shaklini ko‘rishi kerak. Bu yerda abstraksiya ishlaydi: barcha tafsilotlar birlashtiriladi va foydalanuvchiga faqatgina zarur bo‘lgan ma'lumotlar ko‘rsatiladi.

**Algoritmik dizayn**:

Endi, biz bu jarayonlarni avtomatlashtirish uchun algoritm ishlab chiqamiz. Masalan, bir foydalanuvchi avtomobil tanlashni boshlaydi, tizim avval uni shartlar asosida to‘g‘ri variantlarni ko‘rsatadi (filterlash). Keyin, tizim avtomobilni olish jarayonini avtomatik ravishda tekshiradi va tasdiqlaydi. To‘lovni amalga oshirish uchun algoritm orqali to‘lov usulini tanlash va tasdiqlash, ijarani davom ettirish uchun muddatni aniqlash jarayonlari amalga oshiriladi.

Xulosa:

Yuqoridagi misolda, avtomobilni ijaraga olish jarayonida dekompozitsiya, pattern recognition, abstraksiya va algoritmik dizayn birgalikda ishlagan. Dekompozitsiya orqali jarayon kichik qismlarga ajratilib, har bir qism alohida hal qilindi. Pattern recognition yordamida mijozlarning xulq-atvori aniqlanib, eng mos variantlar taklif qilindi. Abstraksiya esa tafsilotlarni yashirib, foydalanuvchiga faqat zarur bo‘lgan ma'lumotlar taqdim etildi. Algoritmik dizayn yordamida jarayon samarali tarzda boshqarildi va avtomobilni ijaraga olish jarayoni tez va oson amalga oshirildi. Bu barcha yondashuvlar birgalikda tizimli va samarali natijaga erishish uchun ishladi.

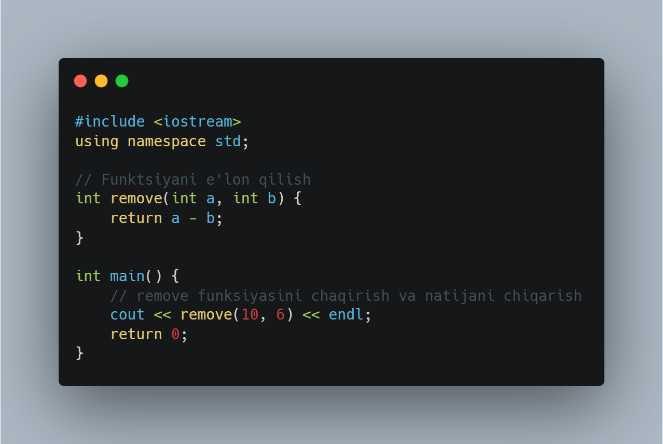
# 2.2. A.P2

**Dasturlash tamoyillari** – bu samarali va kengaytiriladigan kod yozish uchun dasturchilarga yo'l-yo'riq ko'rsatuvchi ko'rsatmalar. Ular hozirgi katta dasturiy loyihalarda keng qo'llaniladi, lekin turli dasturlash tillarida ularning qo'llanilishi farq qilishi mumkin. Bu tamoyillar yordamida kodni o'qilishi oson, kengaytiriladigan va samarali qilish mumkin.

**Modularity or Decomposition**  – bu murakkab tizimlarni kichik va boshqarilishi oson bo'lgan modullarga ajratish jarayoni. Har bir modul o'ziga xos funktsiyalarni bajaradi, bu tizimni tushunishni, saqlashni va boshqarishni osonlashtiradi.

* Python dasturlash tilida kodni modullashtirish, uni kichik qismlarga ajratish orqali amalga oshiriladi. Har bir qism alohida vazifani bajaradigan funksiya yoki sinf sifatida yozilib, keyin kerak bo'lganda chaqiriladi. Bu yondashuv kodni o'qilishi oson, toza va kengaytiriladigan qiladi. Modullarga ajratish orqali kodni qayta ishlatish va uni boshqarishni osonlashtiradi.
* Python dasturlash tilida kodni modullashtirish uchun functionlar, classlar va packagelardan foydalanish mumkin.
* C++ dasturlash tilida ham kodni modullashtirish uchun functionlar, classlar (klasslar) va header fayllari orqali erishish mumkin.

Masalan:

python

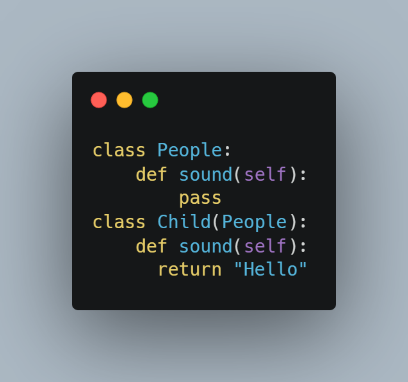
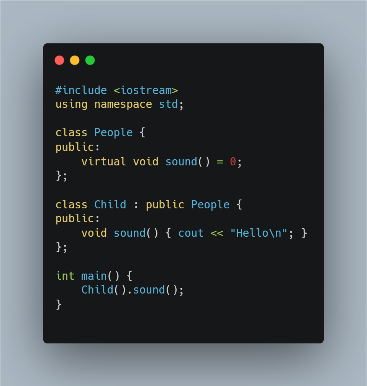

2-rasm. Python: 3-rasm. C++:

**Abstraksiya** – bu kodning murakkab qismlarini yashirib, faqat foydalanuvchi uchun kerakli va muhim qismlarini ko'rsatish jarayonidir. Bu dasturiy ta'minotning ishlashini soddalashtiradi va kodni o'qish va tushunishni osonlashtiradi.

**Python** tilida esa abstraktsiya, asosan, klasslar va dekoratorlar (masalan, @abstractmethod) orqali amalga oshiriladi. Klasslar orqali umumiy xatti-harakatlar va xususiyatlar ta'riflanadi va dekoratorlar yordamida ma'lum metodlarni abstrakt qilish mumkin.

C++ tilida abstraktsiyaga mavhum sinflar (abstract classes) va virtual metodlar orqali erishiladi. C++ da, Java'dagi interfeyslarga o'xshash tarzda, virtual metodlar yordamida umumiy xatti-harakatlar va atributlar ta'riflanadi, va bu metodlar sinfga o'xshash sinflarda o'zgartirilishi mumkin. C++ ning mavhum sinflari orqali kodni yanada soddalashtirish va kengaytirish mumkin.

4-rasm. Python: 5-rasm. C++:

**Meros olish (Inheritance)** - bu bir klassning xususiyatlari va xatti-harakatlarini boshqa bir klassga o'tkazish jarayonidir, bu esa kodni qayta ishlatish va kengaytirish imkoniyatini beradi.  
• Pythonda bu prinsip subclass'lar orqali amalga oshiriladi, ya'ni meros oluvchi klass ota klassdan parametr sifatida xususiyatlar va metodlarni oladi.  
• C++da esa meros olish : belgisidan foydalanib amalga oshiriladi.

*Masalan:*

6-rasm. Python: 7-rasm. 6-rasm. Python:

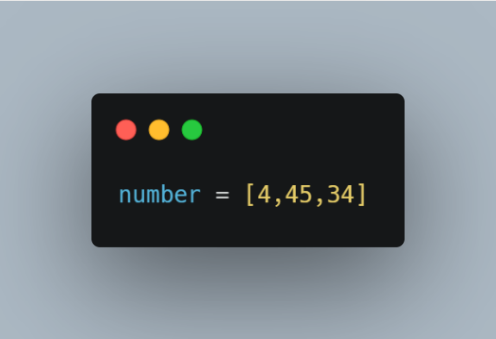
 

"**Algoritmlar va Ma'lumotlar Tuzilmalari** – algoritmlar va ma'lumotlar tuzilmalarini to'g'ri va samarali qo'llash dasturning tezligini oshiradi va unumdorligini yaxshilaydi.

Pythonda list, dictionary va set kabi ma'lumot tuzilmalariga ma'lumotlarni tartiblash, saralash va qidirish kabi amallarni bajarish uchun allaqachon tayyorlangan (built-in) funksiyalarni ishlatish mumkin.

C++ tilida esa massivlar va ko'rsatkichlar yordamida murakkab ma'lumotlar tuzilmalarini yaratish va boshqarish osonlashadi. Bu dasturchiga ma'lumotlarga to'g'ridan-to'g'ri murojaat qilish va ularni samarali boshqarish imkoniyatini beradi."

*Masalan:*

8-ram. Python:  9-rasm. C++:

Xulosa:

Dasturlash tamoyillari kodni samarali va kengaytiriladigan qilishga yordam beradi. Ular murakkab tizimlarni boshqarish va tushunishni osonlashtiradi. Python va C++da bu tamoyillar bir xil maqsadni ko'zlaydi, ammo ularning qo'llanilishi farq qiladi.

**Modullashtirish** – tizimni kichik, boshqarilishi oson qismlarga ajratish. Pythonda bu funksiya va sinflar yordamida, C++da esa header fayllari va funksiyalar orqali amalga oshiriladi.

**Abstraksiya** – murakkab qismlarni yashirib, faqat zarur ma'lumotlarni ko'rsatish. Pythonda klasslar va dekoratorlar, C++da esa mavhum sinflar va virtual metodlar yordamida bu jarayon soddalashtiriladi.

**Meros olish** – bir klassning xususiyatlarini boshqasiga o'tkazish. Pythonda subclass'lar, C++da esa : operatori yordamida bu amalga oshiriladi.

Bu tamoyillar dasturlashda kodni soddalashtirish, kengaytirish va tezlashtirish uchun ishlatiladi.

# 2.3. A.P3

Dasturiy ta'minotni loyihalash tamoyillari – bu yuqori sifatli va foydalanuvchi ehtiyojlariga javob beradigan dasturiy ta'minotni yaratishda asosiy ko'rsatmalar hisoblanadi. Ushbu tamoyillarga amal qilish orqali dasturchilar ishonchli, kengaytiriladigan, texnik xizmat ko'rsatish qulay va foydalanuvchilarga mos dasturlarni ishlab chiqishlari mumkin. Hop, endi ushbu tamoyillar sifatli dasturiy ta'minotni yaratishga qanday ta'sir ko'rsatishini batafsil ko'rib chiqamiz.

1. **Modullik:**  
   **Tamoyil:** Dasturiy ta'minotni turli vazifalarni bajaruvchi kichik va mustaqil modullarga ajratish.  
   **Amalga oshirish:** Misol uchun, veb-ilovada foydalanuvchi interfeysi, ish jarayonlari logikasi va ma'lumotlar bazasi qatlamlarini alohida bo'limlarga bo'lish.  
   **Natija:** Tizimning biror qismida kiritilgan o'zgartirishlar butun dasturga ta'sir qilmaydi. Bu esa uni testlash, texnik xizmat ko'rsatish va yangilashni ancha osonlashtiradi.
2. **Chunarlilik (soddalik va aniqlik):**

**Tamoyil**: Dasturiy ta'minot nafaqat foydalanuvchilar uchun, balki uni kelajakda qo'llab-quvvatlovchi va rivojlantiruvchi dasturchilar uchun ham oson tushunarli bo'lishi lozim.

**Amalga oshirish**: Bu maqsadga sodda, intuitiv interfeyslar yaratish, toza kod yozish tamoyillariga rioya qilish va modulli tuzilma yordamida erishish mumkin.

**Natijada**: Dasturdagi xatolarni tuzatish, uni rivojlantirish va texnik xizmat ko'rsatish osonlashadi, shu bilan birga foydalanuvchilar uchun ham qulaylik ta'minlanadi.

1. **Masshtablash imkoniyati:**

**Tamoyil:** Dasturiy ta'minotni kelajakda ortib boradigan talablar yoki foydalanuvchilar ehtiyojlarini qondira oladigan tarzda loyihalash.

**Amalga oshirish:** Kengaytiriladigan arxitekturalar, masalan, mikroservislar, ma'lumotlar bazasini samarali ishlatish uchun so'rovlarni optimallashtirish va bulut texnologiyalaridan foydalanish orqali amalga oshirish mumkin**.**

**Natija:** Dastur biznesning o'sishi bilan birga rivojlanib, unumdorlikni yo'qotmasdan ko'proq foydalanuvchilarni yoki katta hajmdagi ma'lumotlarni boshqarishi mumkin bo'ladi.

1. **Moslashuvchanlik va ta'mirga qulaylik:**

**Tamoyil:** Dasturiy ta'minot har qanday texnologik yoki talab o'zgarishlariga moslashishi va zarur bo'lgan o'zgartirishlarni kiritish oson bo'lishi lozim.

**Amalga oshirish:** Toza va aniq tuzilgan kod yozish, yaxshi hujjatlar yaratish, versiyalarni boshqarish vositalaridan (masalan, Git) foydalanish hamda avtomatik testlarni ishlab chiqish orqali amalga oshiriladi.

**Natija:** Dasturchilar dasturiy ta'minotni foydalanuvchilarning ehtiyojlariga mos ravishda tez va samarali tarzda yangilash imkoniyatiga ega bo'ladilar.

1. **Yangilashga qulaylik yoki Ta’mirga layoqatlilik.**

**Tamoyil: Dasturiy ta'minotni yangi talablar yoki texnologik o'zgarishlarga moslashtirish hamda har qanday o'zgartirishlarni amalga oshirish oson bo'lishi kerak.**

**Amalga oshirish: Aniq va toza kod yozish, yaxshi hujjatlarni yaratish, versiyalarni boshqarish uchun vositalardan (masalan, Git) foydalanish va avtomatik testlarni ishlab chiqish bu jarayonni yengillashtiradi.**

**Natija: Dasturchilar foydalanuvchilarning ehtiyojlari o'zgarishi bilan dasturni tez va samarali ravishda modernizatsiya qila oladilar.**

**Xulosa:**

Dasturiy ta'minotni loyihalash tamoyillari yuqori sifatli, ishonchli va qulay dasturlar yaratishga yordam beradi. Modullik tizimni kichik bo'laklarga ajratib, oson boshqarish va yangilash imkonini beradi. Soddalik va tushunarlilik dasturiy ta'minotni nafaqat foydalanuvchilar, balki dasturchilar uchun ham qulay qiladi.

Masshtablilik kelajakda o'sadigan talab va foydalanuvchilar sonini qondirishga imkon beradi. Moslashuvchanlik esa dasturga har qanday o'zgartirishni oson kiritishni ta'minlaydi. Ushbu tamoyillarga rioya qilish orqali dasturchilar foydalanish uchun qulay, texnik xizmat ko'rsatish va rivojlantirishga mos dasturlar yaratadilar.

# 2.4. A.M1

Kompyutatsion fikrlash dasturiy ta'minotni ishlab chiqishda asosiy rol o'ynaydi, chunki u murakkab muammolarni oddiyroq va boshqariladigan qismlarga ajratishga yordam beradi. Bu usul orqali dasturchilar har bir qismni mustaqil tarzda hal qilishlari va umumiy tizimni yanada samarali va aniq tarzda yaratishlari mumkin. Shuningdek, bu yondashuv kodni tezroq ishlab chiqishga va keyinchalik texnik xizmat ko‘rsatishga yordam beradi.

Abstraksiya tamoyili dasturchilarga murakkab tizimlarning faqat muhim tomonlariga e’tibor qaratib, ortiqcha tafsilotlardan chalg‘imaslik imkonini beradi. Bu yondashuvni, ayniqsa, foydalanuvchi interfeyslarini ishlab chiqishda ko‘rish mumkin, chunki bunda texnik vositalar emas, balki foydalanish qulayligi birinchi o‘ringa qo‘yiladi. Abstraksiya murakkab muammolarni oddiyroq shaklda ko‘rishga yordam beradi, bu esa butun tizimni boshqarishni yengillashtiradi.

Shuningdek, algoritmik fikrlash ham katta ahamiyatga ega, chunki u samarali va tez ishlovchi algoritmlar yaratish imkonini beradi. Samarali algoritmlar nafaqat dasturiy ta’minot unumdorligini oshiradi, balki tizim resurslarini tejashga ham yordam beradi. Masalan, katta hajmdagi ma'lumotlar bilan ishlovchi tizimlar uchun yaxshi tuzilgan algoritmlar ishlov berish vaqtini qisqartirib, foydalanuvchi tajribasini yaxshilaydi.

Tizimli yondashuv esa dasturlash jarayonini tartibli qiladi. Bu yondashuv yordamida dasturchilar ishlarini bosqichma-bosqich rejalashtiradilar, har bir bosqichda yuzaga kelishi mumkin bo‘lgan xatolarni aniqlaydilar va ularni vaqtida bartaraf qiladilar. Natijada, bu dasturning sifatini uzoq muddat davomida yuqori darajada ushlab turish imkonini beradi.

Umuman olganda, kompyutatsion fikrlash vositalaridan foydalanish dasturiy ta'minotni ishlab chiqishda nafaqat ishlab chiqish tezligini oshiradi, balki uning sifatini va foydalanish qulayligini ham ta’minlaydi. Bu yondashuv dasturiy ta'minotning barqarorligini saqlash, texnik xizmat ko‘rsatish va kengaytirish jarayonlarini yengillashtirishga yordam beradi. Shu bois, dasturchilar ushbu tamoyillarni faol qo‘llagan holda, samarali va sifatli dasturlar yaratish imkoniga ega bo‘ladi.

# 2.5. A.D1

Hisoblashga asoslangan fikrlash dasturiy ta'minot dizaynini va sifatini yaxshilashda katta ahamiyatga ega. Ushbu yondashuv murakkab muammolarni yechish uchun samarali va modulli yechimlarni yaratishga yordam beradi. Hisoblash fikrlashning asosiy tamoyillari, xususan, abstraksiya va dekompozitsiya, dasturiy ta'minotni loyihalash jarayonlarini osonlashtiradi. Bundan tashqari, u algoritmik fikrlashni rivojlantirish va tizimlarni samarali tashkil qilish imkonini beradi.

Shu bilan birga, hisoblash fikrlash tamoyillarini noto'g'ri qo'llash salbiy oqibatlarga olib kelishi mumkin. Masalan, abstraksiya haddan tashqari murakkab bo'lsa yoki dekompozitsiya noto'g'ri amalga oshirilsa, dasturiy ta'minotning ishlashi yomonlashadi va hatolarga yo'l qo'yilishi ehtimoli ortadi. Shuningdek, bu yondashuvni o'zlashtirish yangi boshlovchilar uchun qiyin bo'lishi mumkin, chunki u yanada yuqori darajadagi analitik fikrlashni talab qiladi.

Biroq, hisoblash fikrlashni to'g'ri qo'llash dasturiy ta'minot sifatini oshirish uchun juda muhimdir. Masalan, dasturiy ta'minotni loyihalashda abstraksiya va dekompozitsiya o'rtasidagi muvozanatni saqlash tizimning moslashuvchanligini va samaradorligini oshiradi. Shuningdek, modullarni qayta ishlatish va ularning testlash jarayonlarini osonlashtirishga yordam beradi.

Xulosa qilib aytganda, hisoblashga asoslangan fikrlash dasturchilar uchun muhim ko'nikma bo'lib, uning tamoyillarini to'g'ri qo'llash orqali dasturiy ta'minot dizayni va sifati sezilarli darajada yaxshilanadi. Har bir dasturchi bu yondashuvni o'z ishlarida samarali foydalanishi uchun uni chuqur o'rganishi va amaliyotda qo'llashi zarur.

# 3.0. Dasturiy yechimni mijoz talablari asosida loyihalash.

# 3.1. B.P4

Mijoz talablarini tahlil qilgan holda, dastlab dasturning ma'lumotlar saqlash qismini JSON formatida tashkil etishga qaror qildim. Keyinchalik, dasturni obyektga yo‘naltirilgan dasturlash (OOP) tamoyillariga asoslangan holda UML diagrammasi yordamida loyihalashtirdim. Dasturning MVP (Minimum Viable Product) bosqichida bo‘lgani sababli uning foydalanuvchi interfeysi qismini konsolda amalga oshirish rejasini tuzdim. Endi ushbu jarayonlarni birma-bir ko‘rib chiqamiz:

*JSON formatida ma'lumotlarni saqlash loyihasi:*

1. Ma'lumotlar ombori (Database):

Dasturning ma'lumotlar saqlash qismi mijoz talablariga asoslanib, JSON formatida tashkil etiladi. Ma'lumotlar strukturasini tushunish va boshqarish uchun dastlab ER diagramma chizilib, u JSON tuzilishiga moslashtirildi.

1. JSON orqali modullar:

Dasturning ma'lumotlarini saqlash va boshqarish uchun quyidagi asosiy JSON fayllar yaratildi:

* data.json: Foydalanuvchilar haqidagi ma'lumotlar.

1. JSON asosidagi funksional bo'limlar:

**Users (Foydalanuvchilar):**

Ushbu bo‘lim foydalanuvchi ma'lumotlarini boshqaradi.

JSON fayl tarkibi quyidagicha bo'ladi:

****

10-rasm. JSON faylidagi foydalanuvchi ma`lumotlari

**Funksiyalar:**

* Yangi foydalanuvchi qo'shish.
* Foydalanuvchi ma'lumotlarini o'zgartirish yoki o'chirish.
* Foydalanuvchilar ro'yxatini ko'rish.

**CarPlates (Avtomobil raqamlari):**

Ushbu bo‘lim avtomobil raqamlarini boshqaradi.

JSON fayl tarkibi:

****

11-rasm. JSON faylidagi avtomabil raqami

**Funksiyalar:**

* Raqam qo'shish yoki o'chirish.
* Raqam ma'lumotlarini ID yoki raqam orqali qidirish.
* Barcha raqamlar ro'yxatini olish.

**Sales (Sotuvlar):**

Sotuvlar bilan bog‘liq jarayonlarni boshqaruvchi bo‘lim.

JSON fayl tarkibi:

****

12-rasm. Avtoraqam sotilganligi va sotib olinganligi haqida ma`lumot

**Funksiyalar:**

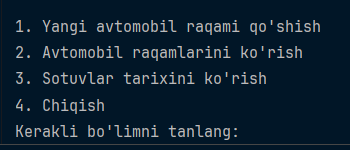
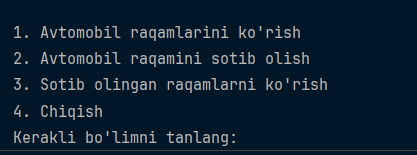
* Yangi sotuv jarayonini qo'shish.
* Sotuvni o'chirish.
* Sotuvlarni foydalanuvchi yoki raqamga ko'ra qidirish.

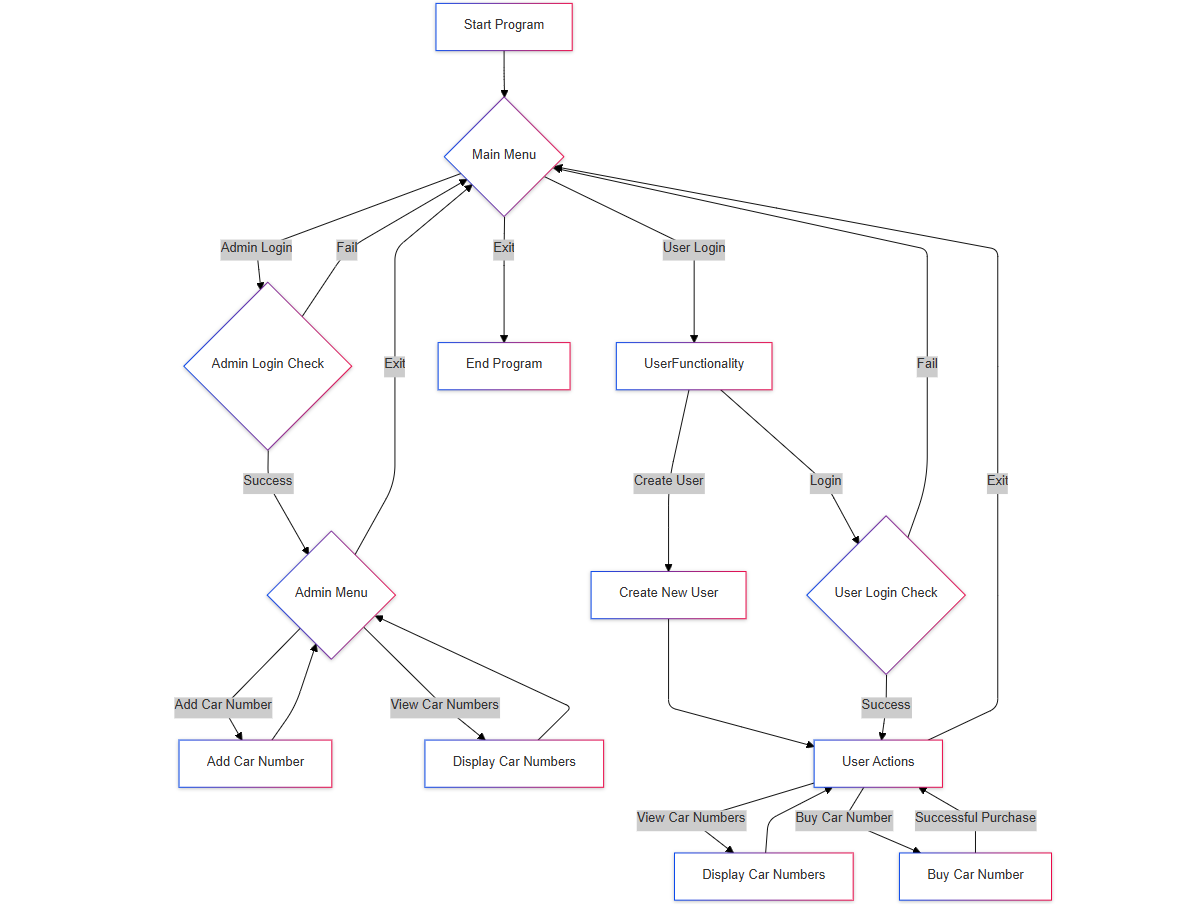
1. **User Interface (Interfeys):**

Barcha jarayonlarni birlashtiruvchi menyu qismini tashkil qilish uchun JSON fayllar bilan ishlovchi qulay boshqaruv tizimi yaratildi. Interfeys konsolda ishlab chiqildi va quyidagi imkoniyatlarni taqdim etdi:

* Foydalanuvchi va avtomobil raqamlarini boshqarish.
* Sotuvlar tarixini kuzatish.
* JSON fayllarga kiritilgan barcha o'zgarishlarni real vaqt rejimida saqlash.

13-rasm. Admin menu: 14-rasm. Foydalanuvchi uchun menu:

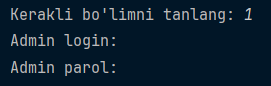


15-rasm. Flowchart

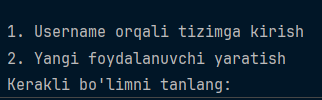
# 3.2. B.P5

Mazkur loyiha ustida ishlash jarayonida men uni boshqa mijozlar va dasturchilarga taqdim etdim. Ularning fikr-mulohazalari asosida bir qancha foydali takliflarni qabul qildim. Ushbu takliflar loyihani yanada yaxshilashga yordam berdi.

1. **JSON ma’lumotlar qismida** har bir foydalanuvchiga u sotib olgan avtomobil raqamlarini ro‘yxat sifatida qo‘shish taklif qilindi. Ushbu taklifni qabul qilib, foydalanuvchini ifodalovchi obyektga purchased\_numbers maydonini qo‘shdim. Biroq, bu usul ma’lumotlarni saqlashda ba’zi kamchiliklarga ega. Chunki ma’lumotlarni bunday formatda saqlash ma’lumotlar tuzilishidagi asosiy tamoyillardan biri – 1NF (birinchi normallashtirish shakli) talablariga mos emas. Bu tamoyilga ko‘ra, bitta maydon ichida faqat bitta qiymat saqlanishi kerak. Bu turdagi ma’lumotlarni saqlashning muqobil va samaraliroq usuli sifatida, sotuvlar haqidagi ma’lumotlarni alohida sales obyektida, foydalanuvchi identifikatori (masalan, user\_id) bilan bog‘lagan holda saqlash tavsiya etiladi. Ammo, mijozning aniq talabi bo‘lganligi sababli, purchased\_numbers maydonini qo‘shishga to‘g‘ri keldi.
2. Menyuni faqat adminlar va xodimlar uchun emas, balki oddiy avtomobil raqamlarini sotib oluvchi foydalanuvchilarga ham qulay bo’lishi uchun qayta ko‘rib chiqdim. Shunga ko‘ra, menyuni ikki qismga – admin va mijozlar uchun bo‘lishga qaror qildim. Admin va mijozlar uchun dasturdagi barcha funksionallar maxsus panelda jamlandi. Mijozlar uchun esa menyuga avtomobil raqamlarini. Endilikda dasturga kirishda foydalanuvchilar o‘z roli – admin/ mijozlar yoki oddiy mijoz sifatida tanlanadi. Admin va mijozlar uchun login va parol talab qilinadi.



16-rasm. Admin uchun menu



17-rasm. Foydalanuvchi uchun menu

Programmadan endi oddiy foydalanuvchilar ham ishlatishi mumkin.

Dastur ish jarayoni quyidagicha amalga oshiriladi. Foydalanuvchi dasturga kirgach, avval uning kim ekanligini aniqlash uchun savol beriladi. Agar foydalanuvchi oddiy mijoz bo‘lsa, u avto raqamlar bilan tanishishi yoki sotib olishi mumkin. Mijoz dastlab tizimga kirgandan so‘ng, ro‘yxatdagi raqamlarni ko‘rishi yoki yangi foydalanuvchi sifatida ro‘yxatdan o‘tishi talab qilinadi. Yangi foydalanuvchi yaratishda login va yashash manzili kiritiladi. Mavjud mijozlar esa tizimga o‘z loginlari orqali kiradilar. Oddiy foydalanuvchi sifatida tizimga kirganingizda, siz avtomobil raqamlarini ko‘rishingiz, kerakli raqamni tanlab sotib olishingiz va avval sotib olgan raqamlaringiz ro‘yxatini ko‘rishingiz mumkin. Sotib olish jarayonida tanlangan avtomobil raqami tizimdan "Sotilgan" holatiga o‘tadi va mijozning shaxsiy ro‘yxatiga qo‘shiladi. Agar foydalanuvchi admin bo‘lsa, login va parol orqali tizimga kiradi. Admin tizimga muvaffaqiyatli kirgach, unga quyidagi imkoniyatlar taqdim etiladi:

 Yangi avtomobil raqami qo‘shish: Bu jarayonda raqam va uning narxi kiritiladi va u mavjud raqamlar ro‘yxatiga qo‘shiladi.

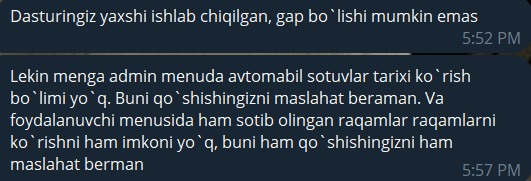
 Barcha mavjud raqamlarni ko‘rish: Admin avtomobil raqamlarining narxlari va holatlarini ko‘rishi mumkin.

 Sotuvlar tarixini ko‘rish: Admin foydalanuvchi tomonidan sotib olingan raqamlar ro‘yxati, xaridorning login ma’lumotlari va sotuv sanasi bilan tanishishi mumkin.

# 3.3. B.M4

Ushbu loyiha ishlab chiqishda dasturlash tamoyillaridan samarali foydalanishga katta e’tibor qaratilgan. Loyihada decomposition (bo‘laklarga ajratish), abstraction (mavhumlashtirish), kengaytiriluvchanlik, shuningdek, soddalik va aniqlik tamoyillari amalda qo‘llangan. Bo‘laklarga ajratish va modullilik tamoyillari, ayniqsa, menyu qismi va jarayonlarni sinflarga bo‘lishda aniq namoyon bo‘lgan. Har bir sinf faqat o‘ziga xos bo‘lgan funksiyalarni bajaradi, bu esa sinflarni o‘zgartirishni osonlashtiradi va dasturning boshqa qismlariga ta’sir qilmaslikni ta’minlaydi. Mavhumlashtirish tamoyili menyuda qo‘llanilgan bo‘lib, unda dastur ishini ta’minlash uchun zarur bo‘lgan funksiyalar aks ettirilgan, ortiqcha va keraksiz jarayonlarga o‘rin berilmagan. Menyuni imkon qadar sodda va tushunarli shaklda yaratishga harakat qilingan, chunki foydalanuvchilar uchun asosiy vazifa ma’lumotlar bilan ishlash bo‘lgani sababli menyudan foydalanish tez va qulay bo‘lishi muhim hisoblangan. Ma’lumotlar bazasi qismi esa decomposition va ma’lumotlar bazasiga oid tamoyillarga asoslangan holda tashkil etilgan. Ushbu yondashuv loyihaning yuqori darajadagi samaradorlikka ega bo‘lishini ta’minlagan va bu tamoyillar ERI diagrammasida ham yaqqol aks etgan. Natijada, loyiha mustahkam va funksional asosga ega holda, foydalanuvchilar uchun qulay shaklda ishlab chiqilgan.

* JSON ma'lumotlar bazasidan foydalanish, ma'lumotlarning xavfsizligi va boshqaruvni osonlashtirishga yordam beradi. Admin panel orqali ma'lumotlar bazasiga kirish faqat yuqori darajadagi foydalanuvchilar uchun mumkin bo'lib, oddiy foydalanuvchilar uchun bu qiyinlashadi. Dastur yaratishda JSON so'rovlarini ishlatish juda qulay, chunki uning sintaksisi intuitiv va tushunarli. Bu haqda internetda ko'plab ma'lumotlar mavjud.
* Dasturdagi asosiy jarayonlarni boshqarishda classlardan, ya'ni OOP konsepsiyasidan foydalanish, dasturni zamonaviy, moslashuvchan va tushunarli qilishga yordam beradi. Classlarni chaqirish va ishlatish juda oson bo'lib, ularni har qanday foydalanuvchi interfeysi bilan integratsiya qilish imkoniyatini yaratadi. Misol uchun, classlarni Python Kivy orqali mobil ilovada, FastAPI yordamida web dasturda yoki oddiy print() bilan konsolga chiqarishda foydalanish mumkin. Bu esa kodni o'zgartirish zaruratini kamaytiradi.
* UI (User Interface) qismini esa console da qilish bu kodni oson ishlatish va test qilish jarayonlarini juda qulaylashtiradi. Ammo buning ham anchagina kamchiliklari bor chunki console orqali dasturni boshqarish ko’pchilik uchun qiyinchiliklar olib kelishi tabiiy. Lekin bu yo’l MVP holatda dasturni namoyish qilish uchun juda qulay va samarali hisoblanadi. Kelajakda esa bu loyihani takomillashtirib mobile, desktop, yoki web dasturga moslashtirish juda qulay.
* Takomillashtirish: Dasturni yaxshilash, foydalanuvchi interfeysini optimallashtirish va texnik jihatlarini mukammallashtirish uchun fikr-mulohazalar asosida yangilanishlar kiritilishi zarur bo`ldi. Shu bilan birga, mijozlarning talablariga mos ravishda yangi imkoniyatlar qo'shilishi kerak. dastur dizaynini takomillashtirish va uning ishlashini yaxshilash maqsadida, boshqalar bilan fikr almashishni boshlaylik. Men dasturimni faqat o'zim yozganim uchun, avvalo bir nechta mijozlarga, dasturchilarga va foydalanuvchilarga ishga tushirib ko'rsatdim. Ular tomonidan quyidagi fikrlar va mulohazalar bildirildi:



18-rasm. Feedback

Bu rasmda foydalanuvchi tomonidan bir nechta mashalatlar berildi. Va unga ko`ra admin menusiga sotuvlar tarixni korish va xaridor menusiga sotib bo`lingan raqamlarni ko`rishni aytgan keyin esa qo`shilgan.

# 4.0. Mijozning ehtiyojlari asosida dasturiy yechim yaratish.

# 4.1. C6

Endi yuqoridagi loyihaga asoslanib, dastur yozishni boshlaymiz. Men bu yerda Python dasturlash tilidan foydalandim va ma'lumotlar bazasini boshqarish uchun json faylidan foydalanaman. Dastur kodi GitHub platformasida joylashgan:

<https://github.com/sagdullayev-a/Avtomabil-raqam-demo->

# 4.2. C.P7

Dastur yuqoridagi loyihaga asoslangan holda ishlab chiqildi va quyidagi funksiyalarni o'z ichiga oladi:

* **Admin/xodim uchun alohida panel.** Bu jarayonning asosiy maqsadi oddiy mijozlarga ham dasturdan foydalanish imkoniyatini yaratish. Mijoz talablariga binoan, ular o'z viloyatlarini tanlab, mavjud raqamlarni ko'rish imkoniyatiga ega bo'lishadi.
* **Avtomobil raqamlarini boshqarish.** Bu bo'limda raqamlarni qo'shish, o'chirish, tahrirlash, sotuvga qo'yilgan raqamlarni ko'rsatish va boshqa bir nechta qo'shimcha funksiyalar mavjud.
* **Sotuvlarni boshqarish.** Bu bo'limda sotuvlarni qo'shish, barcha sotuvlarni ko'rish, shuningdek, har bir mijozga tegishli sotuvlarni ko'rish imkoniyatlari mavjud.
* **Mijozlar ma'lumotlari bilan ishlash.** Yangi mijozlarni qo'shish, ularning ma'lumotlarini tahrirlash, ko'rish, o'chirish va mijozning sotib olgan raqamlar ro'yxatini saqlash imkoniyatlari taqdim etiladi.
* **Xavfsizlik tizimi.** Admin va xodimlar uchun maxsus login va parol orqali tizimga kirish amalga oshiriladi, bu xavfsizlikni ta'minlashga xizmat qiladi.

# 4.3. C.M3

Dastur mijozning talablariga mos kelishi uchun bir qator optimallashtirishlar amalga oshirildi. Ulardan ba'zilari quyidagilardan iborat:

<https://github.com/>

* Dastur optimallashtirilgan holda yana bir nechta bo`limlar qo`shildi. Sababi Foydalanuvchiga ko`proq qulaylik bo`lishi uchun.
* Dasturning menu qismi dastlab oddiy funksiyalar yordamida yaratilgan edi. Keyinchalik, uni optimallashtirish va modullashtirish maqsadida har bir menu bo'limi alohida class'lar sifatida ajratildi. Shuningdek, menuga oid barcha kodlar menu.py fayliga joylashtirildi va dasturni ishga tushurish endi main.py fayli orqali amalga oshiriladi. Loyihaga oid batafsil ma'lumotlarni GitHub'dagi havolada ko'rishingiz mumkin.
* Mijoz dastur butunlay o'zbek tilida bo'lishini talab qilganligi sababli, barcha foydalanuvchi interfeysi (UI) qismlari o'zbek tiliga tarjima qilindi. <https://github.com/sagdullayev-a/Avto-raqam/tree/main>

# 5.0 Dasturni mijoz talablariga qanchalik mos kelishini baholash

# 5.1 Dizayn va optimallashtirilgan kompyuter dasturining mijoz talablariga qanchalik mos kelishini baholash. BC.D2

Dasturimizning ustun tomomlari:

* **Modullashtirilgan kod.** Kod to'liq OOP prinsiplari asosida yozilgan va barcha jarayonlar o'ziga tegishli class'larda joylashtirilgan. Har bir class alohida faylda yozilgan, bu esa kodni intuitiv tarzda boshqarishni osonlashtiradi va xatoliklarni aniqlashni sezilarli darajada soddalashtiradi.
* **Relyatsion ma'lumotlar bazasi.** Relyatsion ma'lumotlar bazasidan foydalanishning asosiy sababi, mijozning talablarini to'liq bajarish kerakligidir. Agar tekst yoki JSON fayllari ishlatilsa, xavfsizlik jihatidan muammolar yuzaga kelishi mumkin, shuningdek, ular dasturlashda ideal deb hisoblanmaydi. SQL esa ommabop, sodda va keng qamrovli til bo'lib, sqlite3 kutubxonasidan foydalanish jarayonini ancha osonlashtiradi. Bu kutubxona boshqa ma'lumotlar bazalariga nisbatan o'z soddaligi bilan ajralib turadi.
* **CLI menyusi.** Dasturdan foydalanish uchun terminaldan foydalanish dasturchiga katta qulayliklar yaratadi. Terminal orqali barcha funksiyalarni tezda amalga oshirish mumkin va dastur testini o‘tkazish ham juda oddiy va aniq bo‘ladi. Agar ilova yoki veb sayt shaklida amalga oshirilgan bo‘lsa, bu dasturchi uchun qiyinchiliklar keltirishi mumkin, chunki bu texnologiyalarni o'rganish uzoq vaqt va tajriba talab qiladi.

Dasturning yaxshilanishi kerak bo`lgan qismlari:

* **UI.** Mijozning asosiy talabi dastur funksiyalarining to'g'ri ishlashi bo‘lganligi sababli, CLI interfeysi foydalanuvchilar uchun biroz zerikarli bo‘lishi mumkin. Shuning uchun, dasturni kelajakda mobil ilova yoki veb sayt shaklida taqdim etish foydalanuvchi tajribasini yaxshilashga yordam beradi.
* **SQLite.** Hozirda dasturning barcha minimal talablarini sqlite bilan hal qilish mumkin, ammo kelajakda dastur kengaytirilsa, sqlite katta hajmdagi ma'lumotlar bilan ishlashda qiyinchiliklar tug‘dirishi mumkin. Hozircha bu loyiha uchun to‘g‘ri keladi, lekin kengaytirish ehtiyoji yuzaga kelganda boshqa ma'lumotlar bazalarini ko‘rib chiqish kerak bo‘ladi.
* **Sinov jarayoni.** Dastur to‘g‘ri ishlayotganini qo‘lda tekshirish mumkin, ammo unittest yoki pytest kutubxonalaridan foydalanish juda foydali bo‘ladi. Test tizimlarini qo‘shish kodning aniq ishlashini va uning har qanday ehtimoliy xatoliklardan himoya qilishini ta'minlaydi.

# 5.2 Dasturni yaratishdagi qiyinchiliklar, o`ziga hos ijodkorlik va samaradorlik

Dastur ishlab chiqishni boshlashda avvaliga CarNumber, User, va Sale class'larini yaratishga qaror qilindi. Bu class'lar avtomobil raqamlarini, foydalanuvchilarni va sotuvlarni boshqarish uchun asosiy tuzilma bo'ldi.

Birinchi xatoliklar foydalanuvchilar va sotuvlar saqlanmayotganligi bilan yuzaga keldi. Buni hal qilish uchun foydalanuvchi ro'yxatini to'g'ri saqlashga e'tibor qaratildi. Sotuvlarni yozishda tarixni aniq saqlash kerak edi, shuning uchun har bir sotuvni to'liq qayd qilishni ta'minlandi.

Dastur terminalda ishlaydi, bu dasturchilar uchun qulay, ammo foydalanuvchilar uchun zerikarli bo'lishi mumkin. Shu bois, interfeysni yaxshilash uchun keyinroq ilova yoki veb-sayt shakliga o'tkazish zarur bo'ladi.

Kodning modullashtirilishi va OOP tamoyillariga rioya qilinishi kodni o'qish va xatoliklarni aniqlashni osonlashtirdi. Hozirgi vaqtda sqlite3 ma'lumotlar bazasi kichik hajmdagi ma'lumotlar uchun yaxshi ishlaydi, lekin ma'lumotlar ko'payganida bu tizimni takomillashtirish zarur bo`ldi.

Dastur muvaffaqiyatli ishlab chiqildi va mijozning talablariga mos keldi. Avtomobil raqamlarini sotib olish va ularni ko'rish imkoniyatlari foydalanuvchilarga taqdim etildi.

# 6.0 Umumiy xulosa

Mazkur hujjatda avtomobil raqamlarini boshqarish uchun dasturiy yechim ishlab chiqish jarayoni batafsil ko‘rib chiqildi. Dasturiy ta'minot ishlab chiqishda hisoblashga asoslangan fikrlash tamoyillari, dasturiy arxitektura va mijoz ehtiyojlarini qondirishga yo‘naltirilgan yondashuvlar tahlil qilindi. Dastur modullik, abstraktsiya va ob'ektga yo‘naltirilgan dasturlash (OOP) tamoyillariga asoslangan bo‘lib, bu dastur samarali va kengaytiriladigan yechim yaratishga imkon berdi.

Natijada, dastur foydalanuvchilar talablariga mos ravishda ishlashga tayyor bo‘ldi. Ma'lumotlar bazasi sifatida SQLite ishlatildi va foydalanuvchi interfeysi (CLI) minimal variant sifatida tanlandi, bu esa loyiha MVP bosqichini muvaffaqiyatli yakunlash uchun yetarli bo‘ldi. Kelajakda dastur funksiyalarini kengaytirish va optimallashtirish imkoniyatlari mavjud.

Ushbu loyiha orqali dasturchi dasturlash olamiga yangicha nuqtai nazardan qarash imkoniyatiga ega bo‘ldi. Dastlab, dasturlash faqat kod yozishdan iborat deb o‘ylagan bo‘lsa, loyiha jarayonida mantiqiy fikrlash, muammoni tahlil qilish va samarali yechish ko‘nikmalarining naqadar muhimligi tushunildi. Bu jarayon dasturchini nafaqat kod yozish, balki innovatsion yondashuvlarni yaratish va muammolarni hal qilishda ham yanada rivojlantirdi.

Loyiha davomida yuqori sifatli dasturiy ta'minot yaratish tamoyillariga amal qilindi va ijodkorlik hamda mas'uliyat bilan vazifa muvaffaqiyatli yakunlandi. Natijada, loyiha muvaffaqiyatli tugatilib, kelajakda yangi rivojlanish imkoniyatlari yaratildi.

# 7.0 Foydalanilgan manbalar

Ko`p ma`lumotlar bir saytdan olinganligi tufayli foydalanilgan manbalar ro`yxati biroz kamroq bo`lishi mumkin.

1. GeeksforGeeks. (2022). *What is Decomposition Computational Thinking?* [online] Available at: <https://www.geeksforgeeks.org/what-is-decomposition-computational-thinking/>.
2. TIWARI, N. (2024). *Understanding Abstraction in Python: Simplifying Complex Concepts*. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2024/02/understanding-abstraction-in-python-simplifying-complex-concepts/>.
3. W3Schools (n.d.). *W3Schools Online Web Tutorials*. [online] W3schools.com. Available at: <https://www.w3schools.com/>.
4. Coursera (2024). *Coursera | Online Courses & Credentials by Top Educators. Join for Free*. [online] Coursera. Available at: <https://www.coursera.org/>.
5. Stack Overflow (2024). *Stack Overflow - Where Developers Learn, Share, & Build Careers*. [online] Stack Overflow. Available at: <https://stackoverflow.com/>.
6. SQLite (2019). *SQLite Documentation*. [online] Sqlite.org. Available at: <https://www.sqlite.org/docs.html>.

**So`zlar soni**: 4524 ta