Programinio kodo analitiko maketo reikalavimų analizė

2025-04-21



# Turinys

[Turinys 2](#_Toc198217713)

[1. Įvadas 3](#_Toc198217714)

[2. Naudotojų analizė 3](#_Toc198217715)

[2.1 Įvadas 3](#_Toc198217716)

[2.2 Scenarijaus analizė: Verslo savininkas (paprasta sistema) 4](#_Toc198217717)

[2.3 Scenarijaus analizė: Verslo savininkas (sudėtinga sistema) 6](#_Toc198217718)

[2.4 Scenarijaus analizė: Programuotojas / IT specialistas 7](#_Toc198217719)

[2.5 Scenarijaus analizė: IT agentūra 8](#_Toc198217720)

[2.6. Santrauka: naudotojų scenarijų analizės įžvalgos 10](#_Toc198217721)

[3. Funkciniai reikalavimai 10](#_Toc198217722)

[3.1. Pagrindiniai funkciniai moduliai ir naudotojų poreikiai 11](#_Toc198217723)

[3.2. MVP funkcijų santrauka 12](#_Toc198217724)

[4. Ne-funkciniai reikalavimai 12](#_Toc198217725)

[4.1. Naudotojo patirties reikalavimai (UX) 12](#_Toc198217726)

[4.2. Veikimo reikalavimai 13](#_Toc198217727)

[4.3. Saugumo reikalavimai 13](#_Toc198217728)

[4.4. Prieinamumo ir tarptautiškumo reikalavimai 14](#_Toc198217729)

[4.5. Architektūrinio lankstumo reikalavimai 14](#_Toc198217730)

[5. Techniniai komponentai 15](#_Toc198217731)

[6. MVP apimtis 16](#_Toc198217732)

[7. Roadmap: tolesnės versijos 18](#_Toc198217733)

[8. Rizikos 20](#_Toc198217734)

# 1. Įvadas

Šiame skyriuje pateikiama AdoreIno produkto maketo reikalavimų analizė, kuri išplaukia iš išsamaus rinkos tyrimo, naudotojų scenarijų analizės ir identifikuotų problemų. Atliekant tyrimą buvo aiškiai nustatyta, kad dauguma IT sprendimų naudotojų – tiek techninių, tiek netechninių – susiduria su tuo pačiu esminiu iššūkiu: **sistemos, kurią jie jau turi, yra neaiški, nevertinta, o sprendimai dėl jos palaikymo, keitimo ar perdavimo priimami remiantis nuojauta, o ne duomenimis**.

Rinkos analizė parodė, kad ši problema aktuali tiek paprastų sistemų savininkams (pvz., WordPress, PrestaShop), tiek sudėtingų sprendimų naudotojams (pvz., SaaS ar individualiai kurtos platformos), taip pat IT specialistams, agentūroms ir projektų vadovams. Esami įrankiai tik fragmentiškai sprendžia šią problemą, todėl AdoreIno produkto esmė – **sujungti analizę, paaiškinimą ir sprendimų pagrindimą viename įrankyje**, pasitelkiant dirbtinį intelektą ir modernią UX logiką.

Maketo reikalavimai formuluojami tam, kad jis galėtų pademonstruoti šią esminę vertę: per kelis veiksmus padėti naudotojui suprasti, ką jis turi, ar tai kokybiška, ką su tuo daryti ir kaip tai komunikuoti. Ši analizė bus pagrindas konkrečiai techninei užduočiai, dizaino prototipavimui ir MVP plano sudarymui.

# 2. Naudotojų analizė

## 2.1 Įvadas

Remiantis rinkos tyrimu ir naudotojų scenarijų analize, AdoreIno sprendimas orientuotas į kelis aiškiai apibrėžtus naudotojų tipus. Kiekviena iš šių grupių susiduria su skirtingais iššūkiais, tačiau juos vienija bendra problema – **negalėjimas greitai, objektyviai ir suprantamai įvertinti jau egzistuojančios IT sistemos būklės ir vertės**.

**Pagrindiniai naudotojų archetipai:**

|  |  |  |
| --- | --- | --- |
| **Naudotojo tipas** | **Aprašymas** | **Pagrindinis poreikis** |
| **Verslo savininkas (paprasta sistema)** | Turi TVS pagrindu veikiančią svetainę ar el. parduotuvę | Nori sužinoti, ar sistema patikima ir ką daryti toliau |
| **Verslo savininkas (sudėtinga sistema)** | Valdo sudėtingą IT sistemą, dažnai be techninio pagrindo | Nori įvertinti būklę, planuoti atnaujinimus ar migraciją |
| **IT specialistas** | Perima svetimą kodą, turi rasti rizikas ir greitai susivokti | Reikia struktūros, AI komentarų ir rekomendacijų |
| **IT agentūra** | Prižiūri daug sistemų, dažnai perima projektus be dokumentacijos | Reikia efektyviai įsigilinti, pagrįsti kainodarą klientui |
| **Rinkodaros specialistas** | Naudoja sistemą komunikacijai, kampanijoms, bet neturi techninių žinių | Nori žinoti, ar sistema „sveika“ ir ar ją galima naudoti laisvai |

**Bendra elgsena ir tikslai**

Visi šie naudotojai, nepriklausomai nuo jų kompetencijos ar vaidmens, ieško:

* aiškumo – *kas yra sistemoje ir kaip ji veikia*;
* skaidrumo – *ar tiekėjas dirba kokybiškai*;
* rekomendacijų – *ką su sistema daryti toliau*;
* paprastos komunikacijos – *kaip apie tai kalbėti su komanda, klientu ar vadovu*.

**Vertė maketo lygiu**

Maketas turi būti pakankamai funkcionalus, kad:

* kiekvienas iš šių naudotojų galėtų įvesti realų pavyzdį (failą, repo, svetainę),
* gautų reikšmingą ir interpretuojamą rezultatą (aiškiai, jo kalba),
* pasinaudotų AI išvadomis ne tik skaitymui, bet ir sprendimui priimti ar argumentuoti.

## 2.2 Scenarijaus analizė: Verslo savininkas (paprasta sistema)

**Kontekstas:**  
Smulkaus verslo savininkas valdo WordPress ar PrestaShop pagrindu veikiančią svetainę, kurią sukūrė samdomas programuotojas ar agentūra. Laikui bėgant sistema tapo mažiau suprantama, neaišku, ar ji kokybiška, ar reikia investicijų, o tiekėjo patikimumas – abejotinas.

**Naudotojo kelias (User Journey – 10 žingsnių)**

1. **Suformuluoja klausimą / problemą:**  
   Ar mano sistema gera, šiuolaikiška ir tinkama naudoti toliau?
2. **Ieško, kam galėtų užduoti šį klausimą:**  
   Klausia IT pažįstamų, agentūrų, gauna prieštaringus atsakymus.
3. **Išgirsta apie AdoreIno arba randa per Google/plugin‘ą:**  
   Sužino, kad tai įrankis, padedantis įvertinti IT sistemą automatiškai.
4. **Apsilanko AdoreIno platformoje arba aktyvuoja plugin‘ą:**  
   Pradeda analizę: įkelia failus, paspaudžia „Pradėti vertinimą“, ar inicijuoja iš WordPress panelės.
5. **Sistema automatiškai atlieka analizę:**  
   Patikrina svetainės struktūrą, plugin’us, kodų tvarkingumą, atnaujinimus.
6. **Gauta ataskaita – aiški ir vizuali:**  
   Spalvomis pažymėti elementai: rizikos vietos, neoptimizuotas kodas, pasenusios integracijos.
7. **Matomas bendras sistemos vertinimas:**  
   Pvz.: „Stabilu, bet rekomenduojamas atnaujinimas“ arba „Didelė rizika, būtinas peržiūrėjimas“.
8. **Gauna konkrečias rekomendacijas, ką daryti:**  
   Pvz.: „Atnaujinti kontaktų formą“, „Pakeisti temą“, „Optimizuoti nuotraukų krovimą“.
9. **Eksportuoja arba pasidalina ataskaita su tiekėju:**  
   Naudotojas siunčia ją savo programuotojui arba sprendžia, ar jį keisti.
10. **Priima sprendimą dėl tolimesnių veiksmų:**  
    Remdamasis ataskaita, pasirenka palaikyti, atnaujinti ar pakeisti svetainę.

**Svarbiausios įžvalgos iš scenarijaus:**

* Verslo naudotojas **neturi techninio pasirengimo**, bet nori objektyvios informacijos.
* Labiausiai vertina **aiškumą, vizualumą ir rekomendacijas sprendimams**.
* Reikia labai **žemo įėjimo barjero**: be paskaitų, be mokymosi, be IT kalbos.

## 2.3 Scenarijaus analizė: Verslo savininkas (sudėtinga sistema)

**Kontekstas:**  
Vidutinės ar sparčiai augančios įmonės vadovas ar produkto savininkas valdo unikalų, individualiai kurtą IT sprendimą – pvz., klientų portalą, užsakymų valdymo sistemą ar SaaS platformą. Per kelerius metus sistema tapo sudėtinga, vystoma keliais etapais, ją keitė keli programuotojai ar agentūros. Verslo pusė nebeturi aiškaus supratimo apie tai, kiek sistema „gyva“, kiek ji „trukdo“ ar „stabdo“.

**Naudotojo kelias (User Journey – 10 žingsnių)**

1. **Susiduria su sprendimo būtinybe:**  
   Reikia įvertinti sistemą prieš investiciją, pokyčius, tiekėjo keitimą arba naujos funkcijos diegimą.
2. **Konstatuoja informacijos trūkumą:**  
   Neturi dokumentacijos, kodas painus, vidinė IT komanda negali greitai pateikti atsakymo.
3. **Surenka projektą (repo, failus, serverio kopiją) ir inicijuoja analizę AdoreIno sistemoje:**  
   Ją atlieka jis pats arba paveda IT vadovui.
4. **Sistema atlieka struktūrinę ir loginę analizę:**  
   Atpažįsta modulius, jų priklausomybes, loginę architektūrą, senus komponentus, duplications.
5. **Gauta analizės ataskaita su semantiniais paaiškinimais:**  
   Pvz.: „Užsakymų modulio struktūra nelogiška, daug persidengiančių komponentų“, „Yra nenaudojamų funkcijų, bet jos vis dar palaikomos“.
6. **Ataskaitoje vizualiai išskirtos rizikos zonos:**  
   Su pažymėtais AI „įspėjimais“ ir argumentais dėl jų svarbos.
7. **Verslo kalba pateikta analizės santrauka:**  
   Pvz.: „Sistemą galima palaikyti dar 12–18 mėn., bet būtina strateginė refaktoringo peržiūra“.
8. **Naudotojas gauna rekomendacijas dėl tolimesnių žingsnių:**  
   Pateikiamos strateginės alternatyvos – palaikyti, refaktoruoti, migruoti, atsisakyti.
9. **Analizės išvados integruojamos į vadybinę diskusiją:**  
   Su vadovybe, investuotojais, partneriais ar IT tiekėju.
10. **Priimamas pagrįstas sprendimas dėl sistemos ateities:**  
    Sprendimas nebe remiasi nuomone ar spėjimu – jis grįstas AdoreIno analize.

**Svarbiausios įžvalgos iš scenarijaus:**

* Šio tipo naudotojui svarbiausia ne detalės, o **bendras sistemos vaizdas ir sprendimo pasekmės**.
* Reikalingas **duomenimis grįstas strateginis argumentas**: ką daryti su šia sistema.
* Naudotojas dažnai deleguoja techninį veiksmą, bet **priima sprendimą pats** – jam reikia verslui pritaikytos išvados.

## 2.4 Scenarijaus analizė: Programuotojas / IT specialistas

**Kontekstas:**  
Programuotojas, dažniausiai freelanceris arba IT specialistas agentūroje, perima projektą be dokumentacijos, kurį anksčiau vystė kita komanda ar tiekėjas. Projektas parašytas nepažįstamu stiliumi, architektūra neaiški, pasitaiko „pataisymų ant pataisymų“. Klientas prašo greitai atlikti pakeitimą, bet programuotojas nežino, kiek rizikinga jį daryti.

**Naudotojo kelias (User Journey – 10 žingsnių)**

1. **Gautas projektas perėmimui:**  
   Pvz., WordPress svetainė, Laravel sistema arba nestandartinis PHP projektas be dokumentacijos.
2. **Atidarius kodą – chaosas:**  
   Neaiški struktūra, nėra komentarų, nenuosekli logika. Peržiūra rankiniu būdu reikalautų kelių valandų.
3. **Prisijungiama prie AdoreIno:**  
   Projektas įkeliamas tiesiai arba analizuojamas per API / plugin.
4. **Sistema automatiškai atlieka analizę:**  
   Struktūros detekcija, komponentų suskirstymas, priklausomybių ir rizikos zonų nustatymas.
5. **Gauta techninė ataskaita su AI komentarais:**  
   Pvz.: „Ši funkcija per daug priklausoma nuo XYZ“, „Kodų dublikavimas modulyje ABC“.
6. **Matomos konkrečios rizikos vietos:**  
   Pažymėtos „karštos“ sritys: neaiškus kodas, aukštas sudėtingumo lygis, dažnai keisti failai.
7. **Sistema siūlo rekomendacijas:**  
   Refaktoringo vietos, testavimo prioritetai, dokumentuotini taškai.
8. **Programuotojas peržiūri ir koreguoja AI siūlymus:**  
   Pažymi, kas aktualu, papildomai komentuoja – ruošiasi bendrauti su klientu ar vadovu.
9. **Sugeneruojama suvestinė ataskaita:**  
   Naudojama kaip pagrindimas kainodarai, darbų apimties įvertinimui, pasiūlymui.
10. **Sprendimai priimami greičiau ir mažiau rizikingai:**  
    Programuotojas žino, ką keisti, ko neliesti, ir gali pagrįsti sprendimus kitai pusei.

**Svarbiausios įžvalgos iš scenarijaus:**

* Programuotojas siekia **greito ir patikimo įsigilinimo**, kad galėtų dirbti su svetimu kodu be klaidų.
* Reikia tikslios, aiškiai išskaidytos informacijos apie **architektūrą, priklausomybes ir rizikas**.
* AI paaiškinimai ir rekomendacijos – tai, kas leidžia sutaupyti valandas analizei ir pagerina komunikaciją su klientu.

## 2.5 Scenarijaus analizė: IT agentūra

**Kontekstas:**  
IT agentūra dirba su dešimtimis klientų, dažnai perima svetimas sistemas – tiek paprastas (WordPress, Shopify), tiek sudėtingesnes (Laravel, custom PHP, integruotos platformos). Klientai reikalauja greitų pataisymų, optimizavimo ar funkcionalumo plėtros. Projektų dokumentacija retai būna išsami. Agentūra turi greitai įsigilinti, įvertinti sistemą, pagrįsti darbų kainą ir sudėtingumą – tai tiesiogiai veikia jų darbo efektyvumą ir reputaciją.

**Naudotojo kelias (User Journey – 10 žingsnių)**

1. **Gaunamas naujas klientas / projektas:**  
   Pvz.: svetainė, kurios ankstesnis tiekėjas „dingo“ arba nebetenkina.
2. **Komanda neturi informacijos apie sistemos struktūrą:**  
   Dokumentacijos nėra, tik FTP / repo / duomenų bazės eksportas.
3. **IT specialistas įkelia projektą į AdoreIno arba integruoja per plugin/API:**  
   Prasideda automatinė analizė – nereikia iš karto skaityti kodo rankomis.
4. **Sistema suskaido projektą į loginius modulius:**  
   Nustatomi priklausomybės ryšiai, rizikos taškai, pasikartojantys komponentai.
5. **Pateikiama techninė ir verslo kalba suformuluota ataskaita:**  
   Ką sistema daro, kur silpnos vietos, kas turėtų būti peržiūrėta ar perrašyta.
6. **Komanda peržiūri AI komentarus, pažymi, kas aktualu:**  
   Galima pridėti savo pastabas, priskirti darbų apimtį, įtraukti į užduočių sąrašą.
7. **Projektų vadovas suformuoja kainodaros pasiūlymą klientui:**  
   Pagrįstas analize – su įžvalgomis, AI paaiškinimais, bendrais vertinimais.
8. **Klientas gauna aiškų pagrindimą darbams:**  
   Sumažėja pasipriešinimas, padidėja pasitikėjimas.
9. **Analizės rezultatai naudojami palaikymo etape:**  
   Pvz.: kas 6 mėn. kartojama analizė, stebima sistema, fiksuojama pažanga.
10. **Agentūra optimizuoja savo darbo procesą:**  
    Mažiau laiko įsigilinimui, aiškesnė komunikacija, greitesnis sprendimų priėmimas.

**Svarbiausios įžvalgos iš scenarijaus:**

* IT agentūrai būtinas ne tik įrankis analizei, bet ir **komunikacijos tiltas su klientu**.
* Svarbu, kad sprendimas veiktų **greitai, patikimai ir būtų „proof-of-work“ tipo**.
* AdoreIno leidžia išvengti „nuo nulio“ analizės kiekvieną kartą – tai tiesioginė vertė agentūros pelningumui.

## 2.6. Santrauka: naudotojų scenarijų analizės įžvalgos

Atlikta naudotojų scenarijų analizė leidžia išskirti keletą universalių AdoreIno naudotojų poreikių ir elgsenos modelių, kurie tiesiogiai formuoja produkto maketo reikalavimus.

**Pagrindinės įžvalgos:**

1. **Visi naudotojai siekia aiškumo apie egzistuojančią sistemą.**  
   Nepriklausomai nuo jų techninės kompetencijos, naudotojai nori atsakyti į tą patį esminį klausimą: *kas yra šioje sistemoje, ar ji tvarkinga, ir ką su ja daryti toliau*.
2. **Analizės rezultatai turi būti suprantami pagal naudotojo profilį.**  
   Techniniams naudotojams reikalinga detalė analizė (struktūra, rizikos, refaktoringas), o netechniniams – supaprastinta, verslo kalba pateikta ataskaita, padedanti priimti sprendimus be IT tarpininko.
3. **Pagrindinė vertė – įsigilinimo laiko sutrumpinimas.**  
   Visi scenarijai rodo, kad šiuo metu daugiausia sąnaudų eikvojama „įėjimui“ į sistemą. AdoreIno turi padėti sumažinti šį barjerą iki minučių, o ne valandų ar dienų.
4. **Sprendimų pagrįstumas yra tiek vidinis, tiek išorinis tikslas.**  
   Verslo naudotojai nori įsitikinti, kad IT tiekėjas dirba efektyviai, o IT specialistai – kad jų pasiūlymai yra matomi kaip pagrįsti. AdoreIno atlieka „verifikatoriaus“ vaidmenį.
5. **Daugelis naudotojų dirba su svetimais sprendimais.**  
   Perėmimas yra norma, ne išimtis. Tai reiškia, kad sprendimas turi būti pritaikytas dirbti su paveldėtomis, nestandartinėmis, dokumentacijos neturinčiomis sistemomis.

**Išvada maketui:**

Maketas turi pademonstruoti vieną esminę produkto funkciją – galimybę greitai, aiškiai ir patikimai suprasti sistemos būklę bei pateikti rekomendacijas, pritaikytas konkrečiam naudotojo profiliui. Tai yra kertinis vertės įrodymas, nuo kurio priklausys tolimesnis produkto vystymas.

# 3. Funkciniai reikalavimai

Šiame skyriuje pateikiami pagrindiniai funkciniai reikalavimai AdoreIno maketui. Jie sudaryti remiantis naudotojų scenarijų analize, identifikuotomis problemomis bei vertės pasiūlymu. Reikalavimai skirti tam, kad maketas galėtų:

* pademonstruoti aiškią produkto vertę realiose situacijose;
* užfiksuoti naudotojų reakciją ir surinkti grįžtamąjį ryšį;
* ištestuoti pagrindines funkcines sąsajas tarp analizės, AI interpretacijos ir naudotojo sprendimų.

## 3.1. Pagrindiniai funkciniai moduliai ir naudotojų poreikiai

**1. Verslininkas: noriu žinoti, ką turiu**  
Sistema turi gebėti automatiškai įvertinti, ar sprendimas yra:

* technologiškai kokybiškas;
* konkurencingas rinkos kontekste;
* šiuolaikiškas (pagal naudotas bibliotekas, praktiką);
* ir ar pasiekia savo numatytus tikslus (pvz., svetainė veikia, bet ar realiai naudojama?).

**2. Verslininkas: noriu matyti techninę struktūrą**  
Turėtų būti pateiktas aiškus sistemos vaizdas – modulių medis, priklausomybės, komponentų sąveika.

**3. Verslininkas: noriu žinoti palaikymo poreikius**  
Automatinis sistemos palaikymo ir techninės skolos įvertinimas: palaikymo užduotys, pasenę elementai, nenaudojami komponentai, nestabilios vietos.

**4. Verslininkas: noriu įvertinti naujus pasiūlymus**  
Sistema turėtų padėti palyginti: dabartinę būklę su siūlomais pokyčiais, jų rizikas, kaštus ir potencialią naudą (pvz.: pasiūlymas persidaryti ant naujos platformos (pvz.: shipify)).

**5. Programuotojas: noriu pagrįsti savo darbo kokybę**  
Sistema turi sugeneruoti AI pagrįstą techninio darbo vertinimą, leidžiantį parodyti tvarkingą struktūrą, standartų laikymąsi, įdiegtas pataisas.

**6. Programuotojas: noriu pagrįsti palaikymo užduotis**  
Ataskaita turi padėti nurodyti, kurios vietos reikalauja priežiūros, kodėl tai svarbu, ir kokios būtų pasekmės jų neatlikus.

**7. Programuotojas: noriu pagrįsti papildomų darbų kainą**  
Sistema turi padėti parengti pasiūlymą klientui: su technine argumentacija ir verslui suprantamais AI paaiškinimais.

**8. Autopogramuotojas: noriu visapusiškai suprasti sistemą**  
Jei sistemoje veikia AI agentas, jam būtina turėti struktūrizuotą sistemos vaizdą – kad galėtų priimti sprendimus dėl kodo kūrimo ir diegimo (atlikti automatinio palaikymo funkciją).

## 3.2. MVP funkcijų santrauka

Maketo etape būtina įgyvendinti šias funkcijas, kad būtų galima įrodyti vertę:

* Projektų įkėlimas arba integracija;
* Automatinė kodo ir struktūros analizė;
* Sistemos vertės, kokybės, šiuolaikiškumo įvertinimas;
* AI paaiškinimai ir verslo kalba pateikiama analizės santrauka;
* Vizuali ataskaita su rekomendacijomis;
* Galimybė bendrinti analizės rezultatą su trečiosiomis šalimis (tiekėjais, kolegomis, vadovais).

# 4. Ne-funkciniai reikalavimai

Nors pagrindinę AdoreIno produkto vertę kuria funkcinės galimybės – tokios kaip analizė, paaiškinimas ir sprendimų pagrindimas – ne mažiau svarbus yra ir tai, **kaip naudotojas patiria šias funkcijas**. Ne-funkciniai reikalavimai formuoja produkto kokybę, pasitikėjimą, suprantamumą ir įvedimo lengvumą, todėl jie yra esminė maketo vertės dalis.

Šie reikalavimai apima naudotojo patirtį (UX), veikimo charakteristikas, saugumą, prieinamumą ir architektūrinį lankstumą.

## 4.1. Naudotojo patirties reikalavimai (UX)

**a) Aiškumas ir paprastumas**

Naudotojai – ypač verslo segmentas – neturi techninių žinių, todėl analizės rezultatai turi būti pateikti verslo kalba, aiškiai ir struktūruotai. UI turi atitikti „skaitymo, o ne mokymosi“ principą.

**b) Vizualumas ir semantinis dizainas**

Techninės sąvokos turi būti pateikiamos su kontekstiniais paaiškinimais, vizualiniais indikatoriais (pvz., šviesoforo principu, rizikos skalėmis). Analizė turi būti ne „kodo siena“, o loginė vizualizacija.

**c) Pasitikėjimo formavimas**

AI išvados turi būti pateikiamos su „paaiškinamumu“ (explainability) – naudotojas turi žinoti, kodėl gavo vienokį ar kitokį vertinimą. Pvz., „Sistema vertinama kaip pasenusi, nes 43 % kodo priklauso nuo senų bibliotekų“.

**d) Mažas įėjimo barjeras**

Produktas turi veikti be būtinybės peržiūrėti dokumentaciją. Procesas – nuo analizės pradžios iki ataskaitos peržiūros – turi būti įgyvendinamas intuityviai, per kelis žingsnius.

## 4.2. Veikimo reikalavimai

**a) Greitis**

Analizė turi būti atlikta per realistišką laiką – mažesnėms sistemoms (iki 100 failų) per 1–3 minutes, sudėtingesnėms – iki 10 minučių.

**b) Patikimumas**

Sistema turi veikti stabiliu režimu, su atstatymo galimybėmis analizės metu (pvz., jei nutrūksta įkėlimas ar AI pertraukia užklausą).

**c) Skalavimas**

Architektūra turi būti paruošta apdoroti kelis projektus vienu metu, tiek MVP etape, tiek ateityje (multi-tenant schema).

## 4.3. Saugumo reikalavimai

**a) Naudotojo duomenų sauga**

Įkeltos bylos ar projektų duomenys turi būti saugomi trumpai (tik analizei), su aiškiu saugojimo trukmės deklaravimu. Duomenys neturi būti perduodami tretiesiems asmenims.

**b) AI sąveikos apsauga**

Jei naudojami LLM modeliai (pvz., OpenAI), turi būti įgyvendintos priemonės užklausų pseudonimizavimui ir duomenų neindeksavimui (pvz., neleidžiama naudoti duomenų modelio treniravimui).

**c) Teisių valdymas ir ribojimai**

Naudotojai turi matyti tik savo duomenis, projektus ar ataskaitas. Jei naudojami komandų režimai – privaloma teisėmis grįsta prieiga.

## 4.4. Prieinamumo ir tarptautiškumo reikalavimai

**a) Kalbinis universalumas**

Net jei pirmoji versija bus tik anglų kalba, architektūroje turi būti numatyta lokalizavimo (i18n) struktūra – kad vėliau būtų galima lengvai išplėsti kalbų skaičių.

**b) Mobilumo reikalavimai**

Maketas orientuotas į darbą per kompiuterį, tačiau analizės peržiūra turėtų būti įmanoma per mobilią naršyklę (skaitymui).

**c) Prieinamumo gairės (A11Y)**

Vizualiniai komponentai (spalvos, kontrastai, tooltip’ai) turi atitikti bazinius prieinamumo reikalavimus – ypač naudotojams su regos sutrikimais.

## 4.5. Architektūrinio lankstumo reikalavimai

* Vartotojų valdymas turi būti atskirtas nuo analizių sistemos – tai leis integruoti kitas autentifikacijos priemones (OAuth, SSO).
* AI komponentas turi būti palaikomas kaip modulinis – kad būtų galima keisti modelį (pvz., OpenAI → local model).
* Būtina API-first struktūra, leidžianti prieigą per naršyklę, CMS plugin, naršyklės extension ar CLI.

# 5. Techniniai komponentai

Šiame skyriuje aprašomi pagrindiniai techniniai komponentai, kuriuos turi apimti AdoreIno maketas. Jie išvedami iš funkcinių ir ne-funkcinių reikalavimų bei planuojamos architektūros krypties – SaaS modelio, AI interpretacijos naudojimo ir kelių naudotojų profilių aptarnavimo.

Techniniai komponentai suskirstyti į loginius sluoksnius – nuo naudotojo sąsajos iki analizės branduolio ir integracijų.

**5.1. Naudotojo sąsajos sluoksnis (UI)**

* Web pagrindu veikianti vartotojo sąsaja.
* Lengvai suprantama net netechniniam naudotojui (business-first UX).
* Vizualūs elementai: modulių medis, priklausomybių žemėlapis, spalvinis kodavimas.
* Verslo santraukos ir ataskaitų peržiūra (PDF peržiūra, ataskaitos eksportas).

**5.2. Duomenų įkėlimo / integracijos sluoksnis**

* Failų įkėlimas (ZIP, .tar, .gz).
* GitHub ar GitLab repo integracija per API (read-only).
* Plugin struktūra (WordPress, PrestaShop) – kaip alternatyvus įkėlimo metodas.
* Naršyklės plėtinys (Extension), galintis inicijuoti analizę.

**5.3. Analizės branduolys (Core Analysis Engine)**

* Failų struktūros analizatorius (map, grupavimas, filtrai).
* Rizikos taškų detektorius: pasenę moduliai, dažnai keičiami failai, techninė skola.
* Techninių metrikų analizatorius: failų dydžiai, keitimo dažnumas, ciklinis sudėtingumas.
* Sistemos struktūros rekonstrukcija: priklausomybių grafas, loginiai blokai.

**5.4. AI interpretacijos modulis**

* AI naudojamas generuoti:
  + Failų ar modulių paaiškinimus;
  + Sistemos kokybės, konkurencingumo ir šiuolaikiškumo įvertinimus;
  + Verslo kalba pateikiamas rekomendacijas.
* LLM prieiga per API (pvz., OpenAI, Claude, arba lokaliniai modeliai).
* AI atsakymų „explainability“ sluoksnis – kodėl tokia rekomendacija pateikta.

**5.5. Projektų / naudotojų valdymo sluoksnis**

* Projektų kūrimas, saugojimas ir būsenų valdymas.
* Naudotojo paskyra, autentifikacija (per e. paštą, OAuth).
* Rolės ir prieigos lygiai (admin, peržiūrėtojas, komandos narys).
* Analizių istorija, rezultatų versijos, veiksmų žurnalas.

**5.6. Duomenų saugykla ir kvotų valdymas**

* Analizių rezultatai saugomi duomenų bazėje (pvz., PostgreSQL, MongoDB).
* Kvotos matuojamos per analizės įvykius, projektų kiekį, naudojimo laiką.
* Laikini duomenys (pvz., failai) laikomi saugiai, automatiškai pašalinami pagal politiką.

**5.7. API ir integracijų sluoksnis**

* REST / GraphQL API: projektų valdymui, analizės inicijavimui, rezultatų peržiūrai.
* Galimybė integruoti su:
  + projektų valdymo įrankiais (Jira, Linear);
  + Git įrankiais (GitHub, GitLab, Bitbucket);
  + CMS platformomis (WordPress, PrestaShop).

**5.8. Saugumo komponentai**

* Naudotojų autentifikacija ir teisės.
* Duomenų pseudonimizavimas (jei siunčiama į AI API).
* HTTPS, įrašų žurnalai, atskirti duomenų konteineriai.
* Duomenų laikymo trukmės valdymas (configurable TTL).

# 6. MVP apimtis

MVP (Minimum Viable Product) tikslas – kuo greičiau ir efektyviau patikrinti, ar AdoreIno sprendimas kuria vertę naudotojams, ir kokiomis sąlygomis jis gali būti taikomas praktikoje. Šiame etape siekiama ne maksimaliai išplėsti funkcionalumą, o **įrodyti pagrindinį produkto principą – kad AI pagrindu sukurta analizė gali padėti suprasti, įvertinti ir pagrįsti sprendimus dėl turimos IT sistemos**.

**6.1. Pagrindiniai MVP tikslai**

* **Pateikti aiškų, naudotojui suprantamą sistemos vertinimą** (kokybė, šiuolaikiškumas, vertė).
* **Sukurti analizės logiką**, kuri geba struktūrizuoti kodą į modulius, nustatyti rizikas ir trūkumus.
* **Integruoti AI paaiškinimų sluoksnį**, kuris pateikia įžvalgas verslo kalba.
* **Leisti naudotojui greitai pradėti ir gauti rezultatą**, be mokymosi ar techninio barjero.
* **Įgalinti rezultatų peržiūrą ir bendrinimą**, kad jie galėtų būti naudojami sprendimų priėmimui ar komunikacijai.

**6.2. MVP funkcijų rinkinys**

|  |  |
| --- | --- |
| **Funkcija** | **Tikslas** |
| **Projektų įkėlimas (failai arba repo)** | Užtikrinti realios analizės pradžią naudotojo duomenimis. |
| **Struktūrinė analizė (moduliai, priklausomybės)** | Parodyti, kaip sistema „išskaidoma“ į suprantamą formą. |
| **Techninės skolos / rizikų nustatymas** | Aptikti silpnas vietas, pasenusius komponentus. |
| **AI paaiškinimai (modulio/failo funkcija)** | Pateikti žmogaus kalba, ką daro kodas ir kodėl tai svarbu. |
| **Verslo ataskaitos santrauka** | Vizualiai ir suprantamai perteikti analizės rezultatus. |
| **Rekomendacijos: ką daryti toliau** | AI siūlymai, ką optimizuoti, taisyti ar keisti. |
| **Rezultatų bendrinimas** | Eksportas arba peržiūros nuoroda, naudojimui komunikacijoje. |

**6.3. Neprivalomos, bet naudingos MVP funkcijos (jei leidžia resursai)**

* **Plugin integracija (WordPress, PrestaShop)**  
  Leidžia analizuoti svetaines be kodo įkėlimo – vienu mygtuku.
* **Analizių versijų istorija**  
  Galimybė palyginti rezultatus prieš ir po pokyčių.
* **„Confidence level“ žymėjimas AI atsakymams**  
  Padeda suprasti, kiek AI išvada yra pagrįsta (įveda pasitikėjimo lygį).

**6.4. MVP sėkmės kriterijai**

* Naudotojas sugeba **pats, be pagalbos**, atlikti analizę ir suprasti rezultatą.
* Rezultatai naudojami priimant realų sprendimą (pvz., kainodarai, tiekėjo vertinimui).
* Gauta analizė laikoma naudinga tiek **verslo**, tiek **techniniam** naudotojui.
* Grįžtamasis ryšys patvirtina, kad **pagrindinė vertė buvo suprasta ir įvertinta**.

# 7. Roadmap: tolesnės versijos

Sėkmingai validavus MVP vertės pasiūlymą, AdoreIno produktas gali būti plėtojamas etapais, kurių kiekvienas išplečia funkcionalumą, plečia naudotojų segmentus ir didina produkto vertę tiek techniniu, tiek verslo požiūriu. Šiame skyriuje nurodomos pagrindinės kryptys, kurios galėtų būti įgyvendinamos antroje ar tolimesnėse versijose.

**7.1. Integracijos**

* **CI/CD ir Git platformų integracijos (GitHub, GitLab, Bitbucket):**  
  Leidžia AdoreIno naudoti kaip automatinį vertinimo etapą kiekviename commit’e ar pull request’e.
* **Projektų valdymo įrankiai (Jira, Linear):**  
  Galimybė perkelti išvadas į užduotis, stebėti sprendimų įgyvendinimą.
* **IDE plugin’ai (VS Code ir kt.):**  
  Programuotojai gali gauti analizės įžvalgas tiesiai kodo rašymo aplinkoje.

**7.2. Funkcijų plėtra**

* **Komandinis darbas / rolės:**  
  Projektų bendrinimas su komanda, komentarai, rolės („skaitytojas“, „analitikas“, „vadovas“).
* **Analizių palyginimas (versijų istorija):**  
  Galimybė matyti, kaip sistema keitėsi per laiką; prieš ir po refaktoringo, atnaujinimo ar tiekėjo keitimo.
* **„What-if“ analizė / simuliacijos:**  
  Pvz., kas nutiktų, jei pašalintume šį modulį? Kokia rizika?
* **Templatuotas vertinimas / scoring modeliai:**  
  Skirtingi profiliai (pvz., „e-komercija“, „paslaugų svetainė“) su unikaliu vertinimo standartu.

**7.3. AI komponentų evoliucija**

* **Paaiškinimų kokybės diferencijavimas:**  
  Galimybė įjungti „verslo“, „techninį“ arba „detalų“ AI atsakymų lygį.
* **AI agentas palaikymui ir dialogui:**  
  Pokalbio tipo sąsaja, kur AI veda naudotoją per analizę („Ką nori sužinoti?“ → „Štai ką siūlau“).
* **Lokalūs AI modeliai (on-premise arba open-source):**  
  Alternatyva OpenAI/Claude – naudotojams su privatumo ar saugos ribojimais.

**7.4. Monetizavimo ir produktinio modelio plėtra**

* **Planų valdymas (freemium, per analizę, per projektą, per komandas):**  
  Išplečiama kvotų sistema ir galimybės naudotojams rinktis pagal poreikį.
* **White-label arba partnerių sprendimai:**  
  Agentūros ar platformos gali naudoti AdoreIno su savo ženklais / integracijose.
* **Sertifikavimo / atitikimo modulis:**  
  AI galėtų pateikti „atitikimo“ įvertinimą: ar sistema atitinka saugumo, priežiūros ar kokybės reikalavimus (pvz., viešieji pirkimai, ISO).

**7.5. Infrastruktūros išplėtimas**

* **Projektų / analizių saugojimo optimizavimas (ilgesni ciklai):**  
  Pvz., naudojant archyvavimo logiką, ilgalaikę analizės versijų saugyklą.
* **Vartotojų administravimas (SaaS įrankis agentūroms):**  
  Leidžiama agentūroms valdyti klientų projektus, naudotojus, sąskaitas vienoje platformoje.

**Išvada:**  
MVP tikslas – įrodyti vertę. Roadmap tikslas – **plėsti tą vertę, nekeičiant esminės AdoreIno prigimties:** padėti suprasti, ką turi, ir priimti sprendimus dėl to, ką daryti toliau. Tolimesnės versijos leistų AdoreIno tapti ne tik naudotojo asistentu, bet ir **visaverčiu sprendimų valdymo moduliu** programinės įrangos gyvavimo cikle.

# 8. Rizikos

Nors AdoreIno idėja grindžiama aiškiu rinkos poreikiu ir inovatyvia vertės logika, maketo kūrimas susiduria su keliomis reikšmingomis rizikomis. Jos skirstomos į keturias pagrindines kategorijas: technologines, vartotojų elgsenos, vertės suvokimo ir vystymo.

**8.1. Technologinės rizikos**

**a) AI atsakymų kokybė ir (ne)tikslumas**  
LLM modeliai gali generuoti klaidingus ar nepakankamai tikslius paaiškinimus. Verslo naudotojui tai gali atrodyti kaip sistema, kuria negalima pasitikėti.

*Švelninimas:*  
– naudoti „confidence level“ žymas,  
– pateikti AI paaiškinimus kartu su pagrindimu,  
– leisti vartotojui patvirtinti / atmesti AI įžvalgas.

**b) Kompleksiškos sistemos analizės apribojimai**  
Struktūrinė analizė gali būti netiksli sistemoms su dinamine logika, modulių „hackais“ ar hibridiniu kodu.

*Švelninimas:*  
– MVP koncentruoti į paprastesnes struktūras (pvz., WordPress, Laravel),  
– riboti analizės apimtį ir pateikti rezultatų „tikslumo ribą“.

**8.2. Vartotojų elgsenos rizikos**

**a) Per aukšti lūkesčiai AI sprendimams**  
Verslo naudotojai gali tikėtis „vieno mygtuko sprendimo“ be konteksto arba manyti, kad AI pateikia neklystančią tiesą.

*Švelninimas:*  
– edukaciniai paaiškinimai per UI,  
– akcentas: „padeda priimti sprendimą, bet nepakeičia žmogaus“.

**b) Nepasitikėjimas AI arba nesupratimas**  
Ypač B2B kontekste svarbus pasitikėjimas – jei paaiškinimas neaiškus, jis nebus priimtas.

*Švelninimas:*  
– verslui skirta analizės santrauka,  
– human-in-the-loop logika (vartotojas visada turi galutinį žodį).

**8.3. Vertės suvokimo rizikos**

**a) Verslo naudotojas nemato vertės be techninio pagrindo**  
Gali būti sunku parodyti, kad „analizė yra verta 50–100 €“, jei nematyti akivaizdžių pokyčių.

*Švelninimas:*  
– pateikti aiškų „sprendimo pasirinkimo“ pagrindimą,  
– įtraukti ataskaitų dalį „ką tai reiškia Jūsų verslui“.

**b) Programuotojai gali jausti konkurenciją arba „kontrolės“ grėsmę**  
Kai kurie specialistai gali manyti, kad AI analizuoja ir vertina jų darbą.

*Švelninimas:*  
– pozicionuoti įrankį kaip pagalbą, ne kaip kontrolės priemonę,  
– AI išvadas leidžiama komentuoti, papildyti ar atmesti.

**8.4. Vystymo rizikos**

**a) Per didelė maketo apimtis**  
Norint parodyti per daug – kyla rizika nepabaigti, neįgyvendinti MVP.

*Švelninimas:*  
– aiškus MVP tikslas: vertės demonstravimas vienu naudotojo keliu,  
– funkcijų prioritetizavimas.

**b) Technologinė priklausomybė nuo konkretaus AI tiekėjo (pvz., OpenAI)**  
Modelių pasikeitimas, prieinamumo ribojimai ar kainodaros pokyčiai gali paveikti produktą.

*Švelninimas:*  
– architektūrinis AI sluoksnio abstrahavimas (pasiruošimas kitam modelio tiekėjui),  
– galimybė pereiti prie vietinio modelio (open-source) ilgainiui.

**Išvada**

Rizikos yra realios, tačiau jos yra žinomos, valdomos ir gali būti sprendžiamos jau maketo etape. Jas būtina įtraukti į testavimo planą, UX dizaino sprendimus ir MVP planavimo eigą – kad būtų užtikrinta ne tik produkto funkcija, bet ir vartotojo pasitikėjimas bei technologinis tvarumas.