VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Praktikos ataskaita

Praktiką atliko : Vismantas Stonkus

Universiteto praktikos vadovas : Prof. dr. Remigijus Paulavičius

Praktikos institucija : Humbility, UAB

Organizacijos praktikos vadovas : Mantas Sakalauskas

Technologijų vadovas (CTO)

Organizacijos praktikos vadovo įvertinimas : 10

Vilnius
2025

Turinys

Įvadas

Motyvacija

Perėjimas iš tradicinės korporacinės įmonės į dinamišką *Humbility, UAB* startup'ą buvo sąmoningas karjeros pasirinkimas. Ankstesnėje darbo vietoje susidūriau su išsibranžiusiais procesais ir lėtu sprendimų priėmimu, kuris ribojo techninę pažangą. *Humbility* pasiūlė visiškai priešingą aplinką - greitus iteracijos ciklus, tiesioginę įtaką produkto vystymui ir galimybę dirbti su pažangiausiomis didelio dažnio prekybos technologijomis. Šis kontekstas tapo idealiu testavimo lauku įgytoms žinioms pritaikyti praktikoje.

Praktikos tikslai ir uždaviniai

Pagrindinė praktikos tema - kriptovaliutų prekybos sistemų kūrimas ir tobulinimas naudojant Go bei Rust programavimo kalbas, su specialiu dėmesiu likvidacijų integravimui į arbitražo sistemas.

Pagrindiniai tikslai:

- Įgyti praktinių žinių debesijos (ang. cloud) technologijų taikyme finansinių sistemų kūrime
- Išmokti integruoti skirtingus kriptovaliutų rinkos mechanizmus
- Išmokti efektyvių tarp-serverių komunikacijos optimizavimo metodų

Konkretūs praktikos uždaviniai:

1. Išstudijuoti Go ir Rust kalbų bibliotekas, reikalingas kriptovaliutų prekybai
2. Įvaldyti įmonėje naudojamą kriptovaliutų prekybos ir stebėjimo programinę įrangą
3. Kūrybiškai taikyti žinias kuriant ir tobulinant prekybos sistemas
4. Atlikti išsamų sukurtų sistemų testavimą realiomis rinkos sąlygomis

Praktikos vykdymo planas

Praktikos periodas (10 savaičių) suskirstytas į logiškus etapus:

- **1 savaitė**: Adaptacinis periodas - susipažinimas su įmonės infrastruktūra, kodo baze ir darbo metodologijomis
- **2 savaitė**: Tyrimų etapas - analizuojamas skolinimosi ir likvidacijų mechanizmų veikimas kriptovaliutų rinkoje
- **3-5 savaitės**: Duomenų rinkimas ir analizė - sistemingas skolininkų stebėjimas, likvidacijos momentų identifikavimas

- **6-7 savaitės**: Integracijos darbai - likvidacijų mechanizmų įterpimas į esamą arbitražo paieškos sistemą
- **8-10 savaitės**: Testavimas ir optimizavimas - sukurtų sprendimų patikrinimas realiomis sąlygomis ir nuolatinis tobulinimas

1. Įmonė

1.1. Bendras apžvalga

Humbility UAB yra kriptovaliutų prekybos įmonė, specializuojanti arbitražo operacijose tarp įvairių biržų. Įmonė veikia tiek centralizuotose (CEX), tiek decentralizuotose (DEX) kriptovaliutų biržose, taikydama tris pagrindines strategijas:

- DEX orientuotą strategiją
- CEX orientuotą strategiją
- Hibridinę strategiją, jungiančią abi platformas

1.2. Organizacinė struktūra

Įmonėje vyrauja lanksti horizontalia struktūra su aiškiai apibrėžtomis funkcinėmis rolėmis:

- **Produkto savininkas**:
  - Formuluoja produkto vystymo strategiją ir ilgalaikę viziją
  - Prioritetizuoja funkcijų kūrimo eigą pagal verslo poreikius
  - Užtikrina produkto stabilumą ir tęstinį tobulinimą
  - Vykdo reguliarų darbuotojų vertinimą:
    * Individualūs susitikimai (1-on-1) kas 4-6 savaites
    * Konstruktyvus atsiliepimas apie darbą
    * Darbuotojų motivavimas ir karjeros planavimas
  - Koordinuoja komunikaciją tarp verslo ir techninių komandų
- **Techninis lyderis**:
  - Apibrėžia techninius standartus ir gaires
  - Užtikrina sistemų architektūros vientisumą
  - Koordinuoja techninius sprendimus tarp komandų
- **Programuotojas**:
  - Kūria ir įgyvendina algoritminės prekybos sprendimus
  - Diagnozuoja ir taiso sistemos problemas, įskaitant:
    * Loginės sistemos klaidas
    * Tarp-serverių komunikacijos trikdžius
  - Integruoja naujas kriptovaliutų biržas į prekybos sistemą

- **Analitikų vadovas**:
  - Koordinuoja analitinių įrankių kūrimą
  - Valdo kapitalo paskirstymo ir balansų kontrolės procesus
  - Vadovauja analitikų komandai
- **Duomenų analitikas**:
  - Atlieka rinkos analizę ir konkurencijos monitoringą
  - Identifikuoja naujus pelno generavimo galimybes
  - Sukuria automatinius sistemos veiklos monitoringo įrankius

Praktikoje dažnai pasitaiko situacijų, kai vienas darbuotojas atlieka kelių rolėms būdingas funkcijas, kas leidžia optimizuoti žmogiškuosius išteklius ir palaikyti organizacijos lankstumą.

1.3. Veiklos ypatumai

Humbility yra savarankiška įmonė be išorinių investuotojų, kurios visi valdybos nariai yra aktyvūs projekto dalyviai. Ši struktūra leidžia:

- Greitai priimti sprendimus be biurokratinių kliūčių
- Lanksčiai reaguoti į rinkos pokyčius
- Nuolat inovuoti ir tobulinti prekybos sistemas

1.4. Darbo sąlygos

Įmonė siūlo unikalias darbo sąlygas, pritaikytas kriptovaliutų rinkos specifikai:

- **Elastingas darbo grafikas** - darbas nevaržomas tradicinių darbo valandų
- **Studentų draugiška aplinka** - studentams suteikiama galimybė derinti darbą su studijomis
- **Moderni darbo erdvė** - Vilniaus biure įrengta atvira darbo zona, skatinanti tiesioginį bendravimą
- **Ribota nuotolinio darbo galimybė** - tam tikrais atvejais, pagal poreikį ir susitarimą, leidžiama dirbti nuotoliniu būdu

2. Darbas praktikos metu

2.1. Vystomas produktas

Įmonėje dirbau komandoje, kurios pagrindinis tikslas – išnaudoti decentralizuotas kriptovaliutų rinkas siekiant pelno. Skirtingai nuo įprastų arbitražo strategijų, mūsų komanda daugiausiai dėmesio skiria atominiam arbitražui, vykdomam pačiose blokų grandinės (blockchain) sistemose, be jokios centrinės keityklos įsikišimo. Tai leidžia pasinaudoti decentralizuotų finansų (DeFi) rinkų neefektyvumais ir saugiai įvykdyti arbitražo operacijas vienos transakcijos metu.

2.1.1. Arbitražas

Arbitražas – tai finansinė strategija, kai pasinaudojama kainų skirtumais tarp skirtingų rinkų siekiant gauti pelną be (ar beveik be) rizikos. Kriptovaliutų kontekste tai dažniausiai reiškia valiutų konvertavimą skirtinguose decentralizuotuose keityklose.

2.1.2. Atomiškumas

Atomiškumas (angl. atomicity) reiškia, kad operacija vyksta „viskas arba nieko“ principu. Blockchain sistemose tai itin svarbu, nes leidžia užtikrinti, kad arbitražo grandinė bus sėkminga tik tada, kai įvykdomi visi sandoriai – kitaip ji atšaukiama. Tokiu būdu išvengiama situacijų, kai tik dalis konversijų įvykdoma, ir likusi dalis sukelia nuostolius. Šis principas yra kertinis atominio arbitražo dalis, nes garantuoja, kad kiekviena grandinė yra pelninga ir rizika minimali.

2.2. Komandos procesai

Komanda dirba be griežtų „sprintų“ ar iteracijų – kiekvienas komandos narys pats planuoja savo darbus ir vykdo užduotis tol, kol jos yra užbaigiamos. Baigus užduotį, kreipiamasi į produktų vadovą (product owner), kuris paskiria sekančią užduotį ar nukreipia tolimesniam darbui.

Darbo eiga prižiūrima per „Slack“ susirašinėjimo platformą – kiekvieną rytą komandos nariai pateikia trumpą ataskaitą: kas buvo padaryta vakar ir ką planuoja šiandien. Tai padeda palaikyti komandinį skaidrumą ir sekti progresą realiu laiku.

Kartą per mėnesį organizuojamas bendras komandos susitikimas, kurio metu aptariami praėjusio mėnesio rezultatai, apžvelgiamos reikšmingesnės techninės ar verslo įžvalgos, bei nusibrėžiamos kryptys didesnio masto (angl. scope) darbams artimiausiam laikotarpiui. Toks lankstus procesas leidžia greitai prisitaikyti prie besikeičiančių prioritetų ir veiksmingai spręsti aktualiausias problemas.

2.3. Likvidacijų integracija į sistemą

Vienas iš praktikos tikslų buvo sujungti likvidacijos mechanizmą su esama arbitražo sistema. Tai leido sukurti papildomą pelno šaltinį: kai arbitražo sistema randa pelningą valiutų keitimą, prie

jos prijungus likvidacijos galimybę, galima ne tik uždirbti iš kainų skirtumų, bet ir papildomai gauti likvidatoriaus atlygį.

Į likvidacijų procesą buvo integruoti algoritmai, leidžiantys automatizuotai patikrinti ar paskola tapo likviduojama, nustatyti optimalų grąžinamos paskolos dydį ir užstato valiutą, bei suplanuoti pelningiausią valiutų konversijų seką. Įgyvendintas sprendimas analizuoja realaus laiko duomenis iš blokų grandinės ir orakulo, bei vykdo transakciją tik jei įvykdoma visa grandinė atominėmis sąlygomis.

2.3.1. Likvidacijos

Praktikos metu buvo analizuotas *Venus* paskolų protokolo likvidavimo mechanizmas, veikiantis *Binance Smart Chain (BSC)* tinkle. Šios analizės pagrindu buvo sukurtas algoritmas, kuris leidžia automatiškai nustatyti, kada paskolos pozicija tampa likviduojama, ir įvykdyti likvidaciją siekiant maksimalaus pelno.

Sukurtas sprendimas vertina tiek galimą finansinę grąžą, tiek ir transakcijų sąnaudas (angl. *gas*), kad būtų užtikrintas optimalus efektyvumo ir pelningumo santykis. Sistema buvo integruota į esamą arbitražo įrankį, išplečiant jo galimybes ir suteikiant papildomą pajamų šaltinį.

Gilinantis į šią temą teko susipažinti su paskolų platformoms būdingais terminais ir principais:

- **Pozicija (angl. *position*)** – naudotojo turimų užstatų ir paskolų rinkinys, išskaidytas pagal valiutas. Kiekviena tokia pozicija turi savo rizikos ir likvidumo lygį.
- **Likvidacijos paskata** – procentinis priedas, kurį gauna likvidatorius už atliktą likvidaciją. *Venus* protokole ši paskata siekia 10% nuo grąžinamos skolos sumos, taip skatinant kuo greitesnį reagavimą į rizikingas pozicijas.
- **Likvidavimo slenkstis (angl. *liquidation threshold, LT*)** – procentas, kuriuo užstato vertė įskaitoma į bendrą skolinimosi pajėgumą. Kiekviena valiuta turi atskirą LT reikšmę (paprastai nuo 60% iki 90%).
- **Uždarymo riba (angl. *close factor, CF*)** – maksimali skolos dalis, kuri leidžiama būti grąžinta vienos likvidacijos metu. *Venus* protokole ši riba yra 50%.
- **Skolinimosi pajėgumas (angl. *borrowing capacity, BC*)** – tai bendra skolos vertė, kurią skolininkas gali turėti, atsižvelgiant į užstato sumą ir LT:

$$\text{BC} = \sum_i (\text{Užstato vertė}_i \times LT_i)$$

- **Sveikumo koeficientas (angl. *health factor, HF*)** – tai santykis tarp skolinimosi pajėgumo ir faktinės skolos:

$$\text{HF} = \frac{\text{BC}}{\sum_i \text{Skolos vertė}_i}$$

  Kai $HF < 1$, skolininko pozicija laikoma pažeidžiančia saugumo reikalavimus ir tampa likviduojama.

Likviduojant poziciją, būtina pasirinkti konkrečią paskolos ir užstato valiutų porą, taip pat – grąžintinos sumos dydį. Jeigu po pirmos likvidacijos pozicija vis dar išlieka nesaugi ($HF < 1$), likvidacija gali būti kartojama kelis kartus. Dėl šių niuansų reikėjo ne tik sukurti logiką sprendimo paieškai, bet ir atsižvelgti į visus ribojančius faktorius, kad sprendimas būtų finansiškai pagrįstas ir techniškai korektiškas.

2.3.2. Našumo problema ir sprendimas

Integruojant likvidacijos algoritmą į arbitražo sistemą, buvo susidurta su rimta našumo problema. Kadangi aktyvių skolininkų kiekis siekia dešimtis tūkstančių, kiekvienas valiutos kainos pokytis reikalauja peržiūrėti visų skolininkų pozicijas ir perskaičiuoti jų sveikumo koeficientą (HF), siekiant nustatyti ar pozicija tampa likviduojama. Šis procesas tapo akivaizdžiai per lėtas realaus laiko sistemai, kur kainos nuolat kinta.

Siekiant šią problemą išspręsti, buvo sukurta specializuota duomenų struktūra – `zeroCenteredSet`. Šios struktūros principas remiasi tuo, kad kiekvienas skolininkas, priklausomai nuo turimų valiutų ir jų užstato santykio, yra priskiriamas prie tam tikros vietos skirtingose valiutų skalėse. Kiekviena tokia skalė yra centrinė pagal pradinę valiutos kainą ir išsiplečia tiek į teigiamą, tiek į neigiamą pusę – t. y. skalė yra simetriška aplink nulį.

Kiekvieno skolininko pozicijos vieta šioje skalėje nusako, kokio dydžio kainos pokytis konkrečiai valiutai reikalingas, kad ta pozicija taptų likviduojama. Kai valiutos kaina keičiasi, apskaičiuojamas kainos pokytis nuo pradinės (ref) reikšmės, ir struktūra leidžia greitai išrinkti tik tuos skolininkus, kuriems šis pokytis reikšmingas. Taip eliminuojama būtinybė tikrinti visų skolininkų pozicijas, kas leidžia drastiškai sumažinti perskaičiavimų kiekį.

`zeroCenteredSet` struktūra tapo esmine sistemos dalimi, leidžiančia užtikrinti greitą reagavimą į rinkos pokyčius ir išlaikyti sistemą efektyvią net ir esant itin didelei aktyvių paskolų bazei.

2.3.3. Darbas su Rust programavimo kalba blockchain aplinkoje

Praktikos metu taip pat teko dirbti su Rust programavimo kalba, ypač kontekste, kai kodas buvo leidžiamas tiesiogiai „on-chain“, t. y. vykdomas blockchain'e kaip išmaniosios sutartys arba jų komponentai. Tokie sprendimai reikalauja ypatingo dėmesio efektyvumui, saugumui ir deterministiniam veikimui – savybėms, kurios būdingos Rust kalbai.

Kadangi Rust iki tol buvau naudojęs tik lokalioms sistemoms ar CLI įrankiams, reikėjo greitai perprasti blockchain'ui pritaikytą programavimo paradigmą, ypač su tokiais framework'ais kaip `Ink!` (Polkadot ekosistemoje) ar `Solang` (Solidity–Rust transpiliatorius). Taip pat teko susipažinti su mažo footprint'o reikalavimais – visi on-chain veikiantys moduliai turi būti maksimaliai kompaktiški ir optimizuoti, nes kiekvienas papildomas baitas padidina transakcijų sąnaudas.

Vienas iš svarbių uždavinių buvo Rust kalba papildyti modulį, kuris tikrina ar vartotojo pozicija atitinka saugumo reikalavimus, ir jei ne – inicijuoja likvidacijos logiką.

Teko išmokti Rust ypatybių, kurios ypač svarbios dirbant su blockchain:

- **Ownership ir borrow checker** – garantuoja atminties saugumą be garbage collection, kas labai svarbu deterministiniam vykdymui.
- **No-std aplinka** – on-chain kodas dažnai vykdomas be standartinės bibliotekos (std), todėl reikia naudoti specializuotus `core` ir `alloc` modulius.
- **Serde serializacija** – duomenų struktūrų konvertavimas į byte formą bei atgal, būtinas siekiant suderinamumo su blockchain protokolais.
- **Panic-free kodas** – kiekviena panika (angl. *panic*) on-chain aplinkoje reiškia transakcijos atmetimą, todėl būtina naudoti saugius ir atidžiai tikrinamus skaičiavimus.

Šis darbas ne tik sustiprino mano Rust kalbos žinias, bet ir atvėrė naują požiūrį į tai, kaip programuojamos decentralizuotos sistemos. Įgyta patirtis bus neabejotinai naudinga tolesniam darbui blockchain srityje, kur Rust tampa vis svarbesniu pasirinkimu dėl savo našumo ir saugumo.

2.4. Darbas su debesija

Praktikos metu taip pat teko prisidėti prie debesijos infrastruktūros palaikymo ir plėtros. Komanda palaiko keliolika serverių, išdėstytų įvairiose geografinėse lokacijose – tai leidžia optimizuoti arbitražo sistemų veikimą, atsižvelgiant į konkrečių biržų ar tinklų geografinį išsidėstymą ir atsako laikus.

Visas mūsų rašomas kodas yra diegiamas į serverius naudojant automatizuotus diegimo procesus – tam pasitelkiami GitHub Actions (CI/CD workflows) ir konfigūracijos valdymo įrankis Ansible. Tokia automatizacija leidžia greitai ir saugiai atlikti pakeitimus visoje infrastruktūroje bei užtikrinti, kad skirtingose lokacijose veikiantys serveriai naudoja tą pačią, patikrintą versiją.

Keli serveriai yra dedikuoti tik testavimui. Visi pakeitimai prieš diegiant į produkciją pirmiausia įkeliami ir testuojami šiose aplinkose. Tai leidžia užtikrinti, kad nauji moduliai, algoritmai ar optimizacijos nesukels netikėtų klaidų ar trikdžių realiuose prekybos procesuose. Tik įsitikinus jų stabilumu ir veikimu, kodas perkeliamas į gamybinę (angl. production) aplinką.

Toks infrastruktūros valdymas ir diegimo procesas yra būtinas norint užtikrinti stabilų ir nenutrūkstamą sistemų veikimą itin jautriame – finansinių operacijų – kontekste.

2.5. Decentralizuotų finansų (DeFi) rinkos dinamika

Dirbant su likvidacijomis ir arbitražo algoritmais neišvengiamai tenka gilintis ne tik į techninę infrastruktūrą, bet ir į pačios DeFi rinkos dinamiką. Ši rinka yra ypatinga tuo, kad viskas vyksta realiuoju laiku, be centrinio valdymo, o naudotojų elgsena ir turto vertės pokyčiai dažnai yra nulemti tiek makroekonominių veiksnių, tiek „on-chain“ paskatų.

Dėl šios priežasties svarbu ne tik gebėti vykdyti arbitražo operacijas ar aptikti rizikingas paskolų pozicijas, bet ir suprasti, kaip kinta likvidumo baseinai, kaip veikia AMM (automatizuoti rinkos formuotojai), kokios rizikos kyla dėl „flash loan“ atakų, ar kaip susidaro „front-running“ situacijos.

Be to, kiekvienas išmanusis kontraktas (smart contract), kurį bandoma iškviesti ar su juo sąveikauti, turi savo būseną, kas reikalauja nuolat palaikyti duomenų aktualumą. Todėl buvo būtina įdiegti realaus laiko stebėseną (angl. live monitoring), kuri leidžia tiksliai fiksuoti rinkos būsenos pokyčius blokas po bloko.

3. Rezultatai, išvados ir pasiūlymai

3.1. Įgytos žinios

Praktikos metu įgytos gilios žinios apie decentralizuotų paskolų protokolus, jų veikimo principus ir likvidacijų mechaniką. Buvo praktiškai suprasti tokie svarbūs DeFi pasaulio terminai kaip pozicija, skolinimosi pajėgumas, sveikumo koeficientas, likvidavimo slenkstis, uždarymo riba bei likvidacijos paskata.

Be finansinių sistemų veikimo principų, taip pat įgyta techninių žinių apie programinę inžineriją debesijoje – nuo infrastruktūros valdymo su `Ansible` ir `GitHub actions`, iki automatizuoto kodo diegimo keliuose geografiniuose serveriuose. Įgyta patirties su test–prod diegimo praktika, užtikrinančia saugią kodų integraciją.

Technologiniu požiūriu pagilintos Go ir Rust programavimo kalbų žinios, įsisavinti įrankiai darbui su blokų grandinės duomenimis (Web3, orakulai, RPC sąsajos). Taip pat sustiprintas supratimas apie atominį arbitražą ir jo įgyvendinimą realiose rinkose su aukštu patikimumu bei rizikos kontrole.

3.2. Universitete įgytų žinių vertinimas

Didžioji dalis darbo praktikos metu reikalavo sisteminio mąstymo, inžinerinės logikos ir gero algoritmų išmanymo – sritys, kurios buvo stipriai lavinamos universiteto studijų metu. Tokie dalykai kaip duomenų struktūros, operacijų sistemų pagrindai, testavimas ir sistemų architektūra pasitarnavo tiek suprantant egzistuojančią infrastruktūrą, tiek ją pritaikant ar tobulinant.

Vis dėlto, debesijos technologijų ir decentralizuotų finansų (DeFi) taikymas buvo ribotai (arba visai ne) nagrinėjami studijų programoje. Praktikoje teko pirmą kartą gilintis į „production-grade“ infrastruktūrą, kur automatizacija, CI/CD procesai ir distribucinis sistemų valdymas yra būtini norint užtikrinti nepertraukiamą paslaugų veikimą.

3.3. Darbo komandoje privalumai

- Laisva, bet efektyvi darbo struktūra be sprintų, leidžianti susikoncentruoti į užduotis.
- Komandos nariai puikiai įvaldę technines sritis, pasiruošę dalintis žiniomis.
- Galimybė iš karto pamatyti, kaip tavo parašytas kodas veikia realiose situacijose – nuo testavimo iki deploy į produkciją.
- Nuolatinis ryšys su produkto savininku, kuris greitai suteikia grįžtamąjį ryšį ir padeda fokusuotis į svarbiausias funkcijas.
- Kiekvieno komandos nario savarankiškumas ir pasitikėjimas vieni kitais.

3.4. Darbo komandoje trūkumai

- Kadangi komanda dirba be sprintų, kartais gali kilti neapibrėžtumo pojūtis – ypač naujiems komandos nariams.
- Nėra automatizuotos užduočių valdymo sistemos (pvz. JIRA, Linear), todėl užduotys valdomos neformaliai per Slack – tai reikalauja geros savidisciplinos.
- Testavimo serverių resursai riboti – kartais tai apsunkina paruoštų funkcijų tikrinimą paraleliai vykdomiems kitiems testams.

3.5. Pasiūlymai

3.5.1. Universitetui

- Įtraukti daugiau modulių, orientuotų į modernias infrastruktūros ir debesijos technologijas, ypač praktiniu lygmeniu.
- Suteikti studentams galimybę rinktis pasirenkamuosius dalykus iš kitų susijusių programų (pvz. DeFi, DevOps, distrib. sistemos).
- Skatinti projektus, kurie simuliuotų realias „legacy“ kodo bazes – studentai galėtų mokytis dirbti ne tik nuo nulio, bet ir tobulinti esamas sistemas.
- Įtraukti decentralizuotų sistemų (blockchain) pagrindus kaip pasirenkamą modulį.

3.5.2. Įmonei

- Sukurti paprastą vidinę žinių bazę, kur būtų aprašyti dažniausiai pasitaikantys procesai (deployment, testing, debug scenarijai).
- Formalizuoti vidinį „mentoring“ procesą naujokams, kad būtų greičiau įsisavinama sistema.
- Praplėsti testinių serverių galimybes arba įgalinti lokalų testavimą „staging“ režimu su realaus duomenų mockais.

3.6. Darbo rezultatai ir išvados

Praktikos metu buvo pasiekti šie pagrindiniai rezultatai:

1. Išanalizuotas *Venus* paskolų protokolo veikimas ir sukurti algoritmai, leidžiantys automatiškai nustatyti likviduojamas pozicijas bei vykdyti pelningą likvidaciją.
2. Integruotas likvidacijos algoritmas į esamą arbitražo sistemą, išplečiant jos funkcionalumą.
3. Suprogramuota atskiro komponento testavimo infrastruktūra su palaikymu test serveriuose.

4. Sukurti CI/CD workflow’ai, leidžiantys automatizuotai deploy’inti kodą į realias geografiškai paskirstytas lokacijas.

Apskritai, praktika buvo ne tik vertinga profesiškai, bet ir leido įgyti praktinių žinių, kurių nebuvo galimybės gauti studijų metu. Dirbant realioje aukšto intensyvumo aplinkoje teko priimti sprendimus, daryti kompromisus tarp rizikos, kaštų ir pelno, taip pat suvokti kodėl architektūriniai sprendimai daromi vienu ar kitu būdu. Tai buvo puiki patirtis ruošiantis tolesnei karjerai technologijų sektoriuje.