МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

Булганы Раднаабазар

МЭДЭЭЛЛИЙН ТЕХНОЛОГИЙН ХӨТӨЛБӨРИЙН ҮЙЛДВЭРИЙН ДАДЛАГЫН ТАЙЛАН

(Internship Report)

Мэдээллийн технологи (D061304)
Дадлагын ажлын тайлан

Улаанбаатар

2025 оны 09 сар

МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

МЭДЭЭЛЛИЙН ТЕХНОЛОГИЙН ХӨТӨЛБӨРИЙН ҮЙЛДВЭРИЙН ДАДЛАГЫН ТАЙЛАН

(Internship Report)

Мэдээллийн технологи (D061304)
Дадлагын ажлын тайлан

Удирдагч: _____ Директори. С.Дамдинсүрэн

Хамтран удирдагч: _____

Гүйцэтгэсэн: _____ Б.Раднаабазар (22B1NUM0286)

Улаанбаатар

2025 оны 09 сар

Зохиогчийн баталгаа

Миний бие Булганы Раднаабазар ”МЭДЭЭЛЛИЙН ТЕХНОЛОГИЙН ХӨТӨЛБӨРИЙН ҮЙЛДВЭРИЙН ДАДЛАГЫН ТАЙЛАН” сэдэвтэй судалгааны ажлыг гүйцэтгэсэн болохыг зарлаж дараах зүйлсийг баталж байна:

- Ажил нь бүхэлдээ эсвэл ихэнхдээ Монгол Улсын Их Сургуулийн зэрэг горилохоор дэвшүүлсэн болно.
- Энэ ажлын аль нэг хэсгийг эсвэл бүхлээр нь ямар нэг их, дээд сургуулийн зэрэг горилохоор оруулж байгаагүй.
- Бусдын хийсэн ажлаас хуулбарлаагүй, ашигласан бол ишлэл, зүүлт хийсэн.
- Ажлыг би өөрөө (хамтарч) хийсэн ба миний хийсэн ажил, үзүүлсэн дэмжлэгийг дипломын ажилд тодорхой тусгасан.
- Ажилд тусалсан бүх эх сурвалжид талархаж байна.

Гарын үсэг: _____

Огноо: _____

ГАРЧИГ

ЗУРГИЙН ЖАГСААЛТ

ХҮСНЭГТИЙН ЖАГСААЛТ

Кодын жагсаалт

УДИРТГАЛ

Энэхүү үйлдвэрийн дадлагын тайлан нь БиДиСЕК ҮЦК компанид Мерчанитийн супер аппликейшнд Mini-App хөгжүүлсэн туршлагыг тайлбарлана. Гол зорилго нь мерчант дээр суурилсан мини аппликейшнээр дамжуулан хэрэглэгчийн бүртгэл, шимтгэл төлөлт, бодит цагийн үнэт цаасны ханш харах, хоёрдогч зах зээлийн арилжаанд оролцох боломжийг бүрдүүлэх явдал байв.

Техникийн хувьд Next.js, React, Node.js, Express, TypeScript, MySQL ашиглаж, SOLID зарчимд суурилсан архитектур бүтээв. Merchant Super App OAuth/refresh token механизм, ҮЦТХТ SOAP API-г амжилттай интеграцчилсан. 300,000 мөр өгөгдөлд binary search ашиглан хурдан хайлт (5–40ms), WebSocket болон cron job ашиглан real-time дата дамжуулах систем хөгжүүлэв.

Бэкенд архитектурыг SOLID зарчмын дагуу зохион байгуулж, Merchant Super App OAuth, ҮЦТХТ retry логик, securities синхронизаци, socket сервер, алдаа мэдэгдэгч сервис зэрэг гол бүрэлдэхүүнүүдийг хэрэгжүүлсэн. Docker ашиглан CI/CD pipeline-тэй VPS дээр суурилуулсан.

1. КОМПАНИЙН ТАНИЛЦУУЛГА БА МЭДЭЭЛЛИЙН СИСТЕМҮҮД

1.1 ”БиДиСЕК ҮЦК” ХК танилцуулга

”БиДиСек ҮЦК” ХК нь 1991 онд байгуулагдсан бөгөөд Монголын хөрөнгийн зах зээлд хамгийн олон харилцагчтай, тэргүүлэгч брокер, дилер, хөрөнгө оруулалтын зөвлөх, андеррайтерын компаниудын нэг юм. Үнэт цаасны зах зээлийн арилжаанд зуучлах, шинээр үнэт цаас гаргахад болон хөрөнгө оруулалтын зөвлөгөө өгөх зэрэг цогц үйлчилгээ үзүүлдэг мэргэжлийн компаниудын нэг бөгөөд салбартаа тасралтгүй 30 гаруй жилийн үйл ажиллагаагаа явуулж байгаа туршлагатай.

Компанийн үндсэн үйл ажиллагаа нь дараах таван чиглэлээр хэрэгждэг.

1. **Брокер, дилерийн үйлчилгээ**

   Монголын хөрөнгийн зах зээлд хувь хүн болон байгууллагуудад хөрөнгө оруулах боломжийг олгож, мэргэжлийн туршлагатай хамт олноор дамжуулан цогц үйлчилгээ үзүүлдэг.

2. **Андеррайтерийн үйлчилгээ**

   Үнэт цаас гаргалтыг зохион байгуулах, эрсдэлийн удирдлагыг хэрэгжүүлэх замаар үйлчлүүлэгчдэд найдвартай, мэргэжлийн шийдэл санал болгодог.

3. **Хөрөнгө оруулалтын зөвлөх үйлчилгээ**

   Компанийн засаглал, бүтцийн өөрчлөлт, хувьцаа эзэмшигчдийн хурлын зохион байгуулалт, арилжаа болон санхүүжилт татах зэрэг чиглэлээр зөвлөгөө өгдөг.

4. **Гадаад арилжааны зуучлал**

   Олон улсын хөрөнгийн зах зээлд арилжаа хийх боломжийг олгохоос гадна шаардлагатай зөвлөгөө, зуучлалын үйлчилгээг үзүүлдэг.

5. **Уул уурхайн бүтээгдэхүүний арилжаа**

   Монголын хөрөнгийн бирж дээр арилжаалагддаг уул уурхайн бүтээгдэхүүнийг харилцагчдад зуучлан арилжаа хийж өгдөг.

БиДиСек ҮЦК ХК нь санхүүгийн салбарын тэргүүлэгчийн хувьд үйлчлүүлэгчдийнхээ хэрэгцээнд нийцсэн шинэчлэл, дижитал үйлчилгээ нэвтрүүлэхэд байнга анхаарч ажилладаг.

Миний оролцсон **”Мини апп”** төсөл нь мерчант байгууллага өөрийн үйлчилгээг томоохон олон супер аппликейшн дотор мини-апп хэлбэрээр хэрэгжсэн бөгөөд брокерийн үйлчилгээ, төлбөр тооцоог илүү хүртээмжтэй, дэвшилтэт технологиор хангах зорилготой юм. Энэхүү систем нь тухайн системтэй OAuth механизмаар хэрэглэгчийн нэвтрэлтийг хангаж, фронтенд URL-ээр дамжуулан манай системтэй холбогддог.

1.2 Компанийн мэдээллийн системүүд

БиДиСек ҮЦК ХК нь брокер, дилерийн үйл ажиллагаагаа дэмжихийн тулд олон төрлийн мэдээллийн системийг ашиглаж байна. Үүнд харилцагчийн бүртгэл, арилжааны мэдээлэл, төлбөр тооцоо, үнэт цаасны хадгаламжийн мэдээлэл зэрэг суурь үйл ажиллагааг дэмжсэн дотоод болон гадаад холболттой системүүд ордог.

1.2.1 Одоогийн системийн байдал

Одоогийн байдлаар харилцагчийн бүртгэл, хөрөнгийн зах зээлийн арилжааг агуулсан үндсэн системийг тусдаа гадны системээр явуулдаг. Уг системд шинэчлэлт хийх, алдааг засахад хугацаа их зарцуулагддаг бөгөөд заримдаа дутуу засагдаж, хэрэглэгчийн туршлага муу байдаг. БиДиСЕК ҮЦК нь өөрийн нийт харилцагч зориулсан систем нарт дотооддоо хөгжүүлдэг боловч жинхэнэ брокер, дилерийн үйл ажиллагааг дэмжих цогц систем байгуулахад хангалтгүй байна. Мөн шинэ системийн модуль хөгжүүлэхэд гадны хөгжүүлэгчдээс хамааралтай байх нь бизнесийн үйл ажиллагааг удаашруулдаг.

1.3 Шинэчлэлийн хэрэгцээ

1.3.1 Шаардлагатай өөрчлөлт

Эдгээр нөхцөл байдлаас шалтгаалан дараах өөрчлөлт шаардлагатай болсон:

1. Компанийн ашиглаж байгаа гадны системийн үйлчилгээг бага багаар өөрсдөө хөгжүүлж, нэвтрүүлэх
2. Одоо байгаа системүүдийн өгөгдлийг илүү үр ашигтайгаар томоохон мерчантийн аппликейшнд нэгтгэх
3. Банкны API-тай шууд холбож, төлбөр тооцоог төвлөрсөн байдлаар гүйцэтгэх
4. ҮЦТХТ-тай интеграц хийж, харилцагчийн бүртгэлийн мэдээлэл болон KYC статусыг найдвартайгаар шинэчлэх
5. Үнэт цаас болон компанийн мэдээллийг бодит цагийн горимд хэрэглэгчдэд шууд хүргэж, дансаа цэнэглэж арилжаанд оролцуулах

Ийм нөхцөлд **”Мини апп”** төслийг хэрэгжүүлснээр компанийн үйлчилгээний хүртээмжийг нэмэгдүүлэх, хэрэглэгчийн туршлагыг сайжруулах, өгөгдөл дамжуулалтын хурд болон найдвартай байдлыг нэмэгдүүлэх боломж бүрдсэн.

2. ТӨСЛИЙН ЕРӨНХИЙ ТАНИЛЦУУЛГА БА ШААРДЛАГА

2.1 Төслийн ерөнхий танилцуулга

”Мини апп” төсөл нь БиДиСЕК ҮЦК-ийн харилцагчдад орчин үеийн дижитал үйлчилгээг хүргэх зорилгоор хөгжүүлсэн фүллстек веб аппликейшн юм. Энэхүү төсөл нь компанийн одоогийн гадны системийн хамаарлыг багасгаж, өөрийн хөгжүүлсэн системээр үйлчилгээний чанарыг сайжруулах зорилготой юм.

Төслийн гол зорилго нь дараах үйлдлүүдийг нэгтгэсэн цогц системийг бүрдүүлэх явдал юм:

- Мерчантийн аппликейшнд мини аппликейшн хэлбэрээр суурилуулах
- Харилцагчийн бүртгэл хийх болон KYC статусыг шалгах
- Бүртгэлийн шимтгэлийг банкны API-аар төлөх
- Үнэт цаасны ханшийн өгөгдлийг бодит цагийн горимд харуулах
- Харилцагчийн өөрийн үнэт цаасуудыг харах болон удирдах
- Төлбөр тооцооны үйлдлийг төвлөрсөн байдлаар гүйцэтгэх
- Үнэт цаасны хоёрдогч зах зээлийн арилжаанд оролцох

2.2 Техникийн шаардлага

2.2.1 Фронтенд шаардлага

Фронтенд хэсэгт дараах техникийн шаардлагуудыг тавьсан:

- **Next.js framework** - React-д суурилсан сервер-сайд рендеринг болон статик сайт генераци

- **TypeScript** - Кодын чанарыг сайжруулах, алдааг урьдчилан сэргийлэх
- **Responsive Design** - Мобайл төхөөрөмж дээр ажиллах
- **Modern UI/UX** - Figma-д боловсруулсан дизайн системийг хэрэгжүүлэх

2.2.2 Бэкенд шаардлага

Бэкенд системд дараах техникийн шаардлагуудыг тавьсан:

- **Node.js** - JavaScript runtime environment
- **Express.js** - RESTful API хөгжүүлэх
- **Prisma ORM** - Өгөгдлийн сантай ажиллах
- **TypeScript** - SOLID зарчмыг баримтлах
- **JWT Authentication** - Аюулгүй байдлыг хангах

2.2.3 Интеграцийн шаардлага

Гадаад системүүдтэй холбогдох шаардлага:

- **Банкны API** - Төлбөр тооцооны үйлчилгээ
- **ҮЦТХТ SOAP API** - Харилцагчийн таних үйлчилгээ
- **Кронжоб** - Өгөгдлийн автомат татах систем
- **300,000+ мөртэй өгөгдөл** - Масс өгөгдлийн интеграци

2.3 Системийн архитектур

Төслийн системийн архитектур нь гурван үндсэн давхаргаас бүрдэнэ:

2.3.1 Презентацийн давхарга (Frontend)

Next.js-д суурилсан реактив веб интерфейс бөгөөд дараах компонентуудыг агуулна:

- Харилцагчийн бүртгэлийн форм
- Төлбөр тооцооны интерфейс
- Үнэт цаасны ханшийн харуулах самбар
- Хувийн дансны удирдлага

2.3.2 Бизнес логикийн давхарга (Backend)

Node.js-д суурилсан API сервер бөгөөд дараах үйлчилгээг үзүүлнэ:

- Харилцагчийн бүртгэлийн үйлчилгээ
- Төлбөр тооцооны үйлчилгээ
- Үнэт цаасны мэдээллийн үйлчилгээ
- Аюулгүй байдлын үйлчилгээ

2.3.3 Өгөгдлийн давхарга (Data Layer)

Өгөгдлийн сан болон гадаад системүүдтэй холбогдох давхарга:

- MySQL өгөгдлийн сан
- Гадаад API интеграци
- Кэш систем
- Файл хадгалалт

2.4 Хэрэглэгчийн шаардлага

2.4.1 Функциональ шаардлага

1. **Харилцагчийн бүртгэл** - Шинэ харилцагч бүртгэх, KYC статус шалгах
2. **Төлбөр тооцоо** - Бүртгэлийн шимтгэл төлөх, данс цэнэглэх
3. **Үнэт цаасны мэдээлэл** - Ханш харах, портфолио удирдах
4. **Хувийн мэдээлэл** - Профайл засах
5. **Алдаа мэдэгдэгч сервис** - Системийн түвшинд гарсан алдааг Discord-ээр мэдэгдэх

2.4.2 Функциональ бус шаардлага

1. **Аюулгүй байдал** - HTTPS, JWT токен, шифрлэлт
2. **Гүйцэтгэл** - 3 секундын дотор хуудас ачаалах
3. **Хүртээмж** - 24/7 үйлчилгээ, 99.9% uptime

3. ТЕХНИКИЙН ХЭРЭГЖҮҮЛЭЛТ

3.1 Системийн архитектур

Мини апп төсөл нь дараах архитектураар бүтээгдсэн:

- **Merchant Super App** - Мерчант аппликейшны мобайл аппликейшн дотор мини-апп хэлбэрээр суурилуулагдсан
- **Frontend** - Next.js TypeScript-ээр бичигдсэн веб интерфейс
- **Backend** - Node.js Express фреймворк дээр суурилсан RESTful API сервер
- **Өгөгдлийн сан** - MySQL өгөгдлийн сан Prisma ORM-ээр удирддаг
- **Гадаад интеграци** - Merchant Super App OAuth, ҮЦТХТ SOAP үйлчилгээ

3.2 Фронтенд хөгжүүлэлт

3.2.1 Next.js болон TypeScript

Next.js framework ашиглан React-д суурилсан фронтенд системийг хөгжүүлсэн. Кодын чанарыг сайжруулах, алдааг урьдчилан сэргийлэх зорилгоор TypeScript ашигласан.

3.2.2 Бүртгэлийн форм - Zod + React Hook Form

Хэрэглэгчийн бүртгэлийн формыг React Hook Form болон Zod schema validation ашиглан хэрэгжүүлсэн. Энэ нь хүчтэй типийн аюулгүй байдал болон хэрэглэгчийн таалагдах туршлагыг хангадаг.

3.3 Бэкенд хөгжүүлэлт

3.3.1 SOLID зарчмууд ба хэрэгжүүлэлт

SOLID зарчим нь объект хандалттай программчлалын таван үндсэн зарчим бөгөөд манай төслийн кодод дараах байдлаар хэрэгжсэн:

Single Responsibility Principle (SRP)

Нэг класс зөвхөн нэг шалтгаанаар өөрчлөгдөх ёстой гэсэн зарчим:

Controller-ууд: Зөвхөн HTTP хүсэлт болон хариулт боловсруулна

```
export class UserController {
  //     HTTP layer  -
  merchantRegistration = catchAsync(async (req, res, next) => {
    const result = await userService.merchantRegistration(req.body);
    res.status(200).json({ success: true, data: result });
  });
}
```

Код 3.1: SRP - UserController

Service-үүд: Зөвхөн бизнес логик боловсруулна

```
export class UserService {
  //
  async merchantRegistration(data: any) {
    //      ,
    //
  }
}
```

Код 3.2: SRP - Service Layer

Dependency Injection ба Inversion of Control

Dependency Injection нь класс өөрийн хамаарлыг шууд үүсгэдэггүй, гаднаас өгөгддөг арга юм:

Notification Service-ийн DI Хэрэгжүүлэлт:

```
// Interface  -      abstract
interface NotificationChannel {
  send(message: string): Promise<void>;
}

// Concrete implementation  -
class DiscordChannel implements NotificationChannel {
  async send(message: string) { /* Discord    */ }
}

class EmailNotifier implements NotificationChannel {
  async send(message: string) { /* Email    */ }
}

// Service
export class NotifierService {
  private channels: NotificationChannel[] = [];

  addChannel(channel: NotificationChannel) {
    this.channels.push(channel); // DI pattern
  }
}
```

Код 3.3: Dependency Injection Pattern

Энэ арга нь:

- Тест хийхэд хялбар (mock dependency inject хийх)

- Код өөрчлөхөд уян хатан
- Шинэ notification channel нэмэхэд хялбар

Interface Segregation Principle

Харилцагч зөвхөн өөрт хэрэгтэй interface ашиглах ёстой:

Merchant Super App-тай холбогдох үйлчилгээнүүдийг тусдаа service-ээр салгасан:

- **MerchantService** - Ecommerce API (төлбөр тооцоо)
- **MerchantSuperService** - Super App API (OAuth, хэрэглэгчийн мэдээлэл)

3.3.2 Express.js API

RESTful API-г Express.js framework ашиглан хөгжүүлсэн. Middleware-ээр JWT токен баталгаажуулалт, алдаа боловсруулалт, лог бичлэг зэргийг хэрэгжүүлсэн.

3.3.3 Өгөгдлийн сан удирдлага

Prisma ORM ашиглан MySQL өгөгдлийн сантай холбогдсон. Энэ нь type-safe өгөгдлийн сангийн хандалт, автомат migration, болон хөгжүүлэлтийн үр ашгийг нэмэгдүүлдэг.

3.4 Merchant Super App OAuth интеграци

3.4.1 OAuth-ийн дэлгэрэнгүй хэрэгжүүлэлт

OAuth Authorization Code Flow нь хамгийн аюулгүй authorization grant төрөл бөгөөд веб аппликейшнд зориулагдсан. Манай системд дараах алхмуудаар хэрэгжсэн:

1. Authorization хүсэлт: Хэрэглэгч Merchant Super App супер апп дотор мини-апп руу шилжих үед систем рүү authorization code хүсэлт илгээнэ.

2. Authorization Code авах: Merchant Super App хэрэглэгчийг баталгаажуулсны дараа authorization code өгч, мини-апп руу redirect хийнэ.

3. **Access Token солилцоо:** Authorization code-г ашиглан access token болон refresh token авах:

```
async getToken(code: string, userId: number | null = null) {
  const url = this.BASE_URL + "/v3/superapp/oauth/token";
  const body = qs.stringify({
    code: code,
    redirect_uri: this.URL,
    client_id: this.CLIENT_ID,
    client_secret: this.CLIENT_SECRET,
  });

  const response = await axios.post(url, body, {
    headers: { "Content-Type": "application/x-www-form-urlencoded" },
    params: { grant_type: "authorization_code" }
  });

  // Refresh token -
  if (userId) {
    await db.user.update({
      where: { id: userId },
      data: { refreshToken: response.data.refresh_token }
    });
  }

  return response.data;
}
```

Код 3.4: Merchant Super App OAuth Token Exchange

3.4.2 *Refresh Token механизм*

Access token дуусах үед refresh token ашиглан шинэ token авах систем:

```
async refreshToken(refresh_token: string, userId: number) {
  const url = this.BASE_URL + "/v3/superapp/oauth/refreshToken";
  const body = new URLSearchParams({ refresh_token });

  const response = await axios.post(url, body.toString(), {
    auth: {
      username: this.CLIENT_ID,
      password: this.CLIENT_SECRET
    },
    headers: { "Content-Type": "application/x-www-form-urlencoded" }
  });

  //    refresh token -
  await db.user.update({
    where: { id: userId },
    data: { refreshToken: response.data.refresh_token }
  });

  return response.data;
}
```

Код 3.5: Refresh Token Rotation

3.5 Үнэт цаасны үйлчилгээ ба синхронизаци

3.5.1 *Үнэт цаасны өгөгдөл татах*

Үнэт цаасны ханш, компанийн мэдээлэл зэрэг өгөгдлийг гадаад API-аас тогтмол татаж авдаг:

- **Cron Job** - Тогтмол хугацаанд үнэт цаасны ханшийг шинэчлэх. Хугацааны давтамжаар Fine-tuning хийв.
- **Delta дата** - Зөвхөн өөрчлөгдсөн өгөгдлийг илгээх
- **Socket Server** - Бодит цагийн горимд клиент руу өгөгдөл дамжуулах

3.5.2 Socket Server бодит цагийн өгөгдөл

WebSocket ашиглан үнэт цаасны ханшийн өөрчлөлтийг бодит цагт илгээх:

```
// Socket Server -
io.on('connection', (socket) => {
  socket.on('subscribe_securities', (symbols) => {
    //
    socket.join(`securities_${symbols.join('_')}`);
  });

  socket.on('unsubscribe_securities', (symbols) => {
    socket.leave(`securities_${symbols.join('_')}`);
  });
});

// Cron job  -
async function broadcastSecuritiesUpdate(updatedData) {
  // Delta           (           )
  const deltaData = calculateDelta(previousData, updatedData);

  //
  deltaData.forEach(security => {
    io.to(`securities_${security.symbol}`)
      .emit('security_update', {
        symbol: security.symbol,
```

```
      price: security.currentPrice,
      change: security.priceChange,
      timestamp: new Date()
    });
  });
}
```

Код 3.6: Real-time Securities Data via WebSocket

3.6 ҮЦТХТ интеграци

3.6.1 300,000+ мөртэй өгөгдөл синхронизаци

Гадны системээс 300,000 гаруй мөртэй өгөгдлийг файл хэлбэрээр татаж авч, хурдан хайлт хийх боломжтой индекс файл үүсгэсэн.

3.6.2 Тэмдэгт мөрөөр хайлт

Binary search ашиглан O(log n) хурдтайгаар тэмдэгт мөрөөр хайлт хийх алгоритм:

```
export async function findAccountByRegistryNumber(registryNumber:
  ↪ string) {
  //
  const index = await loadIndex();
  const indexEntry = binarySearchIndex(index, registryNumber);

  if (!indexEntry) {
    return { success: false, message: "Account␣not␣found" };
  }

  //
  const fileStream = fs.createReadStream(accountsFilePath, {
    start: indexEntry.byteOffset,
```

```
    highWaterMark: 4096
  });

  let data = "";
  for await (const chunk of fileStream) {
    data += chunk;
    const newlineIndex = data.indexOf("\n");
    if (newlineIndex !== -1) {
      const account = JSON.parse(data.slice(0, newlineIndex));
      fileStream.destroy();
      return { success: true, data: account };
    }
  }
}
```

Код 3.7: Optimized Registry Number Search

3.6.3 УЦТХТ SetAccounts давталт механизм

УЦТХТ системд хэрэглэгч бүртгэх үед давхцал гарсан тохиолдолд автомат дахин оролдлого хийх механизм:

```
async function    attemptSetAccounts(userId: number, data: any,
  ↪ maxRetries = 10) {
  let attempts = 0;
  let currentData = { ...data };

  while (attempts < maxRetries) {
    const result = await    SetAccounts(userId, currentData);

    //
    if (result.ResponseCode === 1) {
```

```
      return result;
    }

    //
    if (result.duplicated === "accountId") {
      currentData.accountId = RandomAccountID();
      attempts++;
      continue;
    } else if (result.duplicated === "accountNumber") {
      currentData.accountNumber = RandomAccountNumber();
      attempts++;
      continue;
    }

    throw new AppError(result.ResponseMessage || "  ␣Service␣error",
      ↪ 400);
  }
}
```

Код 3.8: YЦTXT Retry Logic with Duplicate Handling

4. ХӨГЖҮҮЛЭЛТИЙН ЯВЦ

4.1 Багийн ажиллагаа ба хариуцлага хуваарилалт

Төслийг багаар хэрэгжүүлсэн бөгөөд миний үндсэн хариуцлага нь:

- **Бэкенд хөгжүүлэлт** - Node.js Express API, Merchant Super App болон ҮЦТХТ интеграци
- **Фронтенд бүртгэлийн форм** - Zod + React Hook Form ашиглан хэрэглэгчийн бүртгэлийн хэсэг, дизайн, алдаа засах
- **VPS системийн администраци** - Nginx, PM2, MySQL, Docker суурилуулалт болон тохиргоо
- **CI/CD pipeline** - GitHub Actions ашиглан автомат deploy систем

4.2 Төслийн төлөвлөлт

Төслийн төлөвлөлт болон даалгавар хуваарилалтыг Excel файлаар хэрэгжүүлсэн:

- **Системийн төслийн төлөвлөгөө** - Төслийн ерөнхий төлөвлөгөө, milestone-ууд, хугацаа
- **Системийн шаардлага** - Функциональ болон функциональ бус шаардлагууд
- **Даалгаврын хуваарилалт** - Баг гишүүн тус бүрийн хариуцах даалгаврууд

4.3 Хөгжүүлэлтийн аргачлал

4.3.1 Git workflow

GitHub дээр feature branch workflow ашиглан хөгжүүлэлт хийсэн. Код өөрчлөлт бүрийг pull request-ээр хянуулж, code review хийсний дараа main branch руу merge хийсэн. Дараа нь Github Workflow нь CI/CD процессийг ажиллуулдаг.

4.3.2 Code standards

TypeScript, Prettier ашиглан кодын чанарыг хангасан..

4.4 Тулгарсан бэрхшээлүүд

4.4.1 Merchant Super App OAuth интеграци

Merchant Super App-ийн OAuth flow нь refresh token rotation механизмтай байсан. Энэ нь хэрэглэгчийн session-г тогтвортой байлгахад бэрхшээл үүсгэсэн.

Шийдэл: Refresh token-г өгөгдлийн санд хадгалж, access token дуусах үед автомат шинэчлэх механизм хэрэгжүүлсэн.

4.4.2 ҮЦТХТ системийн давхцал

ҮЦТХТ системд хэрэглэгч бүртгэх үед accountId болон accountNumber давхцал гарах асуудал байсан.

Шийдэл: Retry механизм хэрэгжүүлж, давхцал гарсан тохиолдолд шинэ ID, дугаар үүсгэж дахин оролдох систем бүтээсэн.

4.4.3 300,000+ мөртэй өгөгдлийн хайлт

Гадны системээс 300,000 гаруй мөртэй өгөгдлөөс 10 урттай тэмдэгт мөрөөр хайлт хийх нь удаан байсан.

Шийдэл: Индекс файл үүсгэж, binary search алгоритм ашиглан O(log n) хурдтай хайлт хэрэгжүүлсэн. Энэ нь хайлтын хурдыг дараах байдлаар сайжруулсан:

- Хайлтын хугацаа: 5-40ms
- Өмнөх шугаман хайлтын хэрэгжүүлэлтийн хугацаа: 2минут
- Санах ойн хэрэглээ: 50MB-аас бага
- Файл уншилт: Зөвхөн шаардлагатай хэсэг (4KB buffer)

5. ТУРШИЛТ, НЭВТРҮҮЛЭЛТ БА ҮР ДҮН

5.1 Туршилт

5.1.1 API туршилт

Postman ашиглан бүх API endpoint-уудыг туршиж, янз бүрийн тохиолдолд (амжилттай, алдаатай) зөв хариулт буцаах эсэхийг шалгасан.

5.1.2 Merchant Super App интеграцийн туршилт

Merchant Super App-ийн OAuth flow, refresh token механизм, invoice үүсгэлт болон төлбөрийн статус шалгалтыг туршсан.

5.1.3 ҮЦТХТ интеграцийн туршилт

ҮЦТХТ системтэй холбогдох SOAP үйлчилгээг туршиж, хэрэглэгч бүртгэх, тэмдэгт мөрөөр хайх зэрэг функцийг баталгаажуулсан.

5.2 VPS нэвтрүүлэлт

5.2.1 Серверийн тохиргоо

Төслийг VPS сервер дээр нэвтрүүлсэн:

Table 5.1: VPS серверийн хамгийн бага шаардагдах техникийн үзүүлэлт

Параметр	**Утга**
CPU	2 vCPU
RAM	2GB
Хадгалах сан	SSD
Үйлдлийн систем	Ubuntu 20.04 LTS

5.2.2 Серверийн архитектур

Docker container ашиглан нэг VPS сервер дээр ихэнх компонентийг суурилуулсан:

- **Nginx** - Reverse proxy, SSL termination, статик файл өгөх
- **Backend Container** - Node.js Express API сервер Docker image-ээр
- **Frontend Container** - Next.js статик файл Nginx container-ээр өгөх
- **MySQL** - Өгөгдлийн сан
- **Dockerfile** - Автомат нэвтрүүлэлт

5.2.3 CI/CD Pipeline

GitHub Actions ашиглан Docker-тай автомат deploy систем хэрэгжүүлсэн:

1. **Build** - Docker image-ууд build хийх (frontend, backend)
2. **Deploy** - SSH ашиглан серверт Docker image татах
3. **Restart** - Dockerfile ашиглан container-уудыг дахин эхлүүлэх

5.3 Гүйцэтгэлийн үзүүлэлтүүд

5.3.1 Хариу өгөх хугацаа

Системийн хариу өгөх хугацааг хэмжсэн үр дүн:

- **API хүсэлт** - Дундаж 50-200ms
- **Тэмдэгт мөрний хайлт** - 5-40ms (binary search)
- **ҮЦТХТ SOAP үйлчилгээ** - 1-5 секунд
- **Merchant Super App API** - 100ms-1 секунд

5.4 Үр дүн

Төсөл амжилттай хэрэгжиж, дараах үр дүнд хүрсэн:

- Merchant Super App хэрэглэгч системд хүрэх боломжтой болсон
- ҮЦТХТ системтэй найдвартай интеграци бүрдэж, хэрэглэгчийн бүртгэл автоматжсан
- Хурдан хайлт алгоритмаар тэмдэгт мөрөөр хайх цаг хугацааг эрс багасгасан
- JWT болон refresh token механизмаар аюулгүй нэвтрэлт хангасан

6. ДҮГНЭЛТ БА СУРАЛЦАЛТ

6.1 Олж авсан туршлага

6.1.1 Техникийн чадвар

Энэхүү дадлагаар дараах техникийн чадваруудыг олж авсан:

- **Фүллстек хөгжүүлэлт** - Next.js React фронтенд болон Node.js Express бэкенд
- **TypeScript** - Type safety болон кодын чанарыг сайжруулах арга техник
- **Өгөгдлийн сангийн дизайн** - Prisma ORM, MySQL schema design, migration
- **API интеграци** - REST API, SOAP үйлчилгээ, OAuth механизм
- **DevOps** - VPS суурилуулалт, Nginx тохиргоо, PM2, CI/CD pipeline, Docker
- **Алгоритм оновчлол** - Binary search, индекс файл, файл streaming

6.1.2 Санхүүгийн салбарын мэдлэг

БиДиСЕК ҮЦК-д ажиллаж санхүүгийн салбарын тодорхой мэдлэг олж авсан:

- Брокер, дилерийн үйл ажиллагаа
- ҮЦТХТ системийн ажиллагаа
- KYC (Know Your Customer) процесс
- Төлбөрийн систем, invoice механизм
- Үнэт цаасны арилжаа, портфолио удирдлага

6.1.3 Багийн ажиллагаа

- Git workflow, code review процесс
- Agile methodology, task planning Excel-ээр
- Код стандарт тогтоох, ESLint/Prettier setup
- Техникийн баримт бичиг боловсруулах

6.2 Техникийн шийдлийн ач холбогдол

6.2.1 Binary search хайлтын давуу тал

300,000+ өгөгдлөөс тэмдэгт мөрөөр хайх алгоритм:

- **Хурд** - O(log n) алгоритм, 5-40ms хайлт
- **Санах ой** - 50MB-аас бага memory usage
- **Өргөтгөх чадвар** - Өгөгдөл 10 дахин ихэссэн ч хайлтын хурд бараг өөрчлөгдөхгүй
- **Шууд файл хандалт** - Зөвхөн шаардлагатай хэсгийг уншиж, I/O-г хэмнэнэ

6.2.2 OAuth refresh token механизм

Merchant Super App-тай интеграцийн аюулгүй байдлыг хангасан арга:

- Access token богино хугацаатай (15 минут), аюулгүй байдлыг нэмэгдүүлнэ
- Refresh token rotation хэрэглэгчийн мэдээлэл хулгайлагдахаас хамгаална
- Автомат token шинэчлэх механизм хэрэглэгчийн туршлагыг сайжруулна

6.3 Ирээдүйн хөгжүүлэлт

6.3.1 Системийн сайжруулалт

- Өгөгдлийн санг тусдаа серверт шилжүүлж scalability нэмэгдүүлэх

- Redis кэш нэмж API хариу өгөх хурдыг сайжруулах
- Docker compose ашиглан deployment хялбарчлах
- Үнэт цаасны хоёрдогч зах зээлийн арилжааны сервисийг холбох
- Бодит орчинд нэвтрүүлэх

Дүгнэлт

Энэхүү үйлдвэрийн дадлагаар БиДиСЕК ҮЦК компанид ”Мини апп” төслийг амжилттай хэрэгжүүлсэн. Төсөл нь томоохон мерчантийн аппликейшн дотор мини-апп хэлбэрээр суурилуулагдаж, компанийн дижитал шинэчлэлийн зорилгодоо хүрэхэд чухал хувь нэмэр оруулсан.

Техникийн хөгжүүлэлтийн хувьд Next.js React фронтенд, Node.js Express бэкенд, TypeScript, MySQL ашиглан SOLID зарчмыг дагасан архитектур бүтээсэн. Merchant Super App OAuth механизм, ҮЦТХТ SOAP API интеграци, 300,000+ мөртэй өгөгдлөөс binary search алгоритмаар хурдан хайлт хийх(5-40ms), үнэт цаасны real-time дата WebSocket ашиглан дамжуулах систем зэрэг олон чухал арга техникийг амжилттай хэрэгжүүлсэн.

Фүллстек веб хөгжүүлэлт, API интеграци, real-time дата боловсруулалт, Docker containerization, DevOps практик зэрэг олон чиглэлийн техникийн чадваруудыг эзэмшсэн. Мөн багийн ажиллагаа, төслийн удирдлага, Git workflow, CI/CD pipeline зэрэг мэргэжлийн ур чадваруудыг хөгжүүлсэн.

Энэхүү дадлагын үр дүнд санхүүгийн технологийн салбарт бодит ажлын туршлага олж авч, ирээдүйд илүү том, нарийн төвөгтэй системүүд хөгжүүлэх, санхүүгийн технологийн салбарт хувь нэмрээ оруулах чадварыг эзэмшсэн.

Bibliography

[1] Next.js Documentation, https://nextjs.org/docs

[2] Node.js Documentation, https://nodejs.org/docs

[3] TypeScript Documentation, https://www.typescriptlang.org/docs

[4] Prisma Documentation, https://www.prisma.io/docs

[5] JWT.io Documentation, https://jwt.io/introduction

[6] Express.js Documentation, https://expressjs.com/

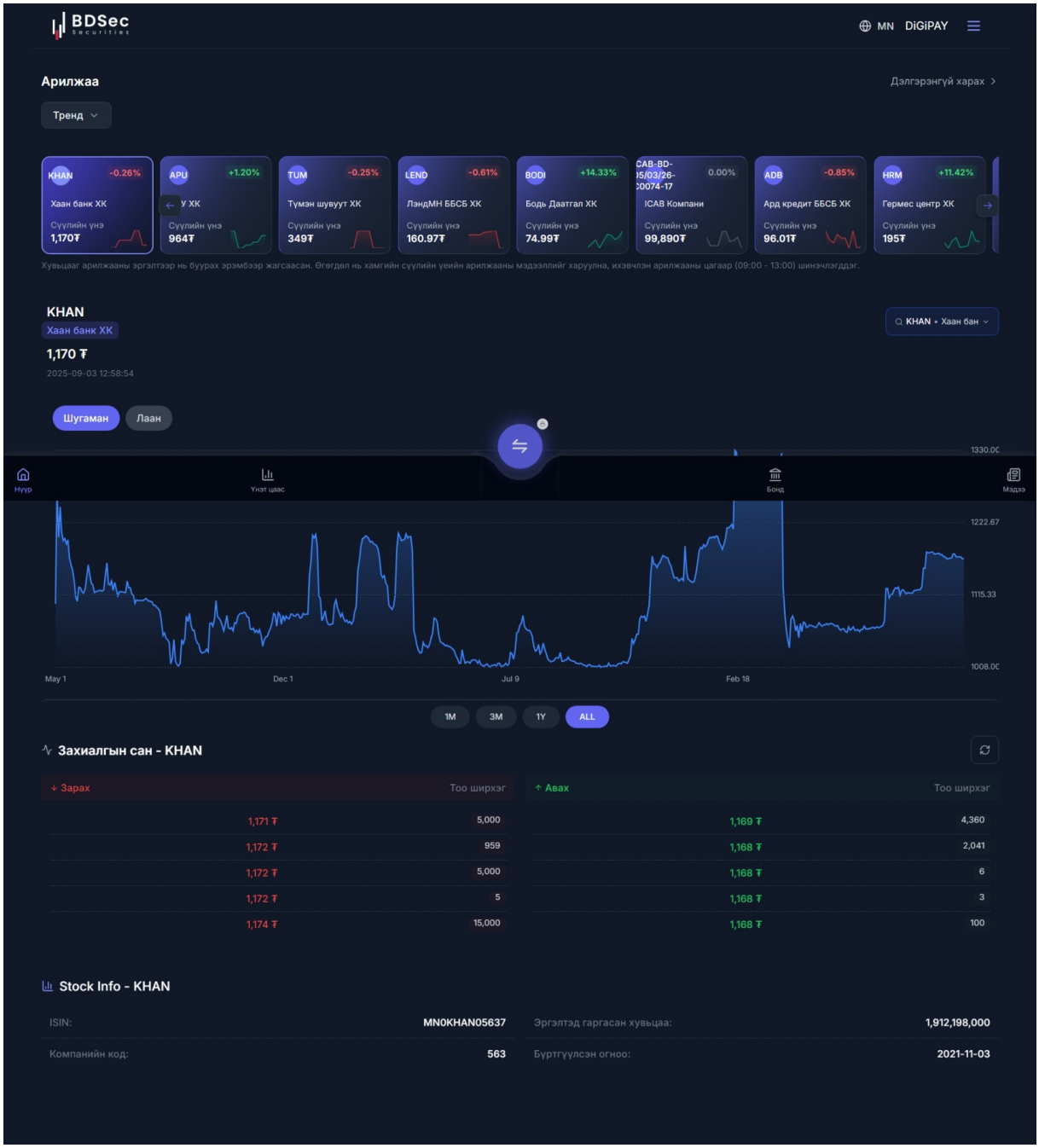
[7] MySQL Documentation, https://dev.mysql.com/doc/

[8] React Documentation, https://react.dev/

[9] Docker Documentation, https://docs.docker.com/

А. НҮҮР ХУУДАС

NextJS дээр Figma-гийн дизайныг амжилттай хэрэгжүүлсэн.



Зураг А.1: Нүүр хуудас

B. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

B.1 Merchant Super App OAuth нэвтрэлт

```
router.get("/merchantA", async (req, res) => {
  const { code, scope } = req.query;

  // Merchant OAuth code - token -
  const tokenRes = await merchantService.getCode(code as string);
  const merchantInformation = await merchantService.getUserInfo(
    tokenRes.access_token
  );

  //
  const user = await db.user.findFirst({
    where: { merchantUser: { userIdMerchant: merchantInformation.userId
       ↪ } },
    include: { user: true, merchantUser: true }
  });

  let token;
  if (user) {
    //                JWT
    const payload = { id: user.user.id, role: "USER", type: "merchant"
       ↪ };
    token = jwt.sign(payload, env.JWT_SECRET);

    //        Refresh Token -
  } else {
    //
    token = jwt.sign({ id: merchantUser.id, role: "USER" }, env.
       ↪ JWT_SECRET);
  }

  // JWT      cookie -
  res.cookie("token", token, {
    httpOnly: false,
    secure: process.env.NODE_ENV === "production",
    sameSite: "lax",
    maxAge: 24 * 60 * 60 * 1000 // 1 day
  });

  res.redirect(process.env.PROD_FRONTURL);
});
```

Код B.1: Merchant App Authorization Redirect Handler

B.2 Cron job болон тогтмол синхронизаци

```
// Cron job -
const cron = require('node-cron');

cron.schedule(crontabExpression, async () => {
```

```
  try {
    console.log('Starting securities data sync...');

    //     API  -
    const latestSecuritiesData = await fetchSecuritiesFromAPI();

    //
    const previousData = await getStoredSecuritiesData();

    // Delta            (            )
    const deltaChanges = calculateSecuritiesDelta(previousData,
      ↪ latestSecuritiesData);

    if (deltaChanges.length > 0) {
      //
      await updateSecuritiesInDatabase(deltaChanges);

      // WebSocket  -
      broadcastSecuritiesUpdate(deltaChanges);

      console.log(`Updated ${deltaChanges.length} securities`);
    }
  } catch (error) {
    console.error('Securities sync failed:', error);
  }
});

// Delta
function calculateSecuritiesDelta(oldData, newData) {
  const changes = [];

  newData.forEach(newSecurity => {
    const oldSecurity = oldData.find(old => old.symbol === newSecurity.
      ↪ symbol);

    if (!oldSecurity || oldSecurity.price !== newSecurity.price) {
      changes.push({
        symbol: newSecurity.symbol,
        oldPrice: oldSecurity?.price || 0,
        newPrice: newSecurity.price,
        change: newSecurity.price - (oldSecurity?.price || 0),
        timestamp: new Date()
      });
    }
  });

  return changes;
}
```

Код B.2: Securities Data Synchronization Cron Job