МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
ХЭРЭГЛЭЭНИЙ ШИНЖЛЭХ УХААН, ИНЖЕНЕРЧЛЭЛИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

Нямдоржын Энхболд

Урамшууллын системийн цогц хөгжүүлэлт

(Full stack loyalty system developing)

Програм хангамж (D061302)
Үйлдвэрлэлийн дадлагын тайлан

Улаанбаатар

2023 оны 09 сар

МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
ХЭРЭГЛЭЭНИЙ ШИНЖЛЭХ УХААН, ИНЖЕНЕРЧЛЭЛИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

Урамшууллын системийн цогц хөгжүүлэлт
(Full stack loyalty system developing)

Програм хангамж (D061302)
Үйлдвэрлэлийн дадлагын тайлан

Гүйцэтгэсэн: _____ Н.Энхболд (20B1NUM0102)

Улаанбаатар

2023 оны 09 сар

Зохиогчийн баталгаа

Миний бие Нямдоржын Энхболд ”Урамшууллын системийн цогц хөгжүүлэлт” сэдэвтэй судалгааны ажлыг гүйцэтгэсэн болохыг зарлаж дараах зүйлсийг баталж байна:

- Ажил нь бүхэлдээ эсвэл ихэнхдээ Монгол Улсын Их Сургуулийн ”Үйлдвэрлэлийн дадлага (INTE401)” хичээлийг тооцуулахаар дэвшүүлсэн болно.
- Энэ ажлын аль нэг хэсгийг эсвэл бүхлээр нь ямар нэг их, дээд сургуулийн зэрэг горилох, хичээл тооцуулахаар 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

Миний бие Н. Энхболд нь үйлдвэрлэлийн дадлагын хугацааны хүрээнд ”MOGUL” группийн салбар компани болох ”Nomadic Software Solution” ХХК-д хөгжүүлэгчдийн ашигладаг технологиудыг судалж, сурсан мэдлэгээрээ HPoint төслийг эхлүүлсэн бөгөөд нийт 5 хүний бүрэлдэхүүн бүхий багийг ахлаж төслийг эхний байдлаар хэрэгжүүлсэн билээ.

HPoint төслийн гол зорилго нь ”Hoome” платформд урамшууллын програмыг нэвтрүүлснээр хэрэглэгчдийн тоог үнэмлэхүйц байдлаар өсгөх юм.

Төслийн баг нь архитектор, back-end хөгжүүлэгч, програм хангамж хөтөлбөрийн 2 дадлагын оюутнуудаас бүрдэх бөгөөд төслийн хүрээнд мобайл хөгжүүлэлт, микросервис хөгжүүлэлт, өгөгдлийн сангийн зохиомж, хэрэглэгчийн интерфейсийн зохиомж зэрэг ажлууд өрнөсөн болно.

Үйлдвэрлэлийн дадлагын хугацаанд эдгээр ажлуудад ашиглагдах технологиудыг судалж, хэрэгжүүлэлтүүдийг гүйцэтгэсэн билээ.

1. ТӨЛӨВЛӨГӨӨ

Table 1.1: Үйлдвэрлэлийн дадлагын төлөвлөгөө

№	Гүйцэтгэх ажил	Хугацаа	Төлөв	Удирдагчийн үнэлгээ
1	Spring boot технологи болон JHipster-ийн талаар судлах	1 өдөр	Гүйцэгтэсэн	10/10
2	Keycloak технологийн талаар судлах	1 өдөр	Гүйцэгтэсэн	10/10
3	Өмнө хөгжүүлэгдсэн микросервис дээр feature нэмэх	3 өдөр	Гүйцэгтэсэн	10/10
4	Flutter-ийн BLoC технологийн талаар судлах	1 өдөр	Гүйцэгтэсэн	10/10
5	Hoome мобайл аппликейшн дээр зогсоолын төлбөр төлөх хэсэг дээр bug засах, карт холбох feature нэмэх	4 өдөр	Гүйцэгтэсэн	10/10
6	Hoome платформын хэрэглэгчийн тоог өсгөх шийдэл олох, зохиомжийг гаргах	1 өдөр	Гүйцэгтэсэн	10/10
7	Микросервис хөгжүүлэлтийг хийх API endpoint-уудыг бэлдэх (Багаар)	6 өдөр	Гүйцэгтэсэн	10/10
8	”Hoome” мобайл аппликейшн дээр шийдлээ оруулж өгөх, дизайныг хэрэгжүүлэх	2 өдөр	Гүйцэгтэсэн	10/10
9	Микросервис дээр integration тест бичих	2 өдөр	Гүйцэгтэсэн	10/10

2. БАЙГУУЛЛАГЫН ТУХАЙ

”MOGUL” групп нь 1997 онд компьютер, компьютерийн тоног төхөөрөмж нэвтрүүлэх, үйлчилгээ үзүүлэх зорилгоор үйл ажиллагаагаа эхэлж байсан бөгөөд Мэдээллийн Технологийн чиглэлээр төрөлжин үйл ажиллагаа явуулдаг 6 компанитайгаар 26 дахь жилдээ ажиллаж байгаа. Нийт 380 гаруй ажилтан, тэдгээрийн 200 гаруй ажилтан нь инженерүүд байдаг. Мэдээллийн технологийн дэд бүтэц, тоног төхөөрөмжийн худалдаа, үйлчилгээ, Мэдээллийн болон биет аюулгүй байдал, Программ хангамж үйлдвэрлэл, Цахим засаг, Клауд болон Менежед үйлчилгээ, Дата болон AI, салбаруудын мэдээллийн технологийн шийдэл чиглэлээр үйл ажиллагаа явуулдаг.

2.1 Компаний тухай

”MOGUL” групп нь 2023 оны өвлийн улиралд 6 салбар компанитай байсан бөгөөд эдгээрт:

- ITZone ХХК
- Новелсофт ХХК
- Могул Сервис энд Саппорт ХХК
- Дижитал Воркс ХХК
- Дижитал Повер ХХК
- Могул Экспресс ХХК

компаниуд орно. ”Новелсофт” ХХК-ийн бизнесийн үйл ажиллагаа өргөжсөнөөр ”Nomadic Software Solution” ХХК компанийг 2023 оны хаврын улирал үүсгэн байгуулсан.

2.2 Үйл ажиллагаа

”Новелсофт” ХХК нь захиалгат програм хангамж хөгжүүлэлт, дата аналитик, дата менежмент, мэргэжлийн үйлчилгээ (outsourcing) зэрэг үйл ажиллагаа явуулдаг байсан ба үүнээс програм хангамж болон дата аналитик гэх үйл ажиллагааны хүрээнд 2 хуваагдан ”Nomadic SS” ХХК бий болсон.

”Nomadic SS” ХХК нь одоогоор програм хангамж хөгжүүлэлт, мэргэжлийн үйлчилгээ (outsourcing) зэрэг үйл ажиллагааг явуулдаг.

2.3 Технологиуд болон системүүд

Компаний хувьд хөгжүүлэгдэж буй системүүд нь цогц системүүд байдаг ба дийлэнх хөгжүүлэгчдийн туршлага, системийн зохиомжоос хамааран

- Front-end
  - Mobile: Flutter - BLoC (Business Logic Components)
  - Web: Angular - Primeng
  - Desktop: .NET
- Server
  - Java - JHipster Spring boot
  - OAuth - Keycloak
  - Cassandra
  - Kafka
  - Redis
  - Prometheus

технологиудыг best practice болгон мөрдөж ашигладаг. Хэрэглэгчийн шаардлагаас үүдэн өөр технологиудыг ашигласан тохиолдлууд ч байдаг.

3. HOOME ТӨСӨЛ

Үйлдвэрлэлийн дадлагын хүрээнд одоо идэвхитэй явагдаж буй ”Hoome” төсөл дээр ажилласан бөгөөд энэхүү төсөл нь Hoome мобайл аппликейшн, Hoome сөх веб, Hoome контор веб зэрэг системүүдээс бүрдэх цогц систем юм. Одоогоор бүртгэлтэй 24000 хэрэглэгч байгаа ба тэдгээрийн 13000 нь идэвхитэй хэрэглэгч.

Бизнесийн үйл ажиллагаа нь голчлон хэрэглэгчдийн өдөр тутмын амьдрал дээр тулгуурласан бөгөөд СӨХ-өөс гадна машины зогсоолын хэсгийг нэвтрүүлээд байгаа билээ. Зогсоолын хэсэг нь бие даасан систем бөгөөд CCP(Cloud Car Parking) гэх төслийн хүрээнд идэвхитэй хэрэгжиж байгаа болно. Уг системийг ”Hoome” мобайл аппликейшнд feature байдлаар оруулж өгсөн байгаа.

”Novelsoft” ХХК-ийн бүтээгдэхүүн болох ”Homebook” СӨХ-ийн системийг сайжруулж сошиал коммунити аппликейшн болгож 2022 онд улмаар ”Hoome” гэх шинэ төслийг эхлүүлсэн.

3.1 Хотхон

”Hoome” платформын хотхоны систем нь СӨХ-ийн менежмент, төлбөр, оршин суугчдын бүх төрлийн харилцааг удирдах тусгай систем болон түүнийг иргэдэд хүргэх ”Hoome” сошиал аппын цогц бөгөөд

- СӨХ-ийн төлбөр бодолт
- Хэрэглэгчийн төлбөр төлөлт
- Автомат иБаримт гаргах, илгээх
- Тайлан гаргах
- Хотхоны бүлгэм үүсгэх, сошиал пост, чат

- Оршин суугчдын жагсаалт, мэдээлэл, автомат бүртгэл
- Зогсоол, агуулах удирдлага

зэрэг функцуудээс бүрдэнэ.

3.2 Контор

”Hoome” платформын хотхоны систем нь конторын төлбөр бодолт, үйлчилгээний захиалга авах гэх мэт бүх үйл ажиллагааг удирдах, системтэй.

- Бүх хэрэглэгчийн жагсаалт
- Конторын төлбөр бодолт
- Хэрэглэгчийн төлбөр төлөлт
- Тайлан гаргах
- Тоолуурын заалт
- Тариф удирдлага тохиргоо
- Автомат иБаримт гаргах, илгээх

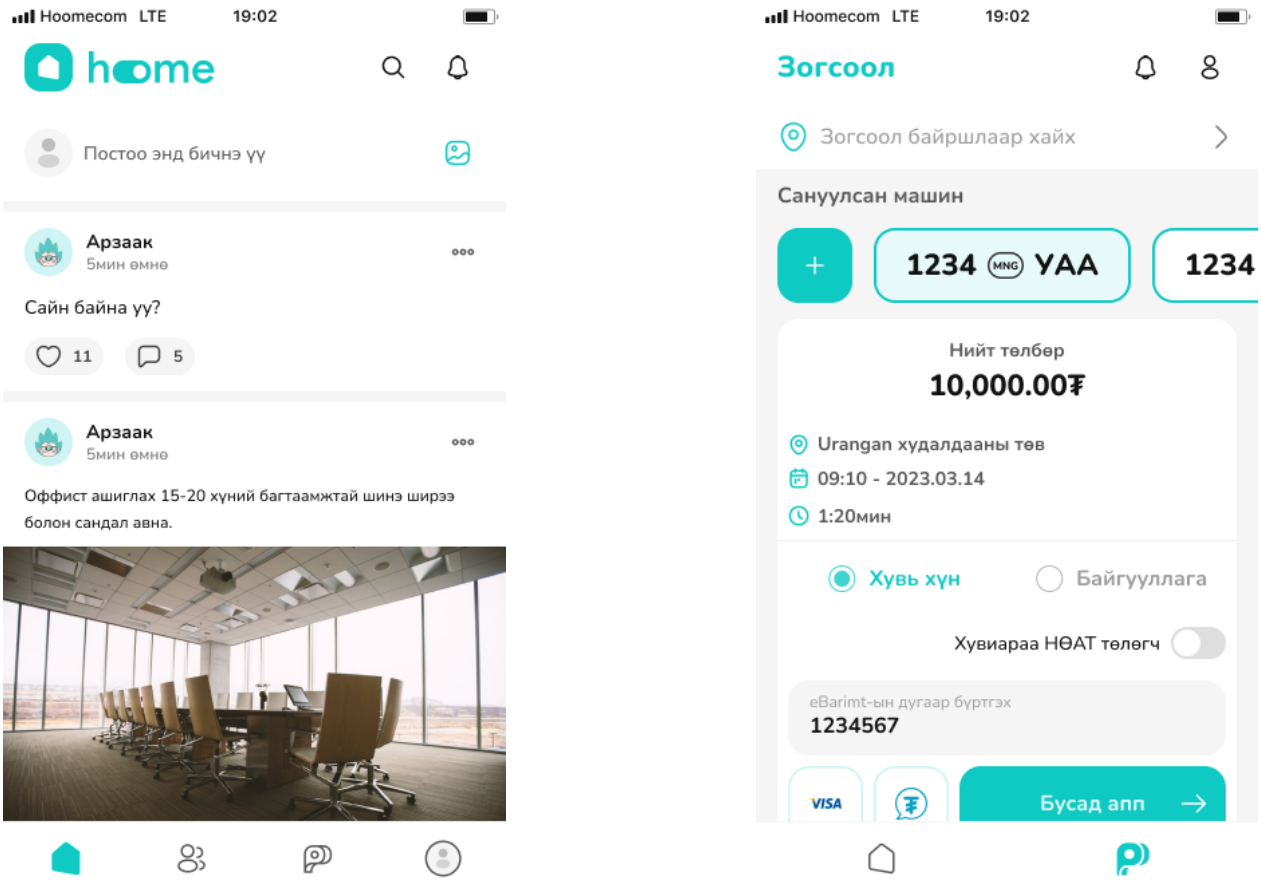
зэрэг функцуудээс бүрдэнэ.

3.3 Мобайл аппликейшн

”Hoome” платформын мобайл аппликейшн нь СӨХ-ийн төлбөр төлөгч болон Hoome-д бүртгэлтэй зогсоолд машинаа тавьсан хэрэглэгчдэд зориулагдсан. Хэрэглэгч өөрийн хотхоны СӨХ-д элссэнээр сар бүрийн төлөх зардлууд болон тооцоог нэг дороос харах боломжтой болно. Зогсоолын төлбөр буюу ”Cloud Car Parking” систем нь Hoome-д бүртгэлтэй зогсоолууд дээр машин тавьсан хэрэглэчийг төлбөрийг орсон хугацаанаас нь бодож тооцдог бөгөөд давхар

orgware буюу хаалгач хүнийг ажиллуулдаг билээ. Тухайн зогсоолын төлбөрийг зөвхөн ”Hoome” биш, зарим тохиолдолд ”Toki” аппликейшнийн хэрэглэгчид аппликейшнээрээ дамжуулан төлбөрөө төлдөг бөгөөд энэ нь Hoome-CCP-ийн зэрэгцээ орших ижил төстэй систем болно.

Энэхүү асуудлыг шийдэхээр шийдэл дэвшүүлж дадлагын хугацаанд ”Hoome” платформын front-end болон back-end хөгжүүлэлтүүдийг хийж гүйцэтгэсэн билээ.



Зураг 3.1: Hoome - Коммунити нүүр.

Зураг 3.2: Hoome - Cloud Car Parking нүүр.

4. ТЕХНОЛОГИЙН СУДАЛГАА

4.1 Flutter - BLoC

”Hoome” платформын мобайл аппликейшн нь програмчлалын Dart хэл буюу Flutter технологийг ашиглан бичигдсэн бөгөөд түүн дотроо төлвийн менежмент сан болох BLoC(Business Logic Components)-ийг ашигладаг.

[1]BLoC нь хэрэглэгчийн интерфейсийг бизнес логикоос тусгаарлаж өгөх зорилготой сан бөгөөд event-driven архитектур дээр суурилсан байдаг.

BLoC нь таны Flutter програмын төлвийг удирдах, хэрэглэгчийн харилцан үйлчлэлийг бүтэцтэй байдлаар зохицуулах design pattern юм. Энэ нь апп доторх data stream болон удирдахын тулд event, stream-ийн тухай ойлголтыг ашигладаг.

4.1.1 Үндсэн бүтэц

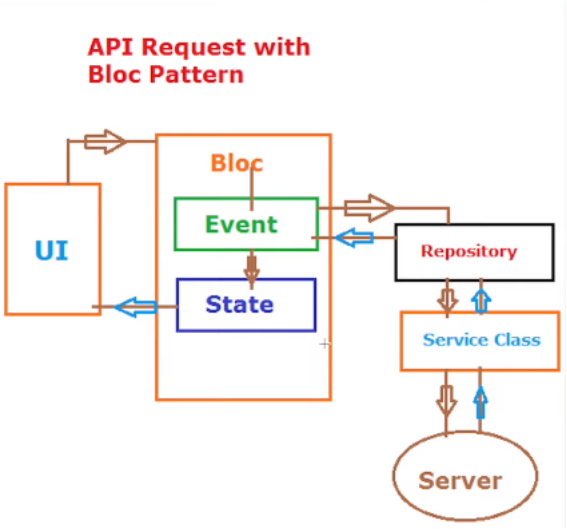
Flutter BLoC технологи нь:

1. **Events**: Event нь хэрэглэгчийн аппликейшнд үзүүлэх ямарваа нэгэн хариу үйлдэл бөгөөд тухайн event-ийг BLoC компонент хүлээн авч, логик үйлдлүүдийг хийж боловсруулснаар одоо байгаа төлвүүдийг шинэчилж шинэ төлвийг үүсгэдэг.

2. **BLoC Component**: Event-үүдийг сонсож, утга хүлээн авахад үргэлж бэлэн байдаг бөгөөд event-үүд нь өөрийн утгатай байх боломжтой. Хүлээн авсан эвентүүдийг хэрэглэгчийн интерфейст ашиглагдаж буй төлөв, төлвийн өгөгдлийг шинэчлэхэд ашигладаг.

3. **Streams**: Event-үүд нь ихэвчлэн stream байдлаар хадгалагдаж, орсон дарааллаараа ачаалладаг бөгөөд BLoC компонентүүд нь тус бүр өөрийн зааж өгсөн урсгалыг сонсож байдаг.

4. **States**: Аппликейшн дээр ашиглагдаж буй бүх төрлийн датаг state буюу төлөв гэж нэрэлж



Зураг 4.1: BLoC

байгаа бөгөөд тухайн төлөв нь BLoC компонентоор дамжуулагдан шинэчлэгдэж, улмаар хэрэглэгчийн интерфейст өөрчлөлт ороход хүргэдэг.

Эдгээр 4 үндсэн элементүүдээс бүрдэх бөгөөд хэрэглэгчийн интерфейст шинэчлэл хийх, өөрчлөлт оруулах үед интерфесийн логикоос хамааран онцгой тохиолдол(exception), алдаа(runtime error)-наас сэргийлэх боломжтой. Аппликейшнийн usability болон scalability-г илүү амар хялбар байдлаар хангаж өгдөг сан юм.

4.1.2 Хэрэгжүүлэлт

Машины дугаар бүртгэх хэсэг дээр жишээ авж үзэв.

Хэрэглэгч машины дугаарыг бүртгүүлэх үед RegisterPlateNumber гэх event явагдах бөгөөд үүнийг onSavePlateNumber функцээр сонсож, тухайн event дээр хийгдэх бизнес логикийг бичиж өгөв.

```
1 import 'dart:convert';
2
3 import 'package:bloc/bloc.dart';
4
```

```dart
part 'parking_event.dart';
part 'parking_state.dart';
class ParkingBloc extends Bloc<ParkingEvent, ParkingState> {
  //... Other event handlers

  void _onSavePlateNumber(
    RegisterPlateNumber event,
    Emitter<ParkingState> emit,
  ) async {
    emit(state.copyWith(registerStatus: RegisterStatus.loading));
    bool isRegistered = await ParkingRepository().registerParking(event
        .phoneNumber, event.plateNumber);
    if (isRegistered) {
      LocalStorage().save('phoneNumber', event.phoneNumber, const
          Duration(days: 365));
      emit(state.copyWith(localPhoneNumber: event.phoneNumber));
    }

    emit(state.copyWith(
      registerStatus: isRegistered ? RegisterStatus.success :
          RegisterStatus.failure,
      statusMessage: 'Car␣registration␣success!',
    ));
  }
}
```

Код 4.1: BLoC компонентийн машины дугаар бүртгэлийн хэрэгжүүлэлт

```dart
import 'package:dio/dio.dart';

class ParkingRepository {
  final Dio dio = Dio();

  //... Other repository methods

  Future<bool> registerParking(String phoneNumber, String plateNumber)
      async {
    final body = {
      'phoneNumber': phoneNumber,
      'plateNumber': plateNumber,
    };

    try {
      final response = await dio.post(ApiConstants.registerParkingUser,
          data: jsonEncode(body));
      return response.statusCode == 200 || response.statusCode == 204;
    } on DioError catch (_) {
      return false;
    }
  }
}
```

Код 4.2: Машины дугаарыг хэрэглэгчийн дугаарын хамтаар сервер дээр бүртгэх

4.2 Spring boot - JHipster

JHipster нь нээлттэй веб аппликейшн болон микросервисүүдийг бэлдэж өгдөг tool бөгөөд хөгжүүлэлтийн процессийг хялбарчилж, хурдлуулж өгдгөөрөө давуу талтай. Ихэвчлэн сервер талдаа Spring boot аппликейшнийг бэлдэж өгдөг бол веб дээр хэрэглэгч талдаа Angular, React зэрэг технологиудыг ашигласан бэлэн төслийг үүсгэж өгдөг.

[2]React - Redux болон Angular дээр хөгжүүлэгчээс ямар зам (route) болон ямар authorization/authentication технологи ашиглахыг аван тохирох аппликейшнийг үүсгэж өгдөг. Энэхүү ажил нь ойролцоогоор хөгжүүлэгчийн хувьд 5-10 ажлын өдөр шаарддаг бол JHipster-ийн тусламжтайгаар төслийн төвөгтэй олон тохиргоонд цаг үрэлгүйгээр шууд үндсэн бизнес логикоо кодлох боломжийг олгодог.

Spring boot дээр ашиглахдаа authorization-ээс гадна model-оо хүртэл үүсгэж зааж өгөх боломжтой бөгөөд хөгжүүлэгч ERD-аа jdl дээр бичиж өгөн JHipster-ээр тухайн model-той холбоо бүхий бүх кодуудыг бэлдүүлэх боломжтой.

4.2.1 Spring boot support

JHipster-ийг ашиглан best practice буюу дагаж мөрдвөл хамгийн зохих, олон хөгжүүлэгчдийн санал нийлсэн байдлаар кодлосон boilerplate төслийг үүсгэх боломжтой ба

- Security - Authentication/Authorization
- Өгөгдлийн сангийн интеграц - JPA/Hibernate
- Build tools - Maven/Gradle
- Docker болон Kubernetes support
- Unit testing

зэрэг олон setup бүхий зүйлсийг хөнгөвчлөх боломжтой.

4.2.2 Хэрэгжүүлэлт

JHipster CLI ашиглан keycloak интеграц хийсэн микросервис үүсгэсэн байдал.

Энэхүү CLI коммандыг ашиглан эхний байдлаар микросервисийг үүсгэнэ.

```
jhipster
```

```
spring:
security:
  oauth2:
    client:
      provider:
        oidc:
          issuer-uri: http://your-keycloak-server/auth/realms/your-
              realm
      registration:
        oidc:
          client-id: your-client-id
          client-secret: your-client-secret
    resource:
      userInfoUri: http://your-keycloak-server/auth/realms/your-realm/
          protocol/openid-connect/userinfo
```

Код 4.3: Keycloak realm-тай холбох Yaml тохиргоо

Entity үүсгэх CLI комманд

```
jhipster import-jdl myErd.jdl
```

Үүний дараагаар ”mvnw” script файлыг үүсгэж өгөх бөгөөд микросервисийг асаахдаа

```
./mvnw
```

```
entity Person {
  keycloakId String,
    username String,
    firstname String,
    lastname String,
    fullname String,
    balance Integer,
    referralCode String unique,
    createdBy String,
    createdDate Instant,
    lastModifiedBy String,
    lastModifiedDate Instant,
}

entity Referral {
  keycloakId String,
    username String,
    firstname String,
    lastname String,
    fullname String,
    createdBy String,
    createdDate Instant,
}

entity PointTrans {
    amount Integer,
    transEnum TransEnum,
```

```
    description String,
    serviceEnum ServiceEnum,
    createdBy String,
    createdDate Instant,
    lastModifiedBy String,
    lastModifiedDate Instant,
}

enum TransEnum {
  INCOME, EXPENSE
}

enum ServiceEnum {
  CCP, SOH, ITSTORE, SYSTEM
}

relationship OneToMany {
  Person to Referral{person required},
    Person to PointTrans{person required},
}

skipClient *
paginate * with infinite-scroll
filter *
```

Код 4.4: myErd.jdl дотор буй ERD

Үр дүнд үүсэх модел болон controller-ууд

```
1  package mn.nomadicss.service.dto;
2
3  import mn.nomadicss.domain.User;
4  public class UserDTO {
5      private String id;
6      private String login;
7
8      public UserDTO() {
9          // Empty constructor needed for Jackson.
10     }
11     public UserDTO(User user) {
12         this.id = user.getId();
13         // Customize it here if you need, or not, firstName/lastName/
               etc
14         this.login = user.getLogin();
15     }
16     public String getId() {
17         return id;
18     }
19     public void setId(String id) {
20         this.id = id;
21     }
22     public String getLogin() {
23         return login;
24     }
25     public void setLogin(String login) {
26         this.login = login;
```

```
27      }
28      @Override
29      public String toString() {
30          return "UserDTO{" +
31              "id='" + id + '\'' +
32              ",␣login='" + login + '\'' +
33              "}";
34      }
35  }
```

Код 4.5: UserDTO.java

```
package mn.nomadicss.web.rest;
...
@RestController
@RequestMapping("/api")
public class PersonResource {
  private final Logger log = LoggerFactory.getLogger(PersonResource.
     class);
    private static final String ENTITY_NAME = "hoomepointPerson";

    @Value("${jhipster.clientApp.name}")
    private String applicationName;
    private final PersonService personService;
    private final PersonRepository personRepository;
    private final PersonQueryService personQueryService;

    public PersonResource(PersonService personService, PersonRepository
        personRepository, PersonQueryService personQueryService) {
        this.personService = personService;
        this.personRepository = personRepository;
        this.personQueryService = personQueryService;
    }

    @PostMapping("/people")
    ...
    @PutMapping("/people/{id}")
    ...
    @PatchMapping(value = "/people/{id}", consumes = { "application/
```

```
        json", "application/merge-patch+json" })
    ...
    @GetMapping("/people")
    ...
    //... More controllers for data access
}
```

Код 4.6: UserDTO.java

```java
package mn.nomadicss.repository;

import java.util.List;
import java.util.Optional;
import mn.nomadicss.domain.User;
import org.springframework.data.domain.*;
import org.springframework.data.jpa.repository.EntityGraph;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

/**
 * Spring Data JPA repository for the {@link User} entity.
 */
@Repository
public interface UserRepository extends JpaRepository<User, String> {
    Optional<User> findOneByLogin(String login);

    @EntityGraph(attributePaths = "authorities")
    Optional<User> findOneWithAuthoritiesByLogin(String login);

    Page<User> findAllByIdNotNullAndActivatedIsTrue(Pageable pageable);
}
```

Код 4.7: UserRepository.java үүсгэсэн байдал

Цаашлаад үүсгэсэн сервисүүд болон тохиргооны файлууд, docker support файлууд, нэгжийн тестүүдийг хавсраагүй болно.

5. АСУУДАЛ БА ШИЙДЭЛ

Дадлагын хүрээнд ”Hoome” платформын микросервисүүд дээр ажиллахад програмчлалын түвшний олон асуудлуудтай тулгарсан бөгөөд шийдлүүдийг хавсаргав.

5.1 Админ dashboard

homebook-metrics сервис нь хэрэглэгчдийн интеракц болон бусад үйл ажиллагаанууд, төлбөр тооцоо бүхий үйл явцыг удирдлагад зориулан харуулах зорилготой бөгөөд Prometheus, Grafana зэрэг системүүдийг ашигладаг.

Prometheus-ийг ашиглан тодорхой хугацааны завсар, өдөр тутмын тодорхой цагт өгөгдлийн сангаас уншилт хийж, тухайн уншсан өгөгдлийг HTTP protocol-оор бусад сервисүүдэд нээлттэй болгож өгдөг. Харин Grafana нь тухайн prometheus-ийн scrape хийсэн өгөгдлийг олон граф болон visualize tools ашиглан хэрэглэгчид харуулдаг.

Эдгээр технологиудыг ашиглан удирдлагын dashboard хийсэн бөгөөд өгөгдлийн сан нь cassandra (CQL) дээр ажилладаг бөгөөд cassandra өгөгдлийн сан нь өгөгдлөө илүү найдвартай ажиллагааны хүрээнд cluster болгон хуваадаг. CQL ашиглан хэрэглэгч бүрийн тохирох датаг уншилт хийхэд өгөгдлийн сангийн зохиомжоос хамааран unique key хийж өгөөгүйгээс үүдэн group-лэх боломжгүй болсон байсан. Үүнээс үүдэн сервис нь бүх датаг өөр дээрээ аван database operation-уудыг хийх болсон билээ.

Логик: Өгөгдлийн сангаас уншилт хийгээд өөр дээрээ боловсруулан тухайн хэрэгцээт id бүхий хэрэглэгчдийн мэдээллийг keycloak-аас уншиж авах.

Жишээ: **Хамгийн их like дарсан хэрэглэгчийн тоо**

```
SimpleStatement statement = SimpleStatement
  .builder(
      "create␣materialized␣view␣if␣not␣exists␣" +
      keyspaceName +
      ".most_liked_users␣as␣select␣*␣from␣reaction␣where␣type='UP'␣and␣
```

```
          actor␣is␣not␣null␣and␣activityid␣is␣not␣null␣and␣type␣is␣not␣
          null␣and␣parent␣is␣not␣null␣and␣id␣is␣not␣null␣primary␣key(
          actor,␣activityid,␣type,␣parent,␣id);"
  )
  .setConsistencyLevel(DefaultConsistencyLevel.LOCAL_QUORUM)
  .setTimeout(Duration.ofSeconds(10))
  .build();
session.execute(statement);

PreparedStatement ps = session.prepare("select␣actor,␣actorname,␣count
   (*)␣as␣likes_count␣from␣most_liked_users␣group␣by␣actor");
ResultSet rs = session.execute(ps.getQuery());

List<Map.Entry<String, Integer>> list = new ArrayList<>();

for (Row user : rs) {
  list.add(Map.entry(user.getString("actorname"), Math.toIntExact(user.
     getLong("likes_count"))));
}

return list;
```

Код 5.1: Materialized view үүсгэн уншилт хийж буй байдал

```
List<Map.Entry<String, Long>> mostLikedUsers = getMostLikedUsers();
mostLikedUsers.sort(Map.Entry.comparingByValue(Comparator.reverseOrder
    ()));

int gaugeBuiltCount = 0;
for (Map.Entry<String, Long> entry : mostLikedUsers) {
    if (gaugeBuiltCount >= mostLikedUsersLimit) {
        break;
    }
    try {
        Gauge
            .builder("homebook.mostLikedUsers", () -> entry.getValue())
            .tag("user_name", getUsernameFromUuid(entry.getKey()))
            .register(registry);
        gaugeBuiltCount++;
    } catch (NotFoundException nfe) {
        log.error("userId: {} not found in keycloak", entry.getKey());
    }
}
return Mono.empty();
```

Код 5.2: Keycloak-аас хэрэглэгчийн мэдээллийг авч буй байдал

5.2 Retry стратеги

Kafka image processing consumer сервис нь хэрэглэгчийн оролт буюу оруулсан зурагны хэмжээнээс хамааран compress хийхдээ эхний оролдлогоор exception буцаах асуудлыг илрүүлсэн бөгөөд үүнд Kafka @RetryableTopic annotation-ийг ашигласан хэдий ч тухайн kafka-д зориулсан annotation-ий хөгжүүлэлтийн баригдмал байдлаас үүдэн Spring boot-ийн өөрийн @Retryable болон @Recover annotation-уудыг ашигласан билээ.

```
@KafkaListener(topics = "#{'${kafka.image-topic.topic-name}'}", groupId
    = "#{'${kafka.image-topic.topic-group-id}'}")
  @Retryable(value = {Exception.class}, maxAttemptsExpression = "#{'${
     kafka.image-topic.consumer.attempts}'}", backoff = @Backoff(
     delayExpression = "#{'${kafka.image-topic.consumer.backoff.delay
     }'}", multiplierExpression = "#{'${kafka.image-topic.consumer.
     backoff.multiplier}'}"))
  public void consumeMessage(String imageTopicJson) throws Exception {
    // ... Logics
      imageTopic = imageService.process(imageTopic);
      cloudClientService.setObjectStorage(imageTopic);
  }
  @Recover
  public void recoverAfterRetryFail(Exception ex, String imageTopicJson
     ) {
      // ... Logics
      try {
          String exceptionMessage = objectMapper.writeValueAsString(
             messageMap);
          config.kafkaTemplate().send(dltTopic, exceptionMessage);
      } catch (JsonProcessingException e) {
```

```
15            e.printStackTrace();
16        }
17    }
```

Код 5.3: Retryable-ийг хэрэгжүүлсэн байдал

Зураг 5.1: Хамгийн их пост хийсэн хэрэглэгчдийн dashboard

Зураг 5.2: Хамгийн их пост хийсэн хэрэглэгчдийн dashboard

6. УРАМШУУЛЛЫН СИСТЕМ

6.1 Төслийн зорилго

Hoome аппликейшнд шинэ хэрэглэгчдийг татах, нийт хэрэглэгч дунд аппликейшны хэрэглээг өргөжүүлэх, улмаар Hoome аппликейшны зах зээлд өрсөлдөх чадварыг нэмэгдүүлэх зорилгоор тус аппликейшн дээр оноо цуглуулах системийг хэрэгжүүлж байна. Хэрэглэгчид оноо (цаашид HPoint гэж нэрлэнэ) цуглуулж түүгээрээ олон төрлийн хөнгөлөлт, урамшууллыг эдлэх боломжийг нээнэ. Цаашлаад Hoome аппликейшныг цахим мөнгөний хэтэвчтэй болгох эхлэлийг тавихад энэ төслийн зорилго оршино.

6.2 Хамрах хүрээ

Энэхүү төслийн хэтийн зорилго нь цахим мөнгөний хэтэвчтэй болох, түүний хэрэглээг өдөр тутмын амьдралд оруулах юм. Цахим хэтэвч гэдэг нь Монгол улсын төлбөрийн үндсэн хэрэгсэлт “төгрөг” болон түүнтэй дүйцэхүйц солилцоог хийх боломжой байх ёстой. Энэхүү үндсэн зорилго хүртэл дараах үе шатууд хэрэгжинэ:

1. Урамшууллын оноо цуглуулах, зарцуулах боломжтой болох
    - Цуглуулах - Дахин давтагдашгүй хэрэглэгчийн редим кодтой байх. Түүгээр уригдсан хүмүүсийн лимит, оноог тооцон цуглуулах
    - Зарцуулах - Оноог төгрөгтэй дүйцүүлэн Hoome аппликейшнээр хийх боломжтой төлбөрт оролцуулах
2. Бусад дижитал бүтээгдэхүүн, платформд ашиглагддаг хэтэвчтэй интеграци хийх боломжтой болгох
    - ITStore (cody) хэтэвчнээс үлдэгдэл төгрөгийг HPoint болгон хөрвүүлж оноо болгох

- HPoint оноог буцаагаад Itstore хэтэвч рүү хөрвүүлэх

3. Хэтэвчтэй бусад төрлийн дижитал бүтээгдэхүүнээ холбох, HPoint болгон цуглуулдаг болгох

6.3 Шаардлага

Table 6.1: HPoint ФШ

ID	Шаардлага	Тайлбар	Төрөл
ФШ01	Хэрэглэгч шинэ хэрэглэгчдийг урих өөрийн гэсэн давтагдашгүй кодтой байна.	Тухайн кодоо бусдад өгөх замаар шинэ хэрэглэгчдийг урина.	Зайлшгүй
ФШ02	Хэрэглэгч аппликейшинд шинээр бүртгүүлэхдээ найзын урилгаа оруулах талбартай байна.	Уригдсан хэрэглэгч тухайн талбарт урилгын кодоо оруулна.	Зайлшгүй
ФШ03	Шинэ хэрэглэгч утасны дугаараа баталгаажуулсны дараа урилга баталгааждаг байна.	Ингэснээр хуурамч хэрэглэгч үүсэхээс сэргийлэхийн зэрэгцээ байгууллагаас гарах зарлагын тааз үнийг тооцоолох боломжтой болно.	Зайлшгүй
ФШ04	Хэрэглэгч бүр өөрийн гэсэн HPoint оноог цуглуулдаг хэтэвчтэй байна.	Хэрэглэгч тухайн оноогоо ашиглаж хөнгөлөлт, урамшуулал авахад тохирох дүнгээр хэтэвчийг нь шинэчилдэг байх.	Зайлшгүй
ФШ05	Хэрэглэгч цуглуулсан HPoint оноогоо ашиглан СӨХ-ийн төлбөрөөс хасуулах боломжтой байна.	Зөвхөн Hoome-д нэвтэрсэн СӨХ-тэй өрхүүдэд хүчинтэй	Зайлшгүй

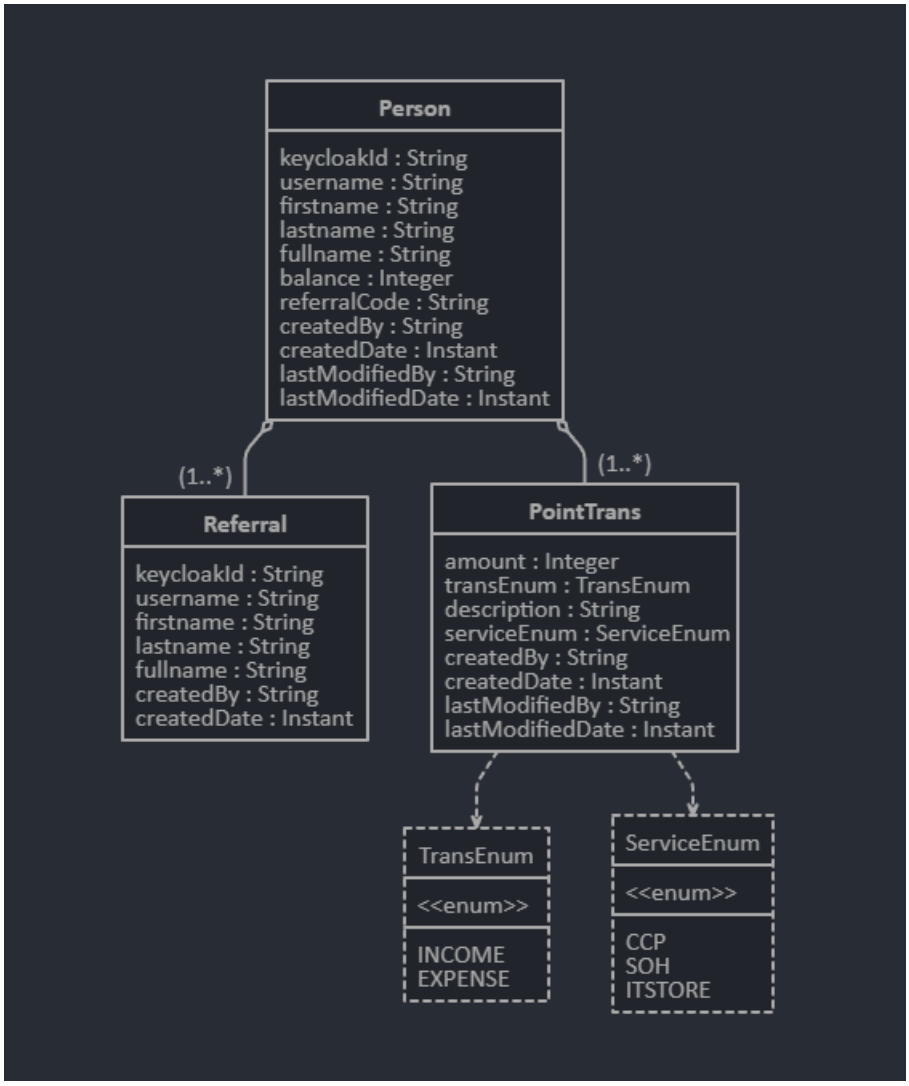
ФШ06	Хэрэглэгч цуглуулсан HPoint оноогоо ашиглан зогсоолын төлбөр төлөх боломжтой байна.	Зөвхөн CCP ашигладаг зогсоолуудад хүчинтэй	Зайлшгүй
ФШ07	Хэрэглэгч урилгаар бүртгүүлэхэд түүнийг урисан болон урилга хүлээн авсан хэрэглэгч тус бүр 1000 HPoint авна.	Урисан ба уригдсан хүмүүст хоёуланд нь HPoint бэлэглэнэ.	Зайшлгүй
ФШ08	Нэг хэрэглэгч сард 5-аас илүүгүй шинэ хэрэглэгч урих хязгаартай байна.		Зайлшгүй
ФШ09	Хэрэглэгч ITStore дээр бүртгэлтэй бол ITStore-ын хэтэвч рүүгээ HPoint-оо оруулах,	ITStore-ын хэтэвчин дэх үлдэгдлээ Hoome аппликейшн рүү HPoint болгон хөрвүүлж болдог байна. Хоёр талын гүйлгээг дэмждэг байна	Зайлшгүй

7. HPOINT ХЭРЭГЖҮҮЛЭЛТ

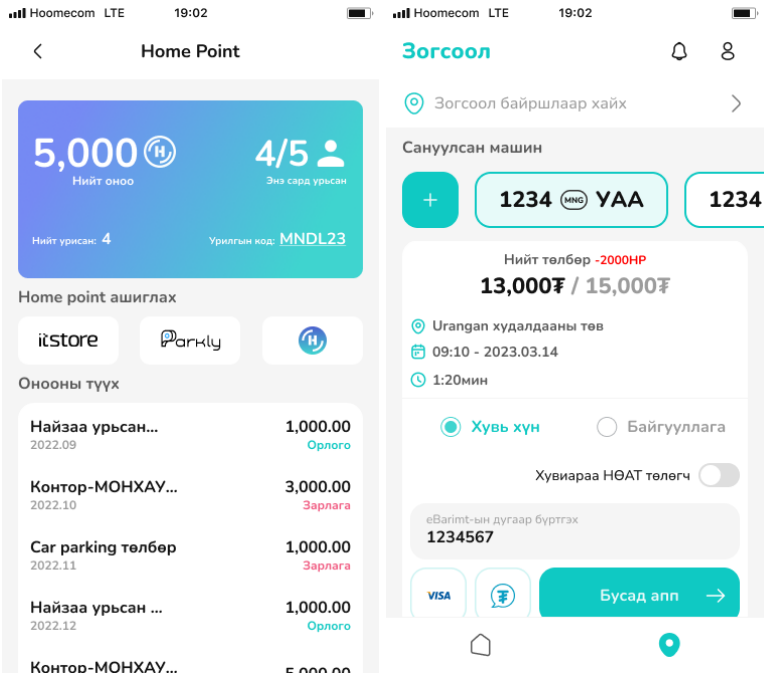
7.1 ER диаграм

Хэрэглэгч бүр өөрийн гэсэн оноотой байх бөгөөд найзаа урьснаар өөртөө болон найздаа оноо цуглуулж системийг ашиглах юм. Хэрэглэгчийн оноог PointTrans-тай сар бүр тулгаж, ямар нэгэн зөрүү үүсээгүйг баталгаажуулна.

Зураг 7.1: HPoint Entity relationship диаграм



7.2 Хэрэглэгчийн интерфейс



Зураг 7.2: HPoint нүүр болон төлбөр төлөхөд ашиглалт

```dart
Container(
  width: double.infinity,
  height: 50,
  decoration: BoxDecoration(
    color: Colors.black,
    borderRadius: BorderRadius.circular(8),
  ),
  child: InkWell(
    onTap: () {
      // ... Auth logic here
    },
    child: Row(
      mainAxisAlignment: MainAxisAlignment.center,
```

```
        children: [
          Text(
            'Login',
            style: TextStyle(
                color: Colors.white,
                fontSize: 18,
                fontWeight: FontWeight.w600),
          ),
          Icon(Icons.person, color: Colors.white),
        ],
      ),
    ),
  )
```

Код 7.1: ITStore нэвтрэх нүүр

```
BlocBuilder<HPointBloc, HPointState>(
  builder: (context, state) => Container(
      constraints: const BoxConstraints.expand(),
      child: Column(
        children: [
          HoomePointJumbotron(required point: state.point, state.
              redeemCde),
          PointProvidersScreen(tabController: widget.tabController),
          PointTransactionScreen(),
        ],)),)
```

Код 7.2: HPoint нүүр

Дүгнэлт

Үйлдвэрлэлийн дадлагын хугацаанд Mobile development, Microservices development зэрэг олон Hard skill-үүдийг сурч авсан бөгөөд багаар ажиллах, бусдыгаа төлөөлөх зэрэг олон үүрэг хариуцлага хүлээж сурсан билээ. Цогц систем хөгжүүлэх нь хичээлийн хүрээнд үзсэн ойлголтуудыг бүх талаас нь бататгаж өгсөн бөгөөд бодит байдал дээрх хэрэглээнүүд болон асуудлуудыг шийдвэрлэхэд ихээхэн хэрэг болсон билээ. Олон хэрэглэгчтэй платформ дээр өөрийн сурсан мэдсэн зүйлсээ шингээж, хариуцлагатайгаар хөгжүүлэлтийг хийж сурсан ба онолын мэдлэг болон түүнийгээ одоогийн технологиуд дээр ашиглах нь богино хугацаанд ямар их үр бүтээмж бий болгодгийг энэхүү дадлагын хөтөлбөрөөр мэдэж авсан гэж дүгнэв.

Bibliography

[1] Why bloc?, BLoC state management library, https://bloclibrary.dev/#/whybloc

[2] JHipster tech stack, https://www.jhipster.tech/tech-stack/