

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

УДИРТГАЛ	1
1. ТӨЛӨВЛӨГӨӨ	2
2. БАЙГУУЛЛАГЫН ТУХАЙ.....	3
2.1 Компаний тухай.....	3
2.2 Үйл ажиллагаа	4
2.3 Технологиуд болон системүүд	4
3. НООМЕ ТӨСӨЛ	6
3.1 Хотхон	6
3.2 Контор	7
3.3 Мобайл аппликейшн	7
4. ТЕХНОЛОГИЙН СУДАЛГАА	9
4.1 Flutter - BLoC	9
4.2 Spring boot - JHipster	13
5. АСУУДАЛ БА ШИЙДЭЛ.....	22
5.1 Админ dashboard	22
5.2 Retry стратеги.....	25
6. УРАМШУУЛЛЫН СИСТЕМ	27
7. ИШЛЭЛ, ЗҮҮЛТ	28
7.1 Ишлэл	28
7.2 Зүүлт	28
8. ЗУРАГ	29
8.1 Зургийн хэмжээ өөрчлөх	29
8.2 Зураг эргүүлэх	30
8.3 Зургийн нэр.....	31
8.4 Зураг голлуулах	32

8.5 Зургийн чанар	32
9. ХҮСНЭГТ ОРУУЛАХ	33
9.1 Хүснэгт зурах хэрэгсэл	33
10. КОД БА АЛГОРИТМ ОРУУЛАХ	34
ДҮГНЭЛТ	36
НОМ ЗҮЙ	36
ХАВСРАЛТ	37
А. ШИНЖИЛГЭЭ ЗОХИОМЖ	38
В. КОДЫН ХЭРЭГЖҮҮЛЭЛТ	39

ЗУРГИЙН ЖАГСААЛТ

3.1	Hoome - Коммунити нүүр.	8
3.2	Hoome - Cloud Car Parking нүүр.	8
4.1	BLoC	10
5.1	Хамгийн их пост хийсэн хэрэглэгчдийн dashboard	26
5.2	Хамгийн их пост хийсэн хэрэглэгчдийн dashboard	26
8.1	Зураг голлуулах	31
8.2	Зургийн нэрийг энд бичнэ	32

ХҮСНЭГТИЙН ЖАГСААЛТ

1.1	Үйлдвэрлэлийн дадлагын төлөвлөгөө	2
9.1	Хүснэгтийн нэр. Хүснэгтийн нэр хүснэгтийн дээд талд байрлана.	33

Кодын жагсаалт

4.1	BLoC компонентийн машины дугаар бүртгэлийн хэрэгжүүлэлт	10
4.2	Машины дугаарыг хэрэглэгчийн дугаарын хамтаар сервер дээр бүртгэх	12
4.3	Keycloak realm-тай холбох Yaml тохиргоо	14
4.4	myErd.jdl дотор буй ERD	15
4.5	UserDTO.java	17
4.6	UserDTO.java	19
4.7	UserRepository.java үүсгэсэн байдал	21
5.1	Materialized view үүсгэн уншилт хийж буй байдал	22
5.2	Keycloak-аас хэрэглэгчийн мэдээллийг авч буй байдал	24
5.3	Retryable-ийг хэрэгжүүлсэн байдал	25
10.1	С хэлний кодын жишээ	34
10.2	Кодын файлаас хэсэгчилж оруулах	34

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 интерфэйсийн 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	Noome мобайл аппликейшн дээр зогсоолын төлбөр төлөх хэсэг дээр bug засах, карт холбох feature нэмэх	4 өдөр	Гүйцэгтэсэн	10/10
6	Noome 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	”Noome” 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

2.3. ТЕХНОЛОГИУД БОЛОН СИСТЕМҮҮД БҮЛЭГ 2. БАЙГУУЛЛАГЫН ТУХАЙ

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

3. НООМЕ ТӨСӨЛ

Үйлдвэрлэлийн дадлагын хүрээнд одоо идэвхитэй явагдаж буй "Нооме" төсөл дээр ажилласан бөгөөд энэхүү төсөл нь Нооме мобайл аппликейшн, Нооме сөх веб, Нооме 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 "Нооме" мобайл аппликейшнд feature байдлаар оруулж өгсөн байгаа.

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

3.1 Хотхон

"Нооме" платформын хотхоны систем нь СӨХ-ийн менежмент, төлбөр, оршин суугчдын бүх төрлийн харилцааг удирдах тусгай систем болон түүнийг иргэдэд хүргэх "Нооме" 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

”Нооме” 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

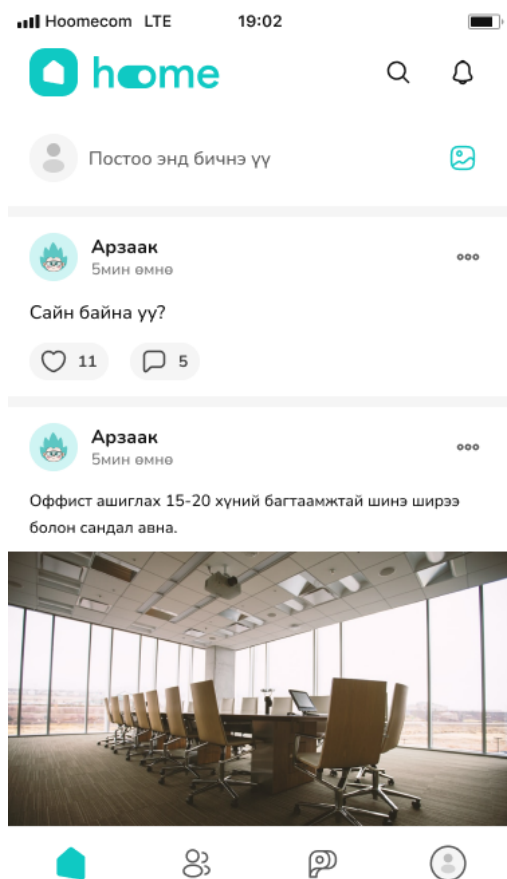
зэрэг функцуудээс бүрдэнэ.

3.3 Мобайл аппликейшн

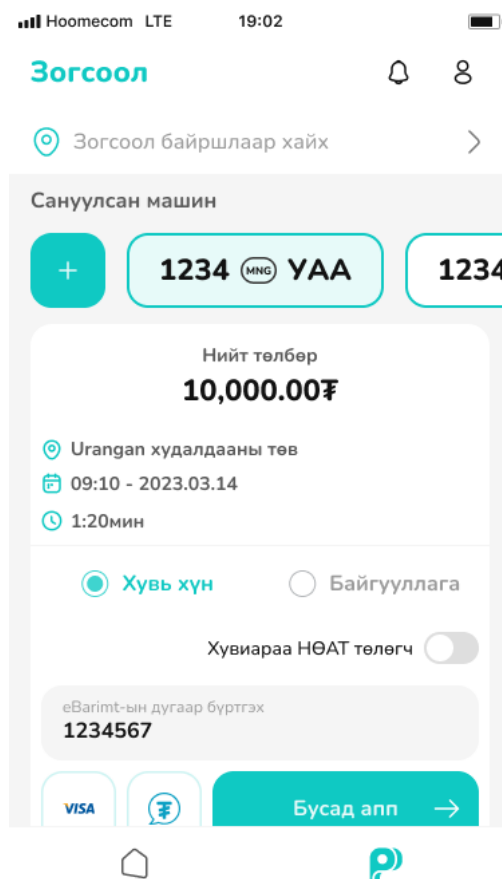
”Нооме” платформын мобайл аппликейшн нь СӨХ-ийн төлбөр төлөгч болон Нооме-д бүртгэлтэй зогсоолд машинаа тавьсан хэрэглэгчдэд зориулагдсан. Хэрэглэгч өөрийн хотхоны СӨХ-д элссэнээр сар бүрийн төлөх зардлууд болон тооцоог нэг дороос харах боломжтой болно. Зогсоолын төлбөр буюу ”Cloud Car Parking” систем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

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

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



Зураг 3.1: Нооме - Коммунити нүүр.



Зураг 3.2: Нооме - 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.

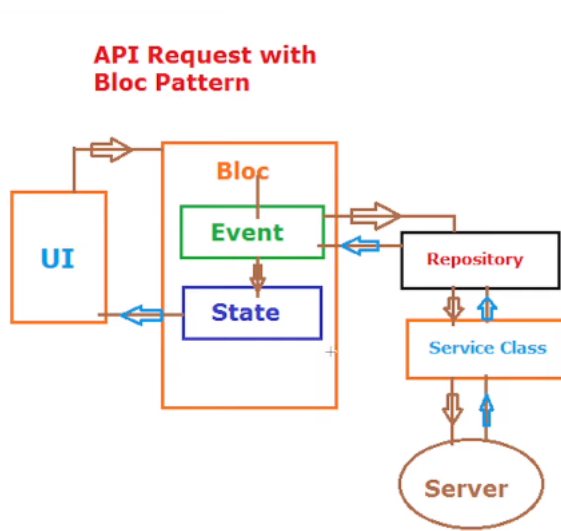
BLoC нь хэрэглэгчийн интерфэйсийг бизнес логигоос тусгаарлаж өгөх зорилготой сан бөгөөд event-driven архитектур дээр суурилсан байдаг.

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 интерфэйст 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 логигоос хамааран онцгой тохиолдол(exception), алдаа(runtime error)-наас сэргийлэх боломжтой. Аппликейшний 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

```

```
5 part 'parking_event.dart';
6 part 'parking_state.dart';
7 class ParkingBloc extends Bloc<ParkingEvent, ParkingState> {
8   //... Other event handlers
9
10  void _onSavePlateNumber(
11    RegisterPlateNumber event,
12    Emitter<ParkingState> emit,
13  ) async {
14    emit(state.copyWith(registerStatus: RegisterStatus.loading));
15    bool isRegistered = await ParkingRepository().registerParking(event
16      .phoneNumber, event.plateNumber);
17    if (isRegistered) {
18      LocalStorage().save('phoneNumber', event.phoneNumber, const
19        Duration(days: 365));
20      emit(state.copyWith(localPhoneNumber: event.phoneNumber));
21    }
22    emit(state.copyWith(
23      registerStatus: isRegistered ? RegisterStatus.success :
24        RegisterStatus.failure,
25      statusMessage: 'Car_registration_success!',
26    ));
27  }
28 }
```

Код 4.1: BLoC компонентийн машины дугаар бүртгэлийн хэрэгжүүлэлт

```
1 import 'package:dio/dio.dart';
2
3 class ParkingRepository {
4   final Dio dio = Dio();
5
6   //... Other repository methods
7
8   Future<bool> registerParking(String phoneNumber, String plateNumber)
9     async {
10     final body = {
11       'phoneNumber': phoneNumber,
12       'plateNumber': plateNumber,
13     };
14
15     try {
16       final response = await dio.post(ApiConstants.registerParkingUser,
17         data: jsonEncode(body));
18       return response.statusCode == 200 || response.statusCode == 204;
19     } on DioError catch (_) {
20       return false;
21     }
22   }
23 }
```

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.

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 логикийг 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 командыг ашиглан эхний байдлаар микросервисийг үүсгэнэ.

```
1 jhipster

1 spring:
2 security:
3   oauth2:
4     client:
5       provider:
6         oidc:
7           issuer-uri: http://your-keycloak-server/auth/realms/your-
              realm
8       registration:
9         oidc:
10           client-id: your-client-id
11           client-secret: your-client-secret
12 resource:
13   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 команд

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

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

```
1 ./mvnw
```

```
1 entity Person {
2     keycloakId String,
3     username String,
4     firstname String,
5     lastname String,
6     fullname String,
7     balance Integer,
8     referralCode String unique,
9     createdBy String,
10    createdAt Instant,
11    lastModifiedBy String,
12    lastModifiedDate Instant,
13 }
14
15 entity Referral {
16     keycloakId String,
17     username String,
18     firstname String,
19     lastname String,
20     fullname String,
21     createdBy String,
22     createdAt Instant,
23 }
24
25 entity PointTrans {
26     amount Integer,
27     transEnum TransEnum,
```

```
28     description String,
29     serviceEnum ServiceEnum,
30     createdBy String,
31     createdAt Instant,
32     lastModifiedBy String,
33     lastModifiedDate Instant,
34 }
35
36 enum TransEnum {
37     INCOME, EXPENSE
38 }
39
40 enum ServiceEnum {
41     CCP, SOH, ITSTORE, SYSTEM
42 }
43
44 relationship OneToMany {
45     Person to Referral{person required},
46     Person to PointTrans{person required},
47 }
48
49 skipClient *
50 paginate * with infinite-scroll
51 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/
14        etc
15        this.login = user.getLogin();
16    }
17    public String getId() {
18        return id;
19    }
20    public void setId(String id) {
21        this.id = id;
22    }
23    public String getLogin() {
24        return login;
25    }
26    public void setLogin(String login) {
27        this.login = login;
28    }
29 }
```

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

Код 4.5: UserDTO.java

```
1 package mn.nomadicss.web.rest;
2 ...
3 @RestController
4 @RequestMapping("/api")
5 public class PersonResource {
6     private final Logger log = LoggerFactory.getLogger(PersonResource.
7         class);
8
9     private static final String ENTITY_NAME = "hoomepointPerson";
10
11     @Value("${jhipster.clientApp.name}")
12     private String applicationName;
13     private final PersonService personService;
14     private final PersonRepository personRepository;
15     private final PersonQueryService personQueryService;
16
17     public PersonResource(PersonService personService, PersonRepository
18         personRepository, PersonQueryService personQueryService) {
19         this.personService = personService;
20         this.personRepository = personRepository;
21         this.personQueryService = personQueryService;
22     }
23
24     @PostMapping("/people")
25     ...
26
27     @PutMapping("/people/{id}")
28     ...
29
30     @PatchMapping(value = "/people/{id}", consumes = { "application/"
```

```
        json", "application/merge-patch+json" })  
26    ...  
27    @GetMapping("/people")  
28    ...  
29    //... More controllers for data access  
30 }
```

Код 4.6: UserDTO.java

```
1 package mn.nomadicss.repository;
2
3 import java.util.List;
4 import java.util.Optional;
5 import mn.nomadicss.domain.User;
6 import org.springframework.data.domain.*;
7 import org.springframework.data.jpa.repository.EntityGraph;
8 import org.springframework.data.jpa.repository.JpaRepository;
9 import org.springframework.stereotype.Repository;
10
11 /**
12  * Spring Data JPA repository for the {@link User} entity.
13  */
14 @Repository
15 public interface UserRepository extends JpaRepository<User, String> {
16     Optional<User> findOneByLogin(String login);
17
18     @EntityGraph(attributePaths = "authorities")
19     Optional<User> findOneWithAuthoritiesByLogin(String login);
20
21     Page<User> findAllByIdNotNullAndActivatedIsTrue(Pageable pageable);
22 }
```

Код 4.7: UserRepository.java үүсгэсэн байдал

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

5. АСУУДАЛ БА ШИЙДЭЛ

Дадлагын хүрээнд "Noome" 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**

```
1 SimpleStatement statement = SimpleStatement
2   .builder(
3       "create_materialized_view_if_not_exists_" +
4       keyspaceName +
5       ".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

```
1 List<Map.Entry<String, Long>> mostLikedUsers = getMostLikedUsers();
2 mostLikedUsers.sort(Map.Entry.comparingByValue(Comparator.reverseOrder
3     ()));
4
5 int gaugeBuiltCount = 0;
6 for (Map.Entry<String, Long> entry : mostLikedUsers) {
7     if (gaugeBuiltCount >= mostLikedUsersLimit) {
8         break;
9     }
10    try {
11        Gauge
12            .builder("homebook.mostLikedUsers", () -> entry.getValue())
13            .tag("user_name", getUsernameFromUuid(entry.getKey()))
14            .register(registry);
15        gaugeBuiltCount++;
16    } catch (NotFoundException nfe) {
17        log.error("userId: {} not found in keycloak", entry.getKey());
18    }
19 }
20 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.

```
1 @KafkaListener(topics = "#{${kafka.image-topic.topic-name}}", groupId
   = "#{${kafka.image-topic.topic-group-id}}")
2 @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}}"))
3 public void consumeMessage(String imageTopicJson) throws Exception {
4     // ... Logics
5     imageTopic = imageService.process(imageTopic);
6     cloudClientService.setObjectStorage(imageTopic);
7 }
8 @Recover
9 public void recoverAfterRetryFail(Exception ex, String imageTopicJson
   ) {
10     // ... Logics
11     try {
12         String exceptionMessage = objectMapper.writeValueAsString(
            messageMap);
13         config.kafkaTemplate().send(dltTopic, exceptionMessage);
14     } 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. УРАМШУУЛЫН СИСТЕМ

7. ИШЛЭЛ, ЗҮҮЛТ

7.1 Ишлэл

Ашигласан материал эсвэл номзүйг бичвэр тодор ишлэхдээ `cite` командаар заалтыг нь оруулна. Үүний тулд энэ хуудасны хамгийн доор байгаа *Ашигласан материал, ном зүй хэсэгт* `bibitem` командыг нэмнэ.

Жишээ нь: `bibitem{image1}` Гарчиг, Зохиогчдын нэр, хэвлэсэн он, хэвлэсэн газар

Дээрх жишээнд `image1` гэдэг нь ишлэх нэр. Доод талын мөрөнд нь байгаа дарааллын дагуу ашигласан материалыг бичнэ.

Ишлэхдээ `cite` командад ишлэх нэрийг дамжуулж өгнө. Жишээ нь `cite{image1}`.

7.2 Зүүлт

Зүүлтийг `footnote` командаар оруулна ¹.

¹Энэ холбоосоос зүүлтийн талаар дэлгэрэнгүй унш: <https://www.sharelatex.com/learn/Footnotes>

8. ЗУРАГ

Зураг оруулахдаа `includegraphics` командыг ашиглана. Доорх жишээнд `figure01.png` гэдэг нь зургийн файлын нэр бөгөөд өргөтгөлийг заавал бичих шаардлагагүй. Зургийн файл нь `main.tex` файлтай нэг фолдерт байх шаардлагатайг анхаарна уу! Дэлгэрэнгүйг [1]-с үз.

ShareLaTeX



Learn

8.1 Зургийн хэмжээ өөрчлөх

Хэмжээг томруулахдаа 0-1 хооронд утга ашиглана. Хэрэв 2 гэвэл 2 дахин томроно.

```
includegraphics[scale=0.5]{figure01}
```

ShareLaTeX



Learn

Өндөр өргөнийг шууд зааж өгч болох бөгөөд дөрвөлжин хаалтан дотор доорх байдлаар бичнэ.

```
includegraphics[width=3cm, height=4cm]{figure01}
```

ShareLaTeX



Learn

8.2 Зураг эргүүлэх

Зургийн эргүүлэхдээ `angle` параметрт эргүүлэх өнцгийн хэмжээг өгнө.

```
includegraphics[width=3cm, height=4cm, angle=45]{figure01}
```



8.3 Зургийн нэр

Зургын нэрийг `begin{figure}` хооронд `includegraphics` командтай хамт оруулна Зураг 8.1-ыг хар.

Энд зургийн нэрээс гадна `label`-ийг давхар бичиж өгөх шаардлагатай ба энэ нь зургийн дугаараар заалт хийхэд ашиглана. Жишээ нь: Зураг 8.2



Зураг 8.1: Зураг голлуулах

8.4 Зураг голлуулах

Зургийг голлуулахдаа `includegraphics` командын өмнө `centering` командыг бичээд `reflectbox` командыг `includegraphics` болон `caption` командуудад үйлчлэхээр оруулна.



Зураг 8.2: Зургийн нэрийг энд бичнэ

8.5 Зургийн чанар

LaTeX-т зургийг вектор форматаар (`svg`, `eps`) оруулбал хэвлэх болон томруулж харахад зургийн чанар алдагдахгүй. Тиймээс аль болох вектор зураг оруулж өгвөл зүгээр.

9. ХҮСНЭГТ ОРУУЛАХ

Хүснэгт оруулахад `tabular` командыг ашигладаг [5].

Table 9.1: Хүснэгтийн нэр. Хүснэгтийн нэр хүснэгтийн дээд талд байрлана.

Багана1	Багана2	Багана3	Багана4	Багана5
өгөгдөл	<i>өгөгдөл1</i>			

9.1 Хүснэгт зурах хэрэгсэл

Цэвэр LaTeX кодоор Хүснэгт үүсгэхэд харьцангуй төвөгтэй байдаг учир хялбар хэрэгслийг ашиглаж болно.

Тухайлбал <https://www.tablesgenerator.com/> холбоосруу орж хүснэгтийг визуал орчинд зураад үүсгэж өгсөн LaTeX кодыг энд хуулж оруулна.

10. КОД БА АЛГОРИТМ ОРУУЛАХ

Код оруулахдаа `begin{lstlisting} ... end{lstlisting}` командын хооронд бичнэ.

```
1 #include <stdio.h>
2 #define N 10
3 /* Block
4  * comment */
5 int main()
6 {
7     int i;
8     // Line comment.
9     puts("Hello_world!");
10    for (i = 0; i < N; i++)
11    {
12        puts("LaTeX_is_also_great_for_programmers!");
13    }
14    return 0;
15 }
```

Код 10.1: C хэлний кодын жишээ

Мөн кодын эх файлыг шууд оруулж ирж болох бөгөөд доорх командыг бичнэ.

```
1     puts("Hello_world!");
2
3     for (i = 0; i < N; i++)
4     {
5         puts("LaTeX_is_also_great_for_programmers!");
6     }
```

Код 10.2: Кодын файлаас хэсэгчилж оруулах

Мэдээллийн технологи, програм хангамжийн ажлын тайланд алгоритмыг хийсвэр кодын бичиглэлээр оруулах шаардлага гардаг. Дараах жишээгээр (Алгоритм 1) хийсвэр кодоор хэрхэн бичиж болохыг харуулав. Мөн бичвэр дотроо алгоритмд ашиглаж байгаа *parentId* хувьсагчийг дурдаж бичиж болдог.

Алгоритм 1 Даалгавар үүсгэх алгоритм

```

1: function TRAVERSE(parentId)                                ▷ parentId–эцэг ойлголтын дугаар
2:   children ← GETCHILDCONCEPTIDS(parentId)
3:   childCount ← children.count
4:   if childCount == 0 then
5:     return
6:   end if
7:   for i = 0 to childCount do
8:     GENERATE TASK(childreni)                                ▷ Орчуулгын даалгавар үүсгэх
9:   end for
10:  for i = 0 to childCount do
11:    TRAVERSE(childreni)
12:  end for
13: end function

```

Дүгнэлт

Дүгнэлтийг энд бич

Bibliography

- [1] Inserting Images, Share LaTeX, https://www.sharelatex.com/learn/Inserting_Images
- [2] Paragraphs and new lines, Share LaTeX, https://www.sharelatex.com/learn/Paragraphs_and_new_lines
- [3] Bold, italics and underlining, Share LaTeX, https://www.sharelatex.com/learn/Bold,_italics_and_underlining
- [4] Lists, Share LaTeX, <https://www.sharelatex.com/learn/Lists>
- [5] Tables, Share LaTeX, <https://www.sharelatex.com/learn/Tables>

А. ШИНЖИЛГЭЭ ЗОХИОМЖ

Хавсралтын агуулга

В. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

```
1 import numpy as np
2
3 def incmatrix(genl1,genl2):
4     m = len(genl1)
5     n = len(genl2)
6     M = None #to become the incidence matrix
7     VT = np.zeros((n*m,1), int) #dummy variable
8
9     #compute the bitwise xor matrix
10    M1 = bitxormatrix(genl1)
11    M2 = np.triu(bitxormatrix(genl2),1)
12
13    for i in range(m-1):
14        for j in range(i+1, m):
15            [r,c] = np.where(M2 == M1[i,j])
16            for k in range(len(r)):
17                VT[(i)*n + r[k]] = 1;
18                VT[(i)*n + c[k]] = 1;
19                VT[(j)*n + r[k]] = 1;
20                VT[(j)*n + c[k]] = 1;
21
22            if M is None:
23                M = np.copy(VT)
24            else:
25                M = np.concatenate((M, VT), 1)
26
27            VT = np.zeros((n*m,1), int)
28
29    return M
```