

**МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ**  
**МЭДЭЭЛЛИЙН ТЕХНОЛОГИЙН ЭЛЕКТРОНИКИЙН СУРГУУЛЬ**  
**МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ**

Баянжаргалын Энх-Амгалан

**Вэб хөгжүүлэлт**  
**(Web developing)**

Мэдээллийн технологи (D061303)  
Үйлдвэрлэлийн дадлагын тайлан

Улаанбаатар

2025 оны 01 сар

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

Вэб хөгжүүлэлт  
(Web developing)

Мэдээллийн технологи (D061303)  
Үйлдвэрлэлийн дадлагын тайлан

Удирдагч:	_____	Др. Ч.Алтангэрэл
Хамтран удирдагч:	_____	Н.Од-Эрдэнэ
Гүйцэтгэсэн:	_____	Б.Энх-Амгалан (21B1NUM0344)

Улаанбаатар  
2025 оны 01сар

# Зохиогчийн баталгаа

Миний бие Баянжаргалын Энх-Амгалан "Вэб хөгжүүлэлт" 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

УДИРТГАЛ .....	1
БҮЛГҮҮД .....	3
1. БАЙГУУЛЛАГЫН ТАНИЛЦУУЛГА .....	3
1.1 Товч Танилцуулга .....	3
1.2 Ямар Үйлчилгээ Үзүүлдэг Вэ? .....	3
1.3 Ямар Систем Дээр Голчлон Төвлөрдөг Вэ? .....	3
2. СИСТЕМИЙН ШААРДЛАГА .....	5
2.1 Хэрэглэгчийн шаардлага .....	5
2.2 Use Case диаграм .....	8
3. АШИГЛАХ ТЕХНОЛОГИ .....	9
3.1 Gitlab .....	9
3.2 NextJS .....	9
3.3 Golang .....	9
3.4 PostgreSQL .....	10
3.5 Postman .....	11
3.6 Docker .....	11
4. ХЭРЭГЖҮҮЛЭЛТ .....	12
4.1 GORM хэрэгжүүлэлт .....	12
4.2 Toast компонент .....	21
4.3 UI автоматжуулсан тест .....	28
5. ДҮГНЭЛТ .....	33
5.1 Үр дүн .....	33
5.2 Үр дүнгийн тайлан .....	36

НОМ ЗҮЙ .....	37
ХАВСРАЛТ.....	38
А. УДИРДАГЧИЙН ҮНЭЛГЭЭ .....	38
В. КОДЫН ХЭРЭГЖҮҮЛЭЛТ.....	39
В.1   Форм .....	39
В.2   Toast компонент.....	44
В.3   Toast компонент дээр интерфэйсийн автоматжуулсан тест бичсэн байдал.....	49

## ЗУРГИЙН ЖАГСААЛТ

2.1	Use Case диаграм	8
4.1	Формын эхний хувилбар	16
4.2	Формын эцсийн байдлаар харагдаж буй байдал	20
5.1	Формын эхний хувилбар	33
5.2	Material ui болон React Select ашигласан байдал	34
5.3	Хүсэлт амжилттай болсон үед харагдах Toast	35
5.4	Хүсэлт амжилтгүй болсон үед харагдах Toast	35

## ХҮСНЭГТИЙН ЖАГСААЛТ

1	Төлөвлөгөө гаргасан байдал .....	2
---	----------------------------------	---

# Кодын жагсаалт

4.1	Next.js дээр бичсэн серверээс датагаа татаж авах . . . . .	13
4.2	Component-н үндсэн логик үйлдлүүд . . . . .	13
4.3	Дотоод төлвийн эхний хувилбар . . . . .	15
4.4	Дотоод төлвийн сайжруулсан хувилбар . . . . .	15
4.5	Хийх үйлдлүүдийн төрлийг зарлах . . . . .	17
4.6	Хийх үйлдлүүдийг тодорхойлох . . . . .	17
4.7	Хүү сонголтуудыг цэвэрлэх . . . . .	18
4.8	React-Select ашигласан байдал . . . . .	18
4.9	Context үүсгэх . . . . .	22
4.10	Toast context-н анхны утгыг зарласан байдал . . . . .	22
4.11	Toast-н интерфэйс тодорхойлох . . . . .	23
4.12	Toast үүсгэх функц . . . . .	23
4.13	Toast-г context-оос цэвэрлэх . . . . .	24
4.14	Uniq ID гаргах . . . . .	24
4.15	useReducer дээр ажиллах үйлдлүүдийг заах . . . . .	24
4.16	Context-г буруу ашигласан үед алдаа гаргах . . . . .	25
4.17	Toast үүсгэх үүрэгтэй hook . . . . .	26
4.18	Material-UI ашиглаж интерфэйсийг үүсгэх . . . . .	26
4.19	Toast-г дуудаж ашиглаж буй байдал . . . . .	27
4.20	FakeTimer ашиглах . . . . .	28
4.21	Toast компонентийг DOM дээр зурна . . . . .	28
4.22	DOM дээр зурагдсан эсэхийг шалгана . . . . .	28
4.23	Хаах товчлуур дээр дарахад утсан эсэхийг шалгах . . . . .	28
4.24	Олон Toast зэрэг гаргаж дурын хугацааг зааж өгөх . . . . .	29
4.25	Зааж өгсөн хугацааны дараа цэвэрлэгдэж байгаа эсэхийг шалгах . . . . .	30
4.26	Өгсөн type-н дагуу зурагдаж байгаа эсэхийг шалгах . . . . .	31



## **УДИРТГАЛ**

Миний бие Б.Энх-Амгалан нь үйлдвэрлэлийн дадлагын хугацаанд Next.js, Golang гэсэн технологиуд дээр голчлон ажилласан ба уг технологиуд ямар шалтгаанаар үүссэн, цаана нь технологийн ямар дэвшил, хөгжүүлэлтийн арга барил ашигладаг, компаниуд хэрхэн үүн дээр хөгжүүлэлт хийж эцсийн бүтээгдэхүүнийг гаргадаг талаар судлахын тулд front-end хөгжүүлэлт дээрээ Next.js фрэймворк болон back-end хөгжүүлэлт дээрээ Golang ашигладаг компани болох "ОНДО" ХХК-г сонгон авч мэргэжлийн дадлагаа гүйцэтгэлээ.

**Зорилго** Next.js болон Golang технологийн талаар судалж, компанийн хөгжүүлэлтийн арга барилтай танилцах

**Зорилт** Удирдагчийн зааварчилгааны дагуу алхам алхмаар судалгаа хийж өгсөн шаардлагын хүрээнд хэрэгжүүлэлт хийх

Table 1: Төлөвлөгөө гаргасан байдал

№	Гүйцэтгэх ажил	Хугацаа	Биелэлт	Үнэлгээ
1	<i>SQL-ийг ORM-руу шилжүүлсэн</i>	<i>4 хоног</i>		
2	<i>Unite Test</i>	<i>3 хоног</i>		
3	<i>Admin Front-End хөгжүүлэлт</i>	<i>7 хоног</i>		
4	<i>Admin Back-End хөгжүүлэлт</i>	<i>7 хоног</i>		

Баталсан: Албан байгууллагын дадлага удирдагч:.....

Төлөвлөгөө боловсруулсан: Оюутан .....

# 1. БАЙГУУЛЛАГЫН ТАНИЛЦУУЛГА

ONDO бол цаг үедээ шинэ давлагаа. Монголын хэрэглэгчдэд үйлчилгээ үзүүлж ашиг олох гэхээсээ илүүтэй Монголд хийсэн технологио дэлхийн зах зээлд хүргэхээр ажиллаж буй компани юм. Stakeholder economy буюу төсөлд хамааралтай бүх хүн өгөөжийг хүртдэг эдийн засгийн тогтолцоог бий болгохоор зорьж, ёс зүйтэй, нийгмийн бүх хэсгүүдэд хүрч үйлчлэх тэр дундаа интернэтийг шаардлагатай бүлгийн иргэдэд таатай үнээр хүргэхээр ажилладаг. Нийт 120 залуусаас ONDO гэр бүл бүрддэг. Зах зээлд нэвтрээд 2 жилийн хугацаанд бид 385 мянган хэрэглэгчидтэй болжээ.

## 1.1 Товч Танилцуулга

2018 онд анх ONDO компани үүсгэн байгуулагдсан. Энэ үедээ ONDO нэрээ авч амжаагүй байсан бөгөөд үйл ажиллагаагаа албан ёсоор 2021 онд явуулж эхлэсэн. Хамгийн анхны бүтээгдэхүүнээ 2022 оны 3 сард зах зээлд нэвтрүүлсэн бөгөөд интернэтийн үйлчилгээг ХУГАЦААГҮЙ болон ХЯЗГААРГҮЙ байх нөхцлийг хэрэглэгчиддээ олгосон. Цаашдаа дэлхийн зах зээлд гарахаар зорин ажиллаж байна.

## 1.2 Ямар Үйлчилгээ Үзүүлдэг Вэ?

ONDO нь шинэ үеийн үүрэн холбооны үйлчилгээ үзүүлэгч компани. Тус компани нь хэрэглэгчдэдээ дэвшилтэт технологи, хурдан сүлжээ, мобайл дата, уян хатан үйлчилгээний багцуудыг санал болгодог. Хэрэглэгчид хангах заавал салбар оролгүй хүссэн газраасаа үйлчилгээ авах боломжыг бүрдүүлдэг.

## 1.3 Ямар Систем Дээр Голчлон Төвлөрдөг Вэ?

ONDO нь дижитал платформд суурилсан, хэрэглэгч төвтэй системүүд дээр ажилладаг

### 1.3. ЯМАР СИСТЕМ ДЭЭР ГОЛЧЛОШ ТӨЛӨРДӨЙ БЭХ ХУЛЛАГЫН ТАНИЛЦУУЛГА

бөгөөд хэрэглэгчдэд хурдан, найдвартай үйлчилгээ үзүүлэхэд анхаардаг.

## 2. СИСТЕМИЙН ШААРДЛАГА

Миний хувьд дадлагын 21 хоногийн хугацаа дахь сүүлийн долоо хоногт уг системийн мерчант вэб хэсэг дээр ажиллаж эхэлсэн ба гүйцэт биш ч тодорхой хэмжээнд системийн талаар судалгаа хийснээ хүргэж байна. Мөн уг систем дээр ажиллахын тулд нууцлалын гэрээ хийсэн учир дэлгэрэнгүй тайлбарлах боломжгүй.

Online аппыг бодит амьдрал дээрх олон дэлгүүр байршдаг нэг барилга гэж үзэж болох ба та тус апп руу орсноор яг л нэг молл дотор дэлгүүр хэсэж байгаа юм шиг хамгийн түрүүнд Online дээр гэрээтэй дэлгүүрүүдийн жагсаалт харагдана. Уг байдлаараа монголд хөгжиж буй бусад e-commerce-уудаас ялгаатай билээ. Харин миний ажилласан мерчант системийн гол үүрэг нь тухайн апп доторх дэлгүүрийг дэлгүүрийн эзэн буюу мерчант админ бүрэн удирддаг байх шаардлагатай.

### 2.1 Хэрэглэгчийн шаардлага

#### 2.1.1 Хэрэглэгчид

- Мерчант админ гэсэн ердөө ганц хэрэглэгчээс бүрдэнэ.

#### 2.1.2 Функционал хэрэглэгчийн шаардлагууд

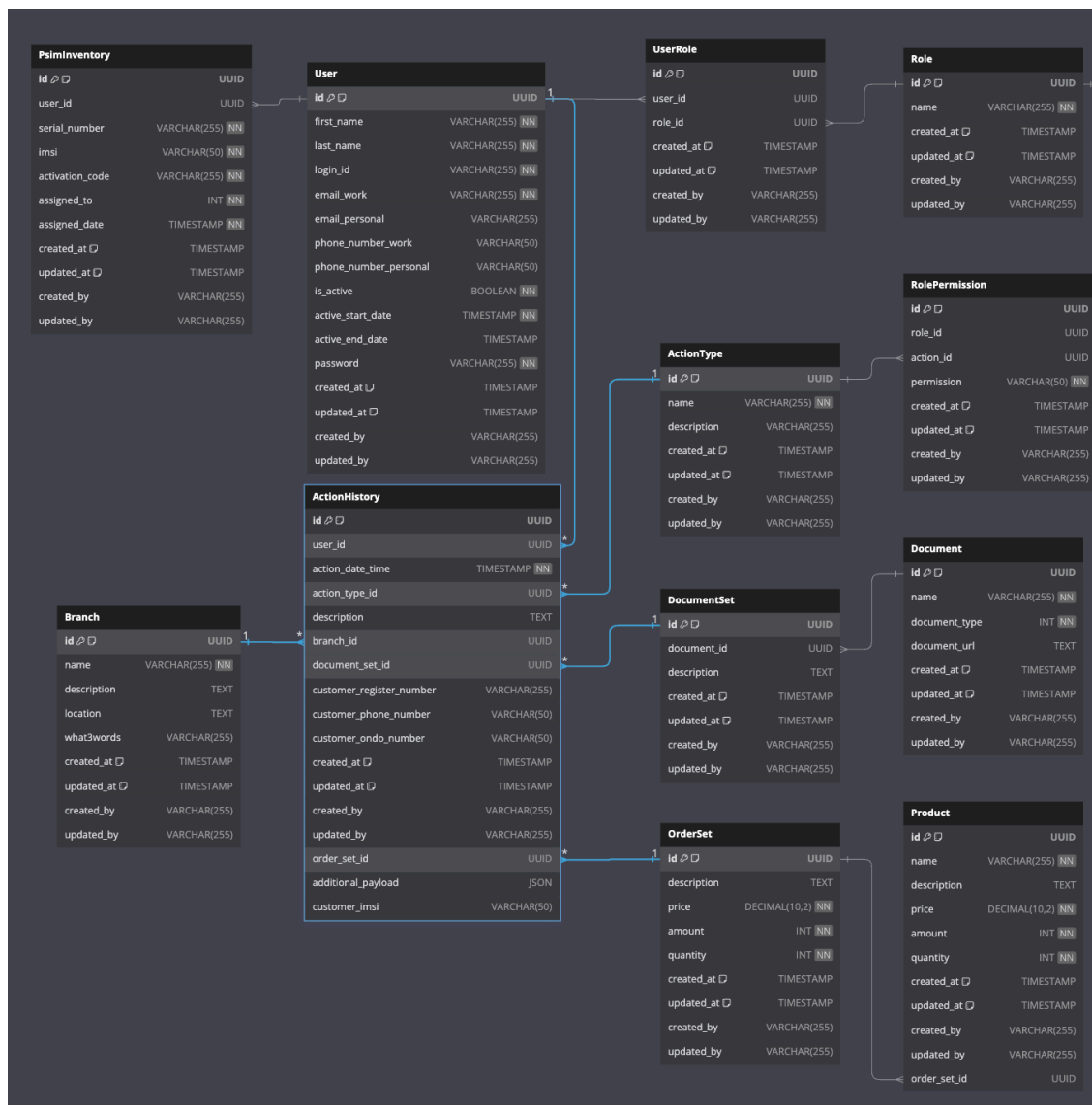
- Хэрэглэгч зөвхөн имэйлээр ирсэн холбоосоор нэвтэрдэг байх
- Хэрэглэгч өөрийн хаяг дээрээ олон дэлгүүр үүсгэх боломжтой байх
- Дэлгүүр бүр ажиллах цагийн хуваарьтай байх
- Дэлгүүр бүр cover болон icon зурагтай байх
- Дэлгүүр бүр бүтээгдэхүүний ангилалтай байх ёстой
- Бүтээгдэхүүний цуглуулга буюу коллекцийг үүсгэдэг байх

- Коллекц заавал cover болон icon зураг авдаг байх
- Хямдрал зарлах боломжтой байх
- Хямдралыг төрлөөр нь хувь, үнэ гэсэн хоёр хуваадаг байх
- Бүх бүтээгдэхүүн дээр хямдрал зарлах боломжтой байх
- Зөвхөн коллекци дээр хямдрал зарлах боломжтой байх
- Хэрэглэгчийн сервертэй харьцсан бүх үйлдэлд Toast-р хариу өгдөг байх
- Бүтээгдэхүүн үүсгэж болдог байх
- Бүтээгдэхүүн заавал нэг буюу түүнээс дээш зурагтай байх
- Бүтээгдэхүүний оруулж буй бүх зураг 1:1 хэмжээтэй байх
- Бүтээгдэхүүн заавал үнэтэй байх
- Мерчант дээр хуулагдаж буй бүх зураг alt таг авдаг байх
- Үүсгэсэн бүтээгдэхүүнээ засдаг байх
- Нүүр хэсэгт тухайн сонгосон дэлгүүрийн борлуулалтын статистикуыг харуулдаг байх
- Нүүр хэсэгт тухайн сонгосон дэлгүүр дээрх барааны жагсаалт харагддаг байх

### 2.1.3 Функционал бус шаардлага

- Вэбсайт нь хэрэглэхэд хялбар байх
- Бүх хуудасны арын суурь өнгө ижил байх
- Responsive дизайнтай байх
- Бүтээгдэхүүн үүсгэх хэсэг хялбар байх

- Цагийн хуваарь болон дэлгүүрийн бусад мэдээлэл оруулахад ойлгомжтой байх
- Бүх хуудас тусламж авах хэсэгтэй байх



Зураг 2.1: Use Case диаграм

## 2.2 Use Case диаграм

Мерчант вэб болон Oneline дээр хийгдсэн Use Case диаграм



## 3. АШИГЛАХ ТЕХНОЛОГИ

### 3.1 Gitlab

GitLab нь Git дээр суурилсан DevOps платформ бөгөөд програм хангамж хөгжүүлэх, тест хийх, нэвтрүүлэх процессыг автоматжуулан нэг дор удирдах боломжийг олгодог. GitLab нь CI/CD (Continuous Integration/Continuous Deployment) багтаасан байдаг ба энэ нь хөгжүүлэгчид өөрсдийн кодыг турших, хянах, шинэчлэлтүүдийг хурдан бөгөөд найдвартай нэвтрүүлэхэд тусалдаг. Хөгжүүлэгчид өөрсдийн төслийг GitLab дээр байршуулж, хамтын ажиллагааг хөнгөвчилж, кодын чанарыг сайжруулах боломжтой байдаг.

### 3.2 NextJS

Next.js нь React-д суурилсан, сервер талын рендэрлэг SSR (Server-side rendering), статик сайт үүсгэгч (Static Site Generation), API маршрут боловсруулагч зэрэг олон төрлийн функцуудыг агуулсан хүчирхэг framework юм. Next.js нь хурдан, SEO (search engine optimization)-д ээлтэй вэб аппликейшнуудыг бүтээхэд туслах зорилготой. Энэ нь хуудасны динамик маршрут, API endpoint-ууд, автомат статик зэрэг олон төрлийн функцуудыг агуулдаг. Next.js нь хөгжүүлэгчидэд илүү хялбар бөгөөд хурдан хөгжүүлэлтийг хангахын зэрэгцээ, вэб аппликейшны гүйцэтгэлийг сайжруулдаг. Мөн Next.js нь TypeScript-ийг дэмжиж, хөгжүүлэгчидэд илүү найдвартай, уншигдахуйц код бичих боломжийг олгодог.

### 3.3 Golang

Golang буюу Go нь Google-ээс гаргасан, хурдан, найдвартай, энгийн синтакс бүхий програмчлалын хэл юм. Go нь системийн програмчлал, сүлжээний програмчлал, cloud computing, микросервис архитектур зэрэг олон төрлийн хэрэглээнд тохиромжтой. Go хэл нь статик төрөлтэй, garbage

collection-тэй, multi-threading-ийг хялбаршуулсан goroutine-уудтай байдаг. Энэ нь хөгжүүлэгчидэд өндөр гүйцэтгэлтэй, масштабдах боломжтой програмуудыг бүтээх боломжийг олгодог. Go нь мөн стандарт сангуудын багцтай байдаг бөгөөд энэ нь хөгжүүлэгчидэд нэмэлт сангуудгүйгээр олон төрлийн ажиллагааг хэрэгжүүлэх боломжийг олгодог.

Go хэл нь мөн GORM гэдэг объектын холбоосын (ORM) санг дэмждэг бөгөөд энэ нь Go хэл дээрх өгөгдлийн сантай ажиллах процессыг хялбаршуулдаг. GORM нь хөгжүүлэгчдэд SQL болон өгөгдлийн сан холбохыг хялбаршуулж, объектуудыг шууд өгөгдлийн сангийн хүснэгтүүдэ хамааруулан ажиллах боломжийг олгодог. Энэ нь CRUD (Create, Read Update, Delete) үйлдлүүдийг гүйцэтгэх, өгөгдлийн миграци, холбол хийх зэрэг үүргийг гүйцэтгэх боломжийг бүрдүүлдэг. GORM-ийн ашиглах нь Go-ийн синтаксийн энгийн байдал, хурдтай холбогдсон, мөн өгөгдлийн сангийн бүхий л үйлдлүүдийг энгийн, хурдан гүйцэтгэх боломжоор хангадаг.

### 3.4 PostgreSQL

PostgreSQL нь хүчирхэг, нээлттэй open source объект хандлагат database бөгөөд сүүлийн 30 гаруй жилийн турш идэвхтэй хөгжиж байгаа найдвартай ажиллагаатай мэдээллийн сангийн програм юм. 1986 онд Калифорни их сургуулийн “Postgres” төслийн нэг хэсэг болж хөгжсөн түүхтэй. PostgreSQL ихэнх үйлдлийн системүүд дээр ажилладаг ба PostGIS гэсэн орон зайн мэдээллийн сангийн нэмэлт хэрэгсэлтэйгээрээ бусад мэдээллийн сангийн програмуудаас давуу талтай. Мэдээллийн сан бол таны мэдээллийг найдвартай хадгалах үүрэгтэй. PostgreSQL мэдээллийн сан нь SQL хэлний дийлэнх стандартууд болон орчин үед хэрэглэгдэж байгаа ихэнх мэдээллийн сангийн цогц асуулгууд буюу queries, гадаад түлхүүр буюу foreign keys, триггер командууд, өөрчлөлт хийх боломжтой view, мэдээ дамжуулах хурд, гэх мэт боломжуудыг өөртөө агуулсан байдаг. Мөн хэрэглэгч нь мэдээний төрөл, функцууд, үйлдлүүд, индексийн арга зэргийг өөрийн хэрэгцээнд зориулан хөгжүүлэх боломжтой нээлттэй эх үүсвэрийн мэдээллийн сан юм. PostgreSQL системийн үндсэн архитектурыг харвал клиент / серверийн загварыг

ашигладаг.

### 3.5 Postman

Postman нь API хөгжүүлэлт, тест хийх, баримтжуулалт хийхэд ашиглагддаг хэрэгсэл юм. Энэ нь хөгжүүлэгчидэд API-уудыг хялбархан турших, өөрчлөлт оруулах, багийн гишүүдтэй хамтран ажиллах боломжийг олгодог. Postman нь REST, GraphQL, SOAP зэрэг олон төрлийн API-уудыг дэмждэг бөгөөд хөгжүүлэгчидэд API-ийн хүсэлт, хариуг хялбархан үүсгэх, илгээх, хариу үйлдлийг шалгах боломжийг олгодог. Мөн Postman нь автомат тест үүсгэх, тест сценари бичих, API-ийн гүйцэтгэлийг шалгах боломжийг олгодог. Энэ нь хөгжүүлэлтийн явцыг хурдасгах, API-ийн найдвартай байдлыг нэмэгдүүлэхэд тусалдаг.

### 3.6 Docker

Docker нь аппликейшнуудыг контейнер хэлбэрээр багцлах, ажиллуулах, түгээхэд ашиглагддаг платформ юм. Docker нь аппликейшны хөгжүүлэлт, тест хийх, production орчинд хүргэх явцыг хялбаршуулдаг. Контейнерчилал нь аппликейшны бүх шаардлагатай нөөцийг (код, санууд, системийн хэрэгслүүд) нэг дор багцлах боломжийг олгодог бөгөөд энэ нь аппликейшныг ямар ч орчинд ажиллуулах боломжийг олгодог. Docker нь хөгжүүлэгчидэд аппликейшны орчны тогтвортой байдлыг хангах, аппликейшныг хурдан, найдвартайгаар хүргэх боломжийг олгодог. Мөн Docker нь микросервис архитектурт тохиромжтой бөгөөд энэ нь орчин үеийн cloud-native аппликейшнуудыг хөгжүүлэхэд тусалдаг.

## 4. ХЭРЭГЖҮҮЛЭЛТ

### 4.1 GORM хэрэгжүүлэлт

Уг функцуудыг шаардлагын дагуу алхам алхмаар хэрэгжүүлсэнээр GORM-ийг ашиглах анхан шатны туршлагатай болох зорилготой байсан ба цаашид ажилласан төсөл дээр GORM-ийн дотоод логик, өгөгдлийн сангийн холболт, тайлбар хийх, өгөгдөл хадгалах болон шинэчлэх процессуудын ажиллагааг ойлгох, тодорхой хэмжээнд практик мэдлэгийг цуглуулж чадсан.

Сурах ур чадвар:

- GORM-ийг ашиглах чадвараа нэмэгдүүлэх
- Өгөгдлийн сантай харьцах үйлдлүүд (GET, POST, UPDATE, DELETE) болон тэдгээрийн процессийг ойлгох
- Хэрэглэгчийн логийг бүртгэх болон алдааг барих чадвар эзэмших
- CRUD үйлдлүүдийг GORM ашиглан бичих
- Горм-ийг ашиглахдаа data validation болон password hashing-ийн талаар мэдлэгтэй болох

Функцийн шаардлага:

- GORM ашиглан хэрэглэгчийн мэдээллийг авах, нэмэх, засах, устгах
- Хэрэглэгчийн нэвтрэх ID-г шалгах функц бичих
- Алдааг бүртгэх болон лог хийх функц ашиглах
- GORM-ийн дотоод функцууд болон query үйлдлийг ойлгох
- Хэрэглэгчийн мэдээлэл устгах үед хамааралтай мэдээллийг устгах

**4.1.1 useEffect болон useState ашиглан формын мэдээллийг шаардлагын дагуу авах**

Эхний ээлжинд ямар нэгэн загваргүй зөвхөн формын зөв ажиллагаа буюу логик үйлдлүүд дээр анхаарах хэрэгтэй байсан ба хамгийн түрүүнд хийх шаардлагатай зүйл нь аймаг сумдын датаг next.js дээрээ үүсгэсэн api-аасаа авч дэлгэцэнд харуулах байсан юм.

```
1 const fetchData = (url) => {  
2   return axios  
3     .get(`http://localhost:3000/api/${url}`)  
4     .then((res) => {  
5       const results = res.data;  
6       return results;  
7     })  
8     .catch((err) => {  
9       console.error(err);  
10    });  
11  };
```

Код 4.1: Next.js дээр бичсэн серверээс датагаа татаж авах

Доор харагдаж буй хэсэгт компонентийн логик үйлдлүүд харагдаж байна. useForm() hook ашиглаж select-н утга өөрчлөгдсөн эсэхийг барьж авах, functional component ашиглан state дотор хэрэглэгчийн сонгосон аймаг, сум, хороог хадгална.

```
1 export default function Home() {  
2   const [values, handleChange] = useForm();  
3   const [data, setData] = useState({  
4     cities: [],  
5     districts: [],  
6     wards: [],  
7   });
```

```
8
9  useEffect(() => {
10    fetchData(`cities`)
11      .then((res) => {
12        setData({ ...data, cities: res });
13      })
14      .catch((err) => {
15        console.error(err);
16      });
17  }, []);
18
19  const register = (e) => {
20    e.preventDefault();
21    console.log(values);
22  };
23
24  const handleSelect = (id, type) => {
25    if (type == "city") {
26      fetchData(`cities/${id}`)
27        .then((res) => {
28          setData({ ...data, districts: res, wards: [] }); //set
29            districts and clear wards data
30        })
31        .catch((err) => {
32          console.error(err);
33        });
34    } else if (type == "district") {
35      //get wards
```

```

35     fetchData(`${cities}/${values.city}/${id}`)
36     .then((res) => {
37         setData({ ...data, wards: res });
38     })
39     .catch((err) => {
40         console.error(err);
41     });
42 }
43 };
44
45 ...

```

Код 4.2: Component-н үндсэн логик үйлдлүүд

State дотор хэрэглэгчийн сонгосон мэдээллийг зөвөөр хадгалах шаардлагатай байсан. Эхний байдлаар доор харагдаж байгаагаар хадгалсан ч үүссэн асуудлууд нь замбаараагүй, эцэг сонголтыг сонгоход хүү сонголтуудыг цэвэрлэхэд төвөгтэй байв.

```

1  const [cities, setCities] = useState([]);
2  const [districts, setDistricts] = useState([]);
3  const [wards, setWards] = useState([]);
4  const [selectedCity, setSelectedCity] = useState();
5  const [selectedDistrict, setSelectedDistrict] = useState();
6  const [selectedWard, setSelectedWard] = useState();

```

Код 4.3: Дотоод төлвийн эхний хувилбар

Иймд state-ээ дараах байдлаар хадгаллаа.

```

1  const [data, setData] = useState({
2      cities: [],
3      districts: [],

```

Хот/Аймаг: Булган ▼  
Сум/Дүүрэг: Бугат ▼  
Баг/Хороо: 03 ▼  
Бүртгүүлэх

Зураг 4.1: Формын эхний хувилбар

```
4     wards: [],  
5   });
```

Код 4.4: Дотоод төлвийн сайжруулсан хувилбар

Эхний шаардлагын дагуу формыг хэрэгжүүлсний дараах вэб дээр харагдах байдал



### 4.1.2 *useState-н оронд useReducer hook ашиглах*

useReducer нь useState hook-н өөр нэг хувилбар бөгөөд өөр дээрээ state болон action гэсэн параметруудийг хүлээн авдаг. State нь өгөгдөл хадгалах бол action нь тухайн state дээр ямар үйлдлүүдийг хийх шаардлагатайг хүлээж авдаг.

Хамгийн түрүүнд Action-ынхаа төрлүүдийг зарлана.

```
1  const SET_CITY = "city";
2  const SET_DISTRICT = "district";
3  const SET_WARD = "ward";
```

Код 4.5: Хийх үйлдлүүдийн төрлийг зарлах

Үүний дараа хийх үйлдлүүдээ switch case дотор тодорхойлно. Switch case ашигласнаар олон дахин функц зарлах шаардлагагүйгээр Action-н төрлөөс хамаарч тухайн үйлдлийг ажиллуулна.

```
1  const reducer = (state, action) => {
2    switch (action.type) {
3      case SET_CITY:
4        return {
5          city: action.index,
6          district: null,
7          ward: null,
8        };
9      case SET_DISTRICT:
10       return {
11         ...state,
12         district: action.index,
13         ward: null,
14       };
15      case SET_WARD:
```

```

16     return {
17         ...state,
18         ward: action.index,
19     };
20     default:
21         return state;
22     }
23 };

```

Код 4.6: Хийх үйлдлүүдийг тодорхойлох

useReducer ашигласнаар өгсөн шаардлагуудын нэг болох эцэг сонголтыг солиход хүү сонголтууд хоосрох ёстой гэснийг маш хялбар байдлаар шийдэх боломжтой болов. Ердөө тухайн сонголтын доор байгаа state-үүдийн утгыг анхны утгаар солисон.

```

1     ...
2     case SET_DISTRICT:
3         return {
4             ...state,
5             district: action.index,
6             ward: null,
7         };
8     ...

```

Код 4.7: Хүү сонголтуудыг цэвэрлэх

Одоо форм дээрээ React-Select санг ашиглаж компонент доторх кодоо бичих шаардлагатай. Энэ хэсэг нь DOM дээр рендерлэгдэнэ.

```

1     ...
2     <form className={styles.grid} onSubmit={register}>
3         <label>

```

```
4      <p>Country:</p>
5      <Select
6          value={address.map((i, index) => ({ ...i, index }))[state.city
              ]}
7          onChange={(e) => handleChange(e.index, SET_CITY)}
8          options={address.map((i, index) => ({ ...i, index })))}
9          getOptionLabel={(option) => option.name}
10         getOptionValue={(option) => option.index}
11         placeholder="Choose country"
12     />
13 </label>
14 ...
```

Код 4.8: React-Select ашигласан байдал

Хот/Аймаг:

Улаанбаатар

Сум/Дүүрэг:

Дүүрэг 2

Баг/Хороо:

Сонгох

Бүртгүүлэх

Зураг 4.2: Формын эцсийн байдлаар харагдаж буй байдал

React-Select болон Material-UI ашигласны дараа формын маань харагдах байдал. Удирдагчийн өгсөн шаардлагуудын дагуу амжилттай хөгжүүлж дууслаа.

## 4.2 Toast компонент

Уг компонентийн гол үүрэг нь Мерчант төсөл дээр ашиглаж буй бүх хүсэлтүүдийн хариуг хэрэглэгч дээр харуулах үүрэгтэй. Мөн хөгжүүлж дууссаны дараа хэрэглэгчийн интерфэйсийн автоматжуулсан тест хийж байж production дээр орох боломжтой.

Сурах ур чадвар:

- State management-н гол ойлголт болох Context-н талаар мэдлэгтэй болно
- Өөрийн шаардлагад нийцсэн custom hook бичиж сурах
- Хэрхэн бичсэн компонент дээрээ UI автоматжуулсан тест бичих мэдлэг

Формын шаардлага:

- Material-UI-н Snackbar ашиглах
- Олон Toast зэрэг гаргадаг байх
- Toast нь үүсгэх, цэвэрлэх, устгах үйлдлүүдтэй байх
- Toast-н цаг дуусахад автоматаар цэвэрлэдэг байх
- UI автоматжуулсан тестийг давсан байх

### 4.2.1 Context үүсгэх

React-н өгөгдлийн урсгал нэг чиглэлд буюу зөвхөн эцгээс хүү компонент рүү утга дамжуулах чадвартай байдаг. Иймд нэг state-ээ нэгэндээ хамааралгүй олон өөр газарт ашиглагдаж байгаа компонентод ашиглахын тулд маш замбаараагүй дамжуулалт хийх шаардлага гардаг. Харин үүний нэг шийдэл нь Context гэх ойлголт бөгөөд бусад програмчлалын хэл дээр байдаг Global хувьсагч шиг ашиглах боломжтой.

Хамгийн эхлээд context-оо үүсгэж, түүний хийх үйлдлүүдийг зарлаж өгөх хэрэгтэй. Өмнөх хэрэгжүүлэлт дээр ашигласан useReducer-н мэдлэг хэрэг болов.

```
1  ...
2  export const ACTION_TOAST = "TOAST";
3  export const ACTION_RESET = "RESET";
4  export const ACTION_CLEAR = "CLEAR";
5
6  export const INITIAL_STATE: { toasts: ToastType[] } = {
7    toasts: [],
8  };
9
10 type Actions =
11   | (ActionType<typeof ACTION_TOAST> & { toast: ToastType })
12   | ActionType<typeof ACTION_RESET>
13   | (ActionType<typeof ACTION_CLEAR> & { id: string });
14
15 export const ToastContext = createContext(null);
16 ToastContext.displayName = "ToastContext";
17 export const ToastControlContext = createContext(null);
18 ToastControlContext.displayName = "ToastControlContext";
19 ...
```

Код 4.9: Context үүсгэх

Toast-н анхны утгыг зарласан байдал

```
1  ...
2  export const INITIAL_STATE: { toasts: ToastType[] } = {
3    toasts: [],
4  };
5  ...
```

Код 4.10: Toast context-н анхны утгыг зарласан байдал

Мөн манайх төсөл дээрээ Typescript ашигладаг тул мэдээж Toast-н интерфэйсийг зааж өгөх хэрэгтэй. Ингэснээр Toast ашиглаж буй хөгжүүлэгч тухайн компонент ямар төрөлтэй ямар props-уудыг хүлээн авах шаардлагатайг хялбараар харах боломжтой. Мөн хөгжүүлэлтийн явцад гарах алдаа багасна.

```
1  ...
2  export interface ToastType {
3      id?: string;
4      message: string;
5      type: "success" | "info" | "warning" | "error";
6      duration?: number;
7  }
8  ...
```

Код 4.11: Toast-н интерфэйс тодорхойлох

Одоо шаардлагын дагуу Toast үүсгэх, цэвэрлэх, устгах үйлдэл хийх функцын кодыг бичсэн

```
1  ...
2  function toast(state: typeof INITIAL_STATE, toast: ToastType): typeof
3      state {
4      return {
5          ...state,
6          toasts: [...state.toasts, toast],
7      };
8  }
9  ...
```

Код 4.12: Toast үүсгэх функц

```
1  ...
2  function clear(state: typeof INITIAL_STATE, toastId: string): typeof
   state {
3    return {
4      ...state,
5      toasts: state.toasts.filter((toast) => toast.id !== toastId),
6    };
7  }
8  ...
```

Код 4.13: Toast-г context-оос цэвэрлэх

Олон Toast зэрэг гарах боломжтой тул үүсгэсэн Toast бүр дээрээ uniq ID үүсгэж, түүний дараа тухайн ID-аар нь хайж олон цэвэрлэх боломжтой болно.

```
1  ...
2  function generateToastId() {
3    return Math.random().toString(36).substr(2, 9);
4  }
5  ...
```

Код 4.14: Uniq ID гаргах

Тухайн үйлдлийг дуудахад дээр тодорхойлж өгсөн тус тусын үүрэгтэй функцууд ажиллана. Бүх Toast-г context дотроосоо устгахдаа case дээр анхны зарласан хоосон массивыг буцааж оноож байгаа.

```
1  ...
2  function reducer(state = INITIAL_STATE, action: Actions): typeof
   state {
3    switch (action.type) {
```



```

4      case ACTION_TOAST:
5          return toast(state, action.toast);
6
7      case ACTION_RESET:
8          return INITIAL_STATE;
9
10     case ACTION_CLEAR:
11         return clear(state, action.id);
12
13     default:
14         return state;
15 }
16
17 ...

```

Код 4.15: useReducer дээр ажиллах үйлдлүүдийг заах

### 4.2.2 Custom Hook бичих

Hook бичиж өгснөөр Toast-г бүрэн ашиглах нөхцөл нь бүрдэнэ. useToast() болон useToastControl() гэсэн хоёр төрлийн hook бичиж өгсөн.

Тухайн context-г төсөл дотор буюу index.jsx файлд wrap хийж өгөөгүй нөхцөлд console дээр ашиглах боломжгүй гэсэн алдааг гаргаж өгөх шаардлагатай.

```

1      ...
2
3      export function useToastControl() {
4          const dispatch = useContext(ToastControlContext);
5
6          if (!dispatch) throw new TypeError("Please use within ToastProvider");
7
8          const reset = useCallback(() => dispatch({ type: ACTION_RESET }), [dispatch]);
9
10         ...

```

Код 4.16: Context-г буруу ашигласан үед алдаа гаргах

Toast-г шинээр үүсгэх шаардлагатай үед уг код ажиллана. Хэрэв Toast-оо үүсгэхдээ дэлгэцэнд харагдах цагийг зааж өгөөгүй бол автоматаар 4000 ms-н дараа тухайн Toast санах ой болон хэрэглэгч дээр харагдах хэсгээс цэвэрлэгдэнэ.

```
1  ...
2  const toast = useCallback(
3    (toast: ToastType) => {
4      const toastId = generateToastId();
5
6      dispatch({ toast: { ...toast, id: toastId }, type: ACTION_TOAST
7        });
8
9      setTimeout(() => {
10        dispatch({ id: toastId, type: ACTION_CLEAR });
11      }, toast.duration || 4000);
12    },
13    [dispatch]
14  );
15  ...
```

Код 4.17: Toast үүсгэх үүрэгтэй hook

Үүний дараа Toast компонентийнхоо интерфэйсийг Material-UI сан ашиглаж шийдэв.

```
1  ...
2  return (
3    <Snackbar
4      anchorOrigin={{ horizontal: "center", vertical: "bottom" }}
5      open={open}
```

```

6      autoHideDuration={duration}
7      onClose={onClose}
8      action={action}
9    >
10     <Alert onClose={onClose} severity={type} sx={{ width: "100%" }}>
11       {message}
12     </Alert>
13   </Snackbar>
14 );
15 ...

```

Код 4.18: Material-UI ашиглаж интерфэйсийг үүсгэх

Hook бичихгүйгээр шийдэх нь Toast-г ашиглахын тулд бүтэн компонент код бичих асуудалтай байсан ба custom hook-р шийдсэнээр хөгжүүлэгч ердөө ганц функцийг л дуудаж ажиллуулахад хангалттай. Ингэснээр хөгжүүлэлтийн хурданд ч эерэгээр нөлөөлнө.

```

1    ...
2    .then(
3      (res) =>
4        res.data &&
5        toast({
6          message: "Created product",
7          type: "success",
8        })
9    )
10   ...

```

Код 4.19: Toast-г дуудаж ашиглаж буй байдал

### 4.3 UI автоматжуулсан тест

Бичсэн компонентдоо тест код бичсэнээр ажиллагааг нь баталгаажуулах, алдааг илрүүлэх, хүнээр тест хийлгүүлэх шаардлагагүй болдог. Тестээ бичихдээ React багийн гишүүн Kent C. Dodds-н Jest сан дээр нэмэлт хөгжүүлэлт хийж гаргасан React Testing Library ашиглав.

Хуурамч цаг ашиглаж, тест хийх stage-н хурдыг нэмэгдүүлнэ. Ингэснээр заавал Toast дээр тавигдсан 4 секундыг хүлээх хэрэггүй.

```

1    ...
2    jest.useFakeTimers();
3    ...

```

Код 4.20: FakeTimer ашиглах

Хамгийн эхэнд хийгдэх тест бол DOM дээр зурагдаж байгаа эсэх, явуулсан string-г хэвлэж байгаа эсэх болон "ХААХ" товчлуурууд дээр дарагдахад цэвэрлэгдэж байгаа эсэхийг шалгах юм.

```

1    ...
2    const message = "Hello, it's toast";
3    render(<Toast message={message} type="success" />);
4    ...

```

Код 4.21: Toast компонентийг DOM дээр зурна

```

1    ...
2    expect(screen.getByText(message)).toBeInTheDocument();
3    ...

```

Код 4.22: DOM дээр зурагдсан эсэхийг шалгана

```

1    ...
2    userEvent.click(
3      screen.getByRole("button", {

```

```

4      name: /close/i,
5    })
6  );
7
8  await waitForElementToBeRemoved(() => screen.getByText(message));
9
10 expect(screen.queryByText(message)).not.toBeInTheDocument();
11 ...

```

Код 4.23: Хаах товчлуур дээр дарахад устсан эсэхийг шалгах

Үүний дараачаар компонентийг хоёр удаа дуудаж ажиллуулаад интерфэйс дээр харагдах нийт хугацааг өөр өөрөөр зааж өгнө. Хөгжүүлэгч дурын хугацаа зааж өгсөн үед уг хугацааны үед ажиллаж байгаа эсэхийг шалгана.

```

1  ...
2  test("duration", async () => {
3    const cssAnimation = 300;
4    const toasts = [
5      {
6        duration: 4000,
7        message: "Toast_1",
8      },
9      {
10       duration: 5000,
11       message: "Toast_2",
12     },
13   ];
14
15   toasts.forEach((toast) =>

```

```
16     render(  
17       <Toast  
18         message={toast.message}  
19         type="success"  
20         duration={toast.duration}  
21       />  
22     )  
23   );  
24   ...
```

Код 4.24: Олон Toast зэрэг гаргаж дурын хугацааг зааж өгөх

```
1   ...  
2   expect(screen.getByText("Toast_1")).toBeInTheDocument();  
3   await waitForElementToBeRemoved(() => screen.getByText("Toast_1"),  
4     {  
5       timeout: toasts[0].duration + cssAnimation,  
6     });  
7   expect(screen.queryByText("Toast_1")).not.toBeInTheDocument();  
8   ...
```

Код 4.25: Зааж өгсөн хугацааны дараа цэвэрлэгдэж байгаа эсэхийг шалгах

Хамгийн сүүлд нь миний бичсэн Toast дөрвөн өөр төрлийг хүлээж аваад тухайн төрлөөс нь хамаарч өөр загвартай харуулдаг тул уг төрлийн дагуу зурагдаж байгаа эсэхийг шалгах юм.

Toast-н хүлээж авах type-н жагсаалт:

- success
- info

- warning
- error

```
1  ...
2  test("check_type", () => {
3      const types = [
4          {
5              class: "filledSuccess",
6              message: "Success_toast",
7              type: "success",
8          },
9          { class: "filledError", message: "Error_toast", type: "error" },
10         { class: "filledInfo", message: "Info_toast", type: "info" },
11         {
12             class: "filledWarning",
13             message: "Warning_toast",
14             type: "warning",
15         },
16     ];
17
18     types.forEach((opt) => {
19         const { container } = render(
20             <Toast
21                 message={opt.message}
22                 type={opt.type as "success" | "info" | "warning" | "error"}
23             />
24         );
25         const alert = container.firstChild;
```

```
26
27     expect(alert.firstChild).toHaveClass(`MuiAlert-${opt.class}`);
28     cleanup();
29   });
30 });
31 ...
```


Код 4.26: Өгсөн type-н дагуу зурагдаж байгаа эсэхийг шалгах



## 5. ДҮГНЭЛТ

### 5.1 Үр дүн

#### 5.1.1 Аймаг сумдын мэдээлэл авдаг форм



The form consists of three stacked dropdown menus and a submit button. The first dropdown is labeled 'Хот/Аймаг:' and has 'Булган' selected. The second dropdown is labeled 'Сум/Дүүрэг:' and has 'Бугат' selected. The third dropdown is labeled 'Баг/Хороо:' and has '03' selected. Below these is a button labeled 'Бүртгүүлэх'.

Хот/Аймаг:	Булган
Сум/Дүүрэг:	Бугат
Баг/Хороо:	03
<input type="button" value="Бүртгүүлэх"/>	

Зураг 5.1: Формын эхний хувилбар

Хот/Аймаг:

Улаанбаатар | ▾

Сум/Дүүрэг:

Дүүрэг 2 | ▾

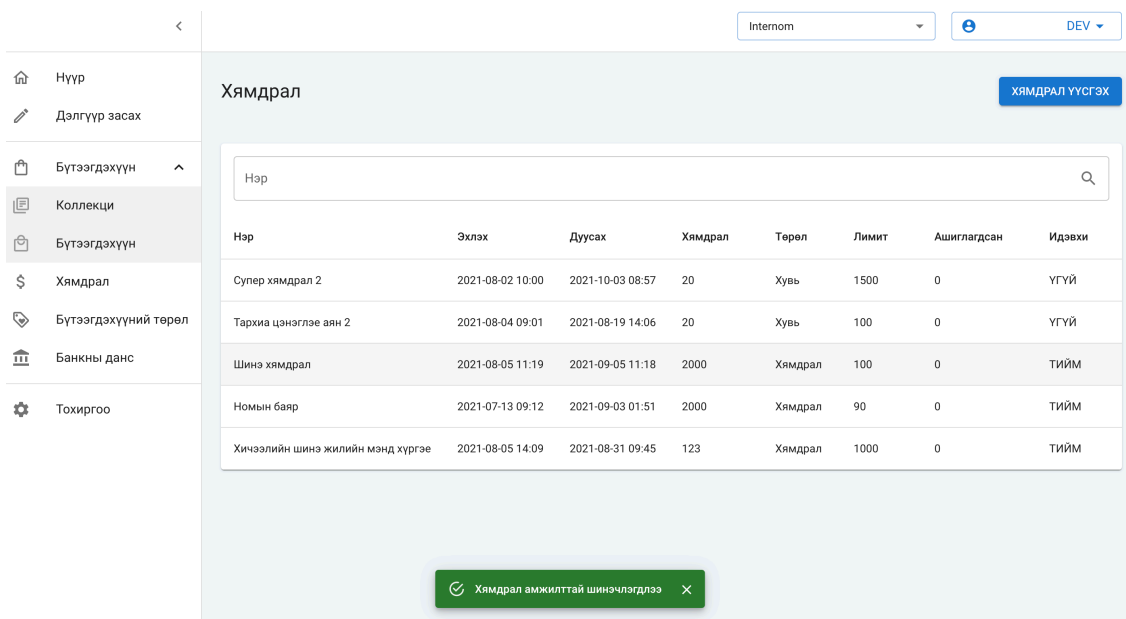
Баг/Хороо:

Сонгох | ▾

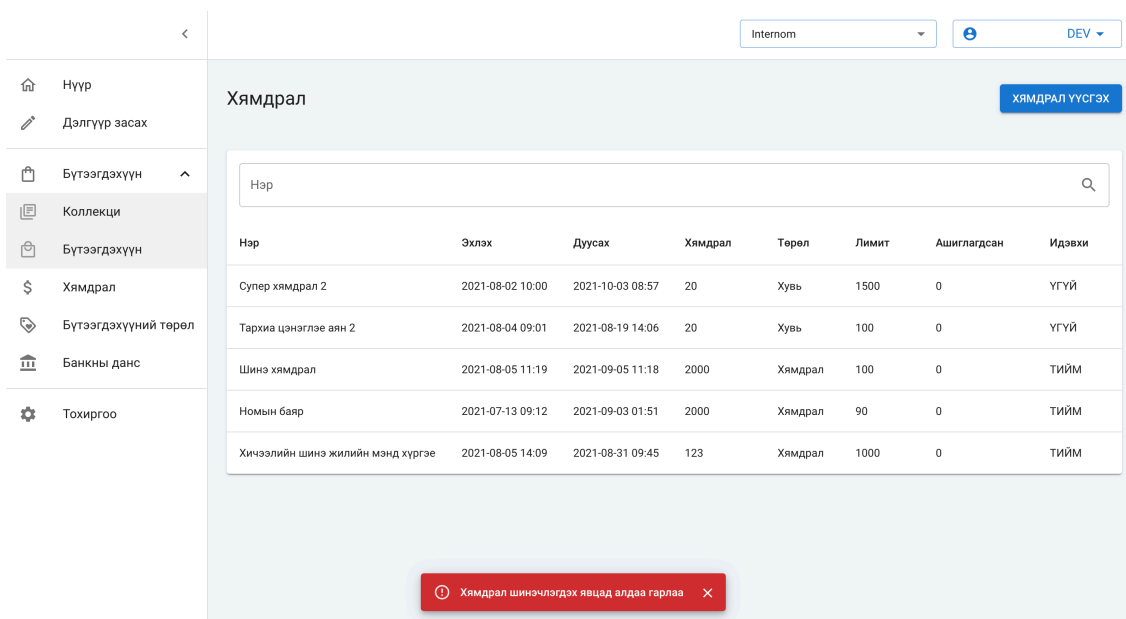
Бүртгүүлэх

Зураг 5.2: Material ui болон React Select ашигласан байдал

### 5.1.2 *Toast компонент*



Зураг 5.3: Хүсэлт амжилттай болсон үед харагдах Toast



Зураг 5.4: Хүсэлт амжилтгүй болсон үед харагдах Toast

## 5.2 Үр дүнгийн тайлан

Миний бие Хуур Мюзик Групп ХХК-д 21 хоногийн хугацаатай мэргэжлийн дадлагыг амжилттай гүйцэтгэж дуусгалаа. Уг хугацаанд хичээлийн хүрээнд үзсэн онолын ойлголтуудыг практик дээр туршиж, хэрэгжүүлсэн ба хөгжүүлэлт голчилсон технологийн компанийн ерөнхий үйл ажиллагаа, баг хооронд зохицон ажиллах чадвар, хөгжүүлэлтийн шинэ арга барилуудыг амжилттай эзэмшсэн гэж дүгнэж байна.

Continuous Integration/Continuous Deployment, GIT дээрх Feature Branch, ашиглаж буй програмчлалын хэлнийхээ давуу талыг судлан уг хэлээрээ сэтгэж бичих, том асуудлыг олон болгон хувааж багаар, алхам дэс дараатай асуудлыг шийдвэрлэх мөн ашиглаж буй сан, технологийнхоо гарын авлага буюу documentation-тай илүү сайн танилцаж уг технологийнхоо цаана нь буй концептийг хялбараар ойлгох гэх мэт чадваруудыг эзэмшсэн. Үүнийгээ цаашид илүү хөгжүүлж мэргэшсэн фронт-энд хөгжүүлэгч болохоор зорьж байна. Дадлага хийсэн компани маань хэрэгжүүлж буй төсөлдөө үргэлж технологийн шинэ туршилтын хувилбаруудыг төвөгшөөлгүйгээр хэрэглэж, түүнийхээ алдааг илрүүлж, ажлын бус цагаараа хамтдаа шийдлийг хайж олон улсын нээлттэй эхийн төсөлд гар бие оролцдог нь бусад компаниудаас онцлог. Үүний үр дүнд манай дадлагын удирдагч болох С. Дөлмандах нь React-Native-н core contributor болж, 2019 онд болсон React-Native EU гэх олон улсын хөгжүүлэгчдийн эвентэд илтгэл тавьж байсан удаатай. Би цаашид өөрийн чөлөөт цагаа ашиглан дотоодын болон олон улсын нээлттэй эхийн төсөлд хувь нэмрээ оруулж гадны чадварлаг хөгжүүлэгчдийн арга барил, код бичих туршлага, тухайн асуудлыг хэрхэн шийдсэн гэх мэт үнэтэй мэдлэгүүдийг хуримтлуулж бусад хүмүүст мөн нээлттэй эхийн төсөлд оролцохын давуу талуудыг танилцуулж уриалахаар төлөвлөсөн байгаа.

Дадлагын эхэн үед React болон Next.js технологийн талаар судлах, түүнийгээ хэрэгжүүлэх, хөгжүүлэлтийн арга барилуудтай танилцах зорилготой байсан ба цаашид мэдээлэл технологийн ямар чиглэлээр мэргэшиж, түүндээ хүрэхийн тулд хэрхэн чадварлаг болох ёстойг ойлгосон тул зорилгодоо бүрэн хүрсэн гэдэгт итгэлтэй байна.

# Bibliography

- [1] Declarative програмчлал болон Imperative програмчлалын ялгаа  
<https://codeburst.io/declarative-vs-imperative-programming-a8a7c93d9ad2>
- [2] Material-ui Beta v5 хувилбарын Card ашиглах заавар  
<https://next.material-ui.com/api/card/>

# А. УДИРДАГЧИЙН ҮНЭЛГЭЭ

Хавсралт 3

МУИС, ХШУИС-ИЙН МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ  
ТЭНХИМ

Програм хамгаалчийн АНГИЙН  
ОЮУТАН Б.Бат-Өлзий -ЫН  
ДАДЛАГЫН АЖЛЫН УДИРДАГЧИЙН ТОДОРХОЙЛОЛТ

2021 оны .. сарын ..

Програм хамгаалчийн ангийн 1861num3474 кодтой оюутан Б.Бат-Өлзий нь манай байгууллагад 2021 оны 06 сарын 02-ны өдрөөс 06 сарын 06-ны өдөр хүртэл мэргэшүүлэх дадлагыг батлагдсан удирдамж, ажлын төлөвлөгөөний дагуу гүйцэтгэлээ. Оюутан Б.Бат-Өлзий-ын удирдамжийн дагуу дадлагын ажлыг гүйцэтгэсэн байдал:

Б.Бат-Өлзий нь үйлдвэрлэлийн дадлагын хүрээнд React.js болон Next.js ашиглан хөгжүүлж байгаа вэб програм хангамжийн хөгжүүлэлтэд оролцсон. Өмнө нь React болон холбоотой технологиудыг ашиглаж байсан туршлагагүй тул судлах, суралцах зүйлс цөөнгүй бөгөөд өгсөн чиглэл, даалгаврын дагуу хичээл зүтгэлтэй, цаг тухай бүрт судалж бас суралцсан болно. Тэрээр React.js компонент болон hooks бичих зарчим, арга ажиллагаа мөн Material UI ашиглан хэрэглэгчийн интерфэйсийг бүтээхэд суралцаж бас хөгжүүлэлтэд оролцсон. Үүнээс гадна Continuous Integration хөгжүүлэлтийн аргачлал болон хэрэглэгчийн интерфэйсийн автоматжуулсан тест бичих зарчимтай танилцсан.

Бат-Өлзий нь өгсөн даалгаврыг цаг тухай бүрт нь биелүүлдэг, нээлттэй харилцаатай тул ойлгохгүй эсвэл чадахгүй байгаа зүйлээ чөлөөтэй асуудаг, тусламж авдаг давуу талтай, мөн UX сонирхдог тул бүтээгдэхүүний хэрэглэгчийн интерфэйс бүтээх ажилд бүтээлчээр оролцож олон үнэтэй санал өгч байсан. Дадлагын хугацаанд вэб хөгжүүлэлтийн тал дээр багагүй өсч хөгжсөн, туршлагатай болсон гэж дүгнэж байна.

Үнэлгээний санал: 10 оноо

Дадлагын удирдагч

Дадлагын удирдагчийн талаарх мэдээлэл:

# В. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

## В.1 Форм

### В.1.1 *useReducer* болон *React-Select* сан ашигласан байдал

```
1 import { useState, useReducer } from "react";
2 import Head from "next/head";
3 import styles from "../styles/Home.module.css";
4 import Select from "react-select";
5 import address from "../data/address";
6
7 const SET_CITY = "city";
8 const SET_DISTRICT = "district";
9 const SET_WARD = "ward";
10
11 const reducer = (state, action) => {
12   switch (action.type) {
13     case SET_CITY:
14       return {
15         city: action.index,
16         district: null,
17         ward: null,
18       };
19     case SET_DISTRICT:
20       return {
21         ...state,
22         district: action.index,
23         ward: null,
24       };
25     case SET_WARD:
26       return {
27         ...state,
28         ward: action.index,
29       };
30     default:
31       return state;
32   }
33 };
34
35 export default function Home() {
36   const [state, dispatch] = useReducer(reducer, {
37     city: null,
38     district: null,
39     ward: null,
40   });
41
42   const handleChange = (index, type) => {
43     dispatch({ type: type, index });
44   };
45 }
```

```

45
46 const register = (e) => {
47   e.preventDefault();
48   console.log(state);
49 };
50
51 return (
52   <div className={styles.container}>
53     <Head>
54       <title>Create Next App</title>
55     </Head>
56
57     <main className={styles.main}>
58       <div>
59         <form className={styles.grid} onSubmit={register}>
60           <label>
61             <p> >/:</p>
62             <Select
63               value={address.map((i, index) => ({ ...i, index }))[
64                 state.city]}
65               onChange={(e) => handleChange(e.index, SET_CITY)}
66               options={address.map((i, index) => ({ ...i, index })))}
67               getOptionLabel={(option) => option.name}
68               getOptionValue={(option) => option.index}
69               placeholder=" "
70             </Select>
71           </label>
72
73           <label>
74             <p> >/:</p>
75             <Select
76               value={
77                 state.district !== null
78                 ? address[state.city].districts.map((i, index) =>
79                   ({
80                     ...i,
81                     index,
82                   }))[state.district]
83                 : []
84             }
85             onChange={(e) => handleChange(e.index, SET_DISTRICT)}
86             options={
87               state.city !== null
88               ? address[state.city].districts.map((i, index) =>
89                 ({
90                   ...i,
91                   index,
92                 })))
93               : []
94             }
95             getOptionLabel={(option) => option.name}
96             getOptionValue={(option) => option.index}

```



```

94         placeholder="    "
95     />
96 </label>
97
98 <label>
99     <p    >/:</p>
100     <Select
101         value={
102             state.ward != null
103             ? address[state.city].districts[state.district].
104               wards.map(
105                 (i, index) => ({ ...i, index })
106               )
107             : []
108         }
109         onChange={(e) => handleChange(e.index, SET_WARD)}
110         options={
111             state.district != null
112             ? address[state.city].districts[state.district].
113               wards.map(
114                 (i, index) => ({ ...i, index })
115               )
116             : []
117         }
118         getOptionLabel={(option) => option.name}
119         getOptionValue={(option) => option.index}
120         placeholder="    "
121     />
122 </label>
123 <button className={styles.registerBtn    }></button>
124 </form>
125 </div>
126 </main>
127 </div>
128 );
129 }

```

### B.1.2 Functional component дээр state ашигласан байдал

```

1  import { useEffect, useState } from "react";
2  import Head from "next/head";
3  import styles from "../styles/Home.module.css";
4  import axios from "axios";
5  import useForm from "../utils/useForm";
6
7  const fetchData = (url) => {
8      //get data from next.js api
9      return axios
10         .get(`http://localhost:3000/api/${url}`)
11         .then((res) => {
12             const results = res.data;

```

```
13     return results;
14   })
15   .catch((err) => {
16     console.error(err);
17   });
18 };
19
20 export default function Home() {
21   const [values, handleChange] = useForm();
22   const [data, setData] = useState({
23     cities: [],
24     districts: [],
25     wards: [],
26   });
27
28   useEffect(() => {
29     fetchData(`cities`)
30       .then((res) => {
31         setData({ ...data, cities: res });
32       })
33       .catch((err) => {
34         console.error(err);
35       });
36   }, []);
37
38   const register = (e) => {
39     e.preventDefault();
40     console.log(values);
41   };
42
43   const handleSelect = (id, type) => {
44     if (type == "city") {
45       fetchData(`cities/${id}`)
46         .then((res) => {
47           setData({ ...data, districts: res, wards: [] }); //set
48             districts and clear wards data
49         })
50         .catch((err) => {
51           console.error(err);
52         });
53     } else if (type == "district") {
54       //get wards
55       fetchData(`cities/${values.city}/${id}`)
56         .then((res) => {
57           setData({ ...data, wards: res });
58         })
59         .catch((err) => {
60           console.error(err);
61         });
62     }
63   };
64 }
```

```

64   const OptionItems = (props) => {
65     const options = props.items.map((item) => {
66       return (
67         <option key={item.id} value={item.id}>
68           {item.name}
69         </option>
70       );
71     });
72
73     return options;
74   };
75
76   return (
77     <div className={styles.container}>
78       <Head>
79         <title>Create Next App</title>
80       </Head>
81
82       <main className={styles.main}>
83         <div>
84           <form className={styles.grid} onSubmit={register}>
85             <label>
86               /:
87             <select
88               value={values.city}
89               defaultValue="default"
90               name="city"
91               onChange={(e) => {
92                 handleChange(e.target.name, e.target.value);
93                 handleSelect(e.target.value, "city");
94               }}
95             >
96               <option value="default" hidden>
97                 /
98               </option>
99
100              {data.cities.length > 0 ? (
101                <OptionItems items={data.cities} />
102              ) : null}
103
104              {/* {data.cities.map((item) => {
105                return (
106                  <option key={item.id} value={item.name}>
107                    {item.name}
108                  </option>
109                );
110              }}} */}
111            </select>
112          </label>
113
114          <label>
115            /:

```

```

116         <select
117             value={values.district}
118             defaultValue="default"
119             name="district"
120             onChange={(e) => {
121                 handleChange(e.target.name, e.target.value);
122                 handleSelect(e.target.value, "district");
123             }}
124         >
125             <option value="default" hidden>
126                 /
127             </option>
128             {data.districts.length > 0 ? (
129                 <OptionItems items={data.districts} />
130             ) : null}
131         </select>
132     </label>
133
134     <label>
135         /:
136         <select
137             value={values.ward}
138             defaultValue="default"
139             name="ward"
140             onChange={(e) => {
141                 handleChange(e.target.name, e.target.value);
142             }}
143         >
144             <option value="default" hidden>
145                 /
146             </option>
147             {data.wards.length > 0 ? (
148                 <OptionItems items={data.wards} />
149             ) : null}
150         </select>
151     </label>
152     <button style={{ margin: "5px" }}></button>
153 </form>
154 </div>
155 </main>
156 </div>
157 );
158 }

```

## B.2 Toast компонент

### B.2.1 Toast/context.tsx - Context үүсгэх

```

1 import { createContext, useReducer } from "react";
2

```

```
3 import { ActionType } from "types";
4
5 export interface ToastType {
6   id?: string;
7   message: string;
8   type: "success" | "info" | "warning" | "error";
9   duration?: number;
10 }
11
12 export const ACTION_TOAST = "TOAST";
13 export const ACTION_RESET = "RESET";
14 export const ACTION_CLEAR = "CLEAR";
15
16 export const INITIAL_STATE: { toasts: ToastType[] } = {
17   toasts: [],
18 };
19
20 type Actions =
21   | (ActionType<typeof ACTION_TOAST> & { toast: ToastType })
22   | ActionType<typeof ACTION_RESET>
23   | (ActionType<typeof ACTION_CLEAR> & { id: string });
24
25 export const ToastContext = createContext(null);
26 ToastContext.displayName = "ToastContext";
27 export const ToastControlContext = createContext(null);
28 ToastControlContext.displayName = "ToastControlContext";
29
30 function toast(state: typeof INITIAL_STATE, toast: ToastType): typeof
31   state {
32   return {
33     ...state,
34     toasts: [...state.toasts, toast],
35   };
36 }
37
38 function clear(state: typeof INITIAL_STATE, toastId: string): typeof
39   state {
40   return {
41     ...state,
42     toasts: state.toasts.filter((toast) => toast.id !== toastId),
43   };
44 }
45
46 function reducer(state = INITIAL_STATE, action: Actions): typeof state
47   {
48   switch (action.type) {
49     case ACTION_TOAST:
50       return toast(state, action.toast);
51     case ACTION_RESET:
52       return INITIAL_STATE;
53     case ACTION_CLEAR:
54       return clear(state, action.id);
```

```

52     default:
53         return state;
54     }
55 }
56
57 export const ToastProvider = ({ children }) => {
58     const [state, dispatch] = useReducer(reducer, INITIAL_STATE);
59
60     return (
61         <ToastContext.Provider value={state}>
62             <ToastControlContext.Provider value={dispatch}>
63                 {children}
64             </ToastControlContext.Provider>
65         </ToastContext.Provider>
66     );
67 };

```

### B.2.2 Toast/hooks.ts - Custom hook бүтээх

```

1  import { useCallback, useContext, useMemo } from "react";
2
3  import {
4      ACTION_CLEAR,
5      ACTION_TOAST,
6      ACTION_RESET,
7      ToastContext,
8      ToastControlContext,
9      ToastType,
10 } from "../context";
11
12 function generateToastId() {
13     return Math.random().toString(36).substr(2, 9);
14 }
15
16 export function useToast() {
17     const state = useContext(ToastContext);
18     if (!state) throw new TypeError("Please use within ToastProvider");
19     return state;
20 }
21
22 export function useToastControl() {
23     const dispatch = useContext(ToastControlContext);
24     if (!dispatch) throw new TypeError("Please use within ToastProvider");
25
26     const reset = useCallback(() => dispatch({ type: ACTION_RESET }), [
27         dispatch
28     ]);
29
30     const toast = useCallback(
31         (toast: ToastType) => {
32             const toastId = generateToastId();

```

```

31
32     dispatch({ toast: { ...toast, id: toastId }, type: ACTION_TOAST
33         });
34
35     setTimeout(() => {
36         dispatch({ id: toastId, type: ACTION_CLEAR });
37     }, toast.duration || 4000);
38     [dispatch]
39 );
40
41 return useMemo(() => {
42     return { reset, toast };
43 }, [toast, reset]);
44 }

```

### B.2.3 Toast/index.ts

```

1 export * from "./Toast";
2 export * from "./context";
3 export * from "./hooks";

```

### B.2.4 Toast/Toast.tsx - Үндсэн Toast компонент

```

1 import { useState, forwardRef } from "react";
2
3 import { IconButton, Snackbar } from "@material-ui/core";
4 import MuiAlert, { AlertProps } from "@material-ui/core/Alert";
5 import { Close as CloseIcon } from "@material-ui/icons";
6
7 import { ToastType } from "./context";
8 import { useToast } from "./hooks";
9
10 const Alert = forwardRef<HTMLDivElement, AlertProps>((function Alert(
11     props,
12     ref
13 ) {
14     return <MuiAlert elevation={6} ref={ref} variant="filled" {...props}
15         />;
16 }));
17
18 export function Toast({ message, type, duration = 4000 }: ToastType) {
19     const [open, setOpen] = useState(true);
20
21     const onClose = () => setOpen(false);
22
23     const action = (
24         <IconButton
25             size="small"

```

```
25     aria-label="close"
26     color="inherit"
27     onClick={onClose}
28   >
29     <CloseIcon fontSize="small" />
30   </IconButton>
31 );
32
33 return (
34   <Snackbar
35     anchorOrigin={{ horizontal: "center", vertical: "bottom" }}
36     open={open}
37     autoHideDuration={duration}
38     onClose={onClose}
39     action={action}
40   >
41     <Alert onClose={onClose} severity={type} sx={{ width: "100%" }}>
42       {message}
43     </Alert>
44   </Snackbar>
45 );
46 }
47
48 export function ToastContainer() {
49   const { toasts } = useToast();
50   return toasts.map((toast) => <Toast key={toast.id} {...toast} />);
51 }
```



## **В.3 Toast компонент дээр интерфэйсийн автоматжуулсан тест бичсэн байдал**

### ***В.3.1 tests/Toast.test.tsx***

```
1 import {
2   cleanup,
3   render,
4   screen,
5   waitForElementToBeRemoved,
6 } from "@testing-library/react";
7 import userEvent from "@testing-library/user-event";
8
9 import { Toast } from "../../components/Toast";
10
11 jest.useFakeTimers();
12 describe("Test: Toast component", () => {
13   test("show message and delete toast after close button", async () => {
14     {
15       const message = "Hello, it's toast";
16
17       render(<Toast message={message} type="success" />);
18       expect(screen.getByText(message)).toBeInTheDocument();
19
20       userEvent.click(
21         screen.getByRole("button", {
22           name: /close/i,
23         })
24       );
25
26       await waitForElementToBeRemoved(() => screen.getByText(message));
27       expect(screen.queryByText(message)).not.toBeInTheDocument();
28     }
29   });
30
31   test("duration", async () => {
32     const cssAnimation = 300;
33     const toasts = [
34       {
35         duration: 4000,
36         message: "Toast 1",
37       },
38       {
39         duration: 5000,
40         message: "Toast 2",
41       },
42     ];
43
44     toasts.forEach((toast) => {
45       render(
46         <Toast
47           message={toast.message}
```

```

46         type="success"
47         duration={toast.duration}
48     />
49     )
50 };
51
52 expect(screen.getByText("Toast_1")).toBeInTheDocument();
53 await waitForElementToBeRemoved(() => screen.getByText("Toast_1"),
54     {
55         timeout: toasts[0].duration + cssAnimation,
56     });
57 expect(screen.queryByText("Toast_1")).not.toBeInTheDocument();
58
59 expect(screen.getByText("Toast_2")).toBeInTheDocument();
60 await waitForElementToBeRemoved(() => screen.getByText("Toast_2"),
61     {
62         timeout: 1000 + cssAnimation,
63     });
64 expect(screen.queryByText("Toast_2")).not.toBeInTheDocument();
65
66 test("check_type", () => {
67     const types = [
68         {
69             class: "filledSuccess",
70             message: "Success_toast",
71             type: "success",
72         },
73         { class: "filledError", message: "Error_toast", type: "error" },
74         { class: "filledInfo", message: "Info_toast", type: "info" },
75         {
76             class: "filledWarning",
77             message: "Warning_toast",
78             type: "warning",
79         },
80     ];
81
82     types.forEach((opt) => {
83         const { container } = render(
84             <Toast
85                 message={opt.message}
86                 type={opt.type as "success" | "info" | "warning" | "error"}
87             />
88         );
89         const alert = container.firstChild;
90
91         expect(alert.firstChild).toHaveClass(`MuiAlert-${opt.class}`);
92         cleanup();
93     });
94 });

```