

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

Батбаярын Бат-Өлзий

**МЭДЭЭЛЛИЙН ЭХ СУРВАЛЖАА  
ХУВААЛЦАХ СОШИАЛ ПЛАТФОРМ**  
**(Information sources sharing social platform)**

Програм Хангамж (D 061302)  
Бакалаврын судалгааны ажил

Улаанбаатар

2022 он

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

**МЭДЭЭЛЛИЙН ЭХ СУРВАЛЖАА ХУВААЛЦАХ  
СОШИАЛ ПЛАТФОРМ**  
**(Information sources sharing social platform)**

Програм Хангамж (D 061302)  
Бакалаврын судалгааны ажил

Удирдагч: \_\_\_\_\_ Мастер Р. Жавхлан  
Гүйцэтгэсэн: \_\_\_\_\_ Б. Бат-Өлзий (18B1NUM3474)

Улаанбаатар

2022 он

# Зохиогчийн баталгаа

Миний бие Батбаярын Бат-Өлзий ”МЭДЭЭЛЛИЙН ЭХ СУРВАЛЖАА ХУВААЛЦАХ СОШИАЛ ПЛАТФОРМ“ сэдэвтэй судалгааны ажлыг гүйцэтгэсэн болохыг зарлаж дараах зүйлсийг баталж байна:

- Ажил нь бүхэлдээ эсвэл ихэнхдээ Монгол Улсын Их Сургуулийн зэрэг горилооор дэвшүүлсэн болно.
- Энэ ажлын аль нэг хэсгийг эсвэл бүхлээр нь ямар нэг их, дээд сургуулийн зэрэг горилооор 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 .....	iv
ХҮСНЭГТИЙН ЖАГСААЛТ .....	vi
КОДЫН ЖАГСААЛТ .....	vii
УДИРТГАЛ .....	1
Зорилго .....	1
Зорилт .....	1
Сэдэв сонгох үндэслэл .....	1
Ач холбогдол .....	3
БҮЛГҮҮД .....	4
1. СЭДВИЙН ЕРӨНХИЙ СУДАЛГАА .....	4
1.1 Үндсэн ойлголтууд .....	4
1.2 Ижил төсөөтэй систем .....	6
1.3 Ашиглах технологи .....	8
2. СИСТЕМИЙН ШААРДЛАГА .....	15
2.1 Шаардлагын шинжилгээ .....	15
2.2 UX/UI шаардлага .....	16
3. СИСТЕМИЙН АРХИТЕКТУР, ЗОХИОМЖ .....	18
3.1 Системийн архитектур .....	18
3.2 Ажлын явцын диаграм .....	20
3.3 UX/UI дизайн .....	22
3.4 Өгөгдлийн сан .....	29
4. ХЭРЭГЖҮҮЛЭЛТ .....	34
4.1 Хөгжүүлэлтийн орчныг бэлдэх .....	34
4.2 Код хөгжүүлэлт .....	35
4.3 Yp дүн .....	42

5. ДҮГНЭЛТ .....	46
НОМ ЗҮЙ .....	47
ХАВСРАЛТ .....	48
A. PRISMA МОДЕЛ .....	48
B. PRISMA CLIENT АШИГЛАХ .....	50
C. NEXT.JS ДЭЭР REMOTE ДОМАЙН ЗӨВШӨӨРӨХ .....	51
D. TAILWIND CUSTOM-CLASS YYCГЭХ .....	52
E. ХЭРЭГЛЭГЧИЙН МЭДЭЭЛЛИЙГ ГАРГАХ ENDPOINT .....	53
F. ХОЛБООС ДЭЭР VOTE ӨГӨХ ENDPOINT .....	54
G. LAYOUT КОМПОНЕНТИЙГ APP КОМПОНЕНТ ДЭЭР АШИГЛАХ .....	57

## Зургийн жагсаалт

Зураг	Хуудас	
1	Интернэт хэрэглэгчдээс авсан судалгааны үр дүн .....	2
1.1	Open Graph протоколын тагуудын систем дээр ашиглаж буй байдал	5
1.2	AllTop.com сайтын харагдах байдал .....	6
1.3	Linktree сайтын харагдах байдал .....	8
1.4	2022 онд хамгийн ихээр ашиглаж буй вэб фрэймворкийн жагсаалт .	8
1.5	Prisma тойм зураг .....	13
1.6	Figma ашиглаж Card-н hover эффект дээр prototype хийсэн компонент	13
3.1	Client-Server Архитектурын үлгэр загварын дүрслэл .....	18
3.2	Хэрэгжүүлэх гэж буй системийн архитектур .....	19
3.3	Ажлын явцын диаграм .....	20
3.4	Persona 1 - Болорчуууны мэдээлэл .....	22
3.5	Persona 2 - Шүрэнцэцг .....	23
3.6	Нүүр хуудас .....	24
3.7	Холбоосуудаа оруулах форм .....	24
3.8	Оруулсан холбоосуудыг нийтлэсний дараах хуудас .....	25
3.9	Оруулсан нийтлэл бусад хүмүүст харагдах байдал .....	25
3.10	Ашигласан компонентуудын жишээ .....	27
3.11	Нэвтэрсний дараах нүүр хуудас .....	27
3.12	Бүх холбоосуудыг төрлөөр нь шүүж харах хуудас .....	27
3.13	Хэрэглэгчдийн жагсаалт харагдах хуудас .....	28
3.14	Бусдын профайлыг харах хуудас .....	28
3.15	Бүлэг холбоос доторх холбоосуудыг харах хуудас .....	28
3.16	Системийн RD Schema .....	29
4.1	Төслийн файлын бүтэц .....	35

4.2 Post хүснэгтийн харагдац .....	38
4.3 Нийтлэлийн жагсаалтыг агуулсан JSON Object .....	40
4.4 Нүүр хуудас .....	43
4.5 Платформ дээрх сэдвүүдийн жагсаалт болон бүх нийтлэлийг харах .	43
4.6 Холбоосуудыг сэдвээр нь шүүж харах .....	44
4.7 Платформ дээр бүртгүүлсэн хэрэглэгчдийн жагсаалтыг харах .....	44
4.8 Олон холбоосыг бүлэглэж харуулж буй хуудас .....	45
4.9 Сонгосон хэрэглэгчийн оруулсан холбоосуудыг нэг дор харах .....	45

## Хүснэгтийн жагсаалт

2.1	Функциональ шаардлага .....	15
2.2	Функциональ бус шаардлага .....	16
2.3	User Interface дизайны шаардлага .....	17
3.1	user хүснэгт .....	30
3.2	post хүснэгт .....	31
3.3	post_tag хүснэгт .....	31
3.4	follower хүснэгт .....	32
3.5	user_vote хүснэгт .....	32
3.6	group хүснэгт .....	32
3.7	category хүснэгт .....	33
3.8	bookmark хүснэгт .....	33
3.9	tag хүснэгт .....	33

# Кодын жагсаалт

1.1	JSX ашиглаж 'container' кластай html элемент буцаах компонент . . . . .	10
4.1	Хэрэглэгчийн оруулсан холбоосыг харагдах компонент . . . . .	36
4.2	Өгөгдлийн сангийн хүснэгтийг Prisma ашиглан үүсгэх . . . . .	37
4.3	Next.js дээрээ end point гаргах . . . . .	38
4.4	ServerSideProps болон axios ашиглан хүсэлт илгээж өгөгдлөө татаж авах .	41
4.5	Card компонентод ирсэн утгуудыг props-оор дамжуулж DOM дээр рендерлэх	41

## **УДИРТГАЛ**

Орчин цагт интернэт сүлжээ хүмүүсийн амьдралын салшгүй нэг хэсэг болж түүнийгээ дагаад бүхий л мэдээллийг интернэт дэх нэмэлт эх сурвалжуудаас авдаг болсон. Аливаа зүйлс томрох тусам найдвартай, чанартай зүйлстэй зэрэгцэн худал хуурмаг зүйлс их тархдаг билээ. Үүнтэй ижлээр хүмүүс интернэт дэх хэт олон эх сурвалжууд дунд төөрч, хэрэгцээгүй мэдээлэл унших, түүнийгээ бусдад хуваалцаж бусдыг болон өөрсдийгөө үргэлжлүүлэн хохироскоор байна. Үүнээс авч үзвэл интернэт хэрэглэгчид мэргэжлийн хүмүүсийн цуглуулсан чанартай агуулгатай эх сурвалжуудыг хялбар, нэгдсэн байдлаар харах, мөн өөрөө эх сурвалжуудаа цуглуулж бусдад хуваалцах шаардлага тулгарч байна.

### **Зорилго**

Мэдээллийн эх сурвалжуудаа нэгтгэж бусдад хуваалцах, бусдын цуглуулсан эх сурвалжуудыг хялбар байдлаар харах, үнэлгээ өгөх боломжтой веб апп бүтээж интернэт хэрэглэгчдийн нэмэлт эх сурвалж олох явцыг хялбаршуулахаар зорьж байна.

### **Зорилт**

Уг веб аппыг хөгжүүлэхдээ дараах үе шатын дагуу ажиллана.

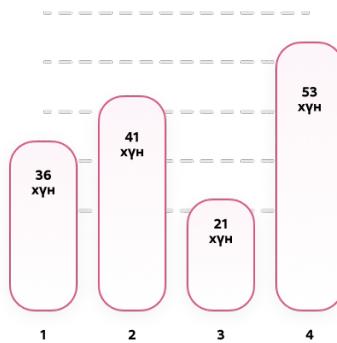
1. Хэрэглэгчийн үндсэн шаардлагуудыг тодорхойлох
2. UX судалгаа хийж хэрэглэгч суурьтай хялбар интерфейс дизайн гаргах
3. Ашиглах технологийг онол болон практик дээр суурилж судлах,
4. Системийн архитектурыг зохион байгуулж бэлдэх
5. Гаргасан баримт бичиг дээрээ тулгуурлаж хөгжүүлэлтээ хийх

### **Сэдэв сонгох үндэслэл**

2022 оны 4-р сарын байдлаар дэлхийн нийт хүм амын 63.1 хувь буюу ойролцоогоор 5 тэрбум хүн интернэт сүлжээг хэрэглэдэг ба үүнээс 4.7 тэрбум хүн буюу 94 хувь нь өдөр

тутмын амьдралдаа сошиал сүлжээг ашиглаж бусадтай харилцах, мэдээ мэдээллээ авах хэрэгцээгээ хангадаг байна<sup>1</sup>. Θөрөөр хэлбэл дэлхийн хүн амын талаас их хувь нь хэвмэлэл биет материал, телевиз, радиогоос гадна нэмэлтээр интернэт дэх веб холбоосуудаар дамжин нэмэлт мэдээллээ авдаг гэсэн үг билээ.

Миний хувьд сошиал сүлжээ ашиглан уг сэдэвтэй холбоотой судалгааг<sup>2</sup> 58 интернэт хэрэглэгчдээс авсан бөгөөд тэдгээрт тулгарсан асуудлууд болон хариултуудыг дүгнэж үз-вэл



Зураг 1: Интернэт хэрэглэгчдээс авсан судалгааны үр дүн

1. Mash олон хуурамч мэдээллүүдэд өртдөг - 36 хүн
2. Θөрт хэрэгтэй мэдээллээ олоход цаг их зарцуулдаг - 41 хүн
3. Сэдвийнхээ хүрээнд өөрийн олж авсан эх сурвалжуудын тоонд сэтгэл хангалуун бус байдаг - 21 хүн
4. Ном, зурагтаас илүү интернэт ашиглан мэдээллээ авдаг - 53 хүн

гэсэн хариу гарсан. Иймд эдгээр болон бусад хэрэглэгчдэд шаардлагатай зөвхөн эх сурвалжуудаа олж авах, бусад хуваалцах боломжтой сошиал платформыг бүтээсэн нь зөв гэж үзсэн тул уг сэдвийг сонгон хөгжүүлж байна.

<sup>1</sup>Дэлхийн интернэт хэрэглэгчдийн судалгаа: <https://www.statista.com/statistics/617136/digital-population-worldwide/>

<sup>2</sup>Эх сурвалжтай холбоотой судалгаа: <https://docs.google.com/spreadsheets/d/1CfVnHT29pkBiL0JBv7bXiKtOxERQFTzt0PmOaQ-mcd0/edit?usp=sharing>

## Ач холбогдол

Уг платформыг бүтээснээр миний хувьд бүтээгдэхүүнийг эхнээс нь дуусах хүртэлх хөгжүүлэлтийн үе шатуудтай танилцах, түүнийгээ практикаар хэрэгжүүлэх боломж бүрдэнэ.

Мөн бүтээгдэхүүн болгон гаргаж хэрэглээнд нэвтрүүлснээр дээр дурдсан асуудлыг тодорхой хэмжээнд багасах боломжтой гэж үзэж байна. Mash олон төрлөөр хэрэглэх боломжтой ба жишээ нь их сургуулийн багш оюутнууддаа зориулж сэдвийн хүрээнд нэмэлт материал бэлдэхдээ интернэт сүлжээнд байрласан унших ёстой судалгааны ажил, үзэх ёстой бичлэг, дагаж хийх ёстой практик хичээлүүдийнхээ веб холбоосуудыг хялбар байдлаар нэгтгэж, хуваалцах боломж бүрдэх юм. Мөн багш нь өөрийн сонирхдог сэдвийн хүрээнд бусдын оруулсан эх сурвалжуудтай танилцах боломжтой.

# 1. СЭДВИЙН ЕРӨНХИЙ СУДАЛГАА

Сэдвийн хүрээнд өмнө нь ажиллаж үзээгүй олон шинэ технологиудыг судалж, хооронд нь харьцуулан системдээ ашиглахад хамгийн тохиромжтой технологиудыг сонгох, цаашлаад технологийн цар хүрээнд тааруулж системээ хэрхэн өргөжүүлэх боломжтой талаарх мэдлэгийг авлаа. Уг бүлэгт өөрийн хийсэн судалгаан дээрээ үндэслэж сонгосон технологиуд болон веб платформоо бүтээхэд суурь болох ойлголтуудын талаар оруулав.

## 1.1 Үндсэн ойлголтууд

Уг судалгааны ажлыг амжилттай хийж дуусгахад доорх ойлголтуудыг өөрийн болгосон байх ёстой ба эдгээр үндсэн аргачлалууд дээр манай платформ маань суурилж байгаа билээ.

### 1.1.1 *News Aggregation*

Тодорхой ижил контентууд буюу онлайн мэдээ, нийтлэл, подкаст болон видео блогуудыг хялбар байдлаар нэгтгэж харах зориулттай хэрэглэгчид зориулсан веб аппликацийнг хэлдэг. Өөрөөр хэлбэл олон вебсайтууд дээрх нийтлэлүүдийг хамтад нь нэг хуудаснаас харах, түүнийг унших боломжтой апп юм. Гэхдээ анхаарах зүйл нь өөр вебсайт дээр тавигдсан мэдээллийг бүтнээр нь хуулж авах нь зохиогчийн эрхийн асуудлыг үүсгэдэг. Иймд тухайн мэдээллийн эхний 50 үгийг харуулах эсвэл хэрэглэгчид веб холбоос руу шилжих боломжийг санал болгож өгөх шаардлагатай. News Aggregation нь дотроо хэд хэдэн төрлүүд байдаг ба үүнд

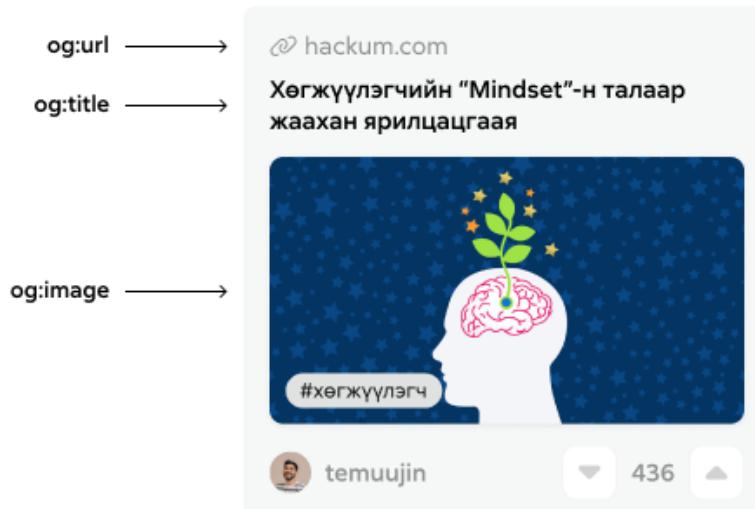
- News Aggregator Websites - Өөр өөр төрлийн эх сурвалжуудаас системчлэгдсэн аргаар автоматаар хэрэгтэй мэдээллийг татан авч хэрэглэгчдэд нэгтэн харуулдаг вебсайт
- Web-Based Feed Readers - Хэрэглэгчдэд интернэтээс эх сурвалжуудаа олж тухайн үндсэн домайныг өөрийн веб дээрээ нэмэх боломжийг олгосон веб дээр суурилсан

систем

- Feed Reader Applications - Суурин компьютер, гар утас, таблет зэрэг төхөөрөмж дээр суусан мэдээлэл цуглуулах, тэдгээр мэдээллийг бүлэглэн харуулах зориулалттай хэ-рэглэгч сурьтай интерфейс дизайнтай бүтээгдсэн аппликашнууд. Хамгийн өргөн хэрэглээтэйд гар утасны имэйл аппууд ордог
- Social News Aggregators - Интернэт сүлжээнд хамгийн эрэлттэй байгаа мэдээнүүдийг цуглуулж, нэмэлтээр засварлан хүмүүст хүргэх зориулалттай вебсайт эсвэл аппликашн

### 1.1.2 *Open Graph Protocol*

Бусад вебсайтууд дээрх мэдээний гарчиг, тайлбар, зураг гэх мэт мэдээллийг авч платформ дээрээ харуулахын тулд Open Graph протоколыг ашиглана. Уг протоколыг 2010 онд Facebook буюу одоогийн Meta компани өөрсдийн сошиал сүлжээндээ ашиглахын тулд гаргаж байсан. Энэхүү протокол нь Facebook сошиал сүлжээндээ бусад вэбсайтуудын тодорхой мэдээллүүдийг хялбараар авч, хэрэглэгчдэд харагдуулах зориулалттай. Open Graph дээр үндсэн 4 tag байдаг ба тус бүр өөрсдийн гэсэн үүрэгтэй. Үүнд



Зураг 1.1: Open Graph протоколын тагуудын систем дээр ашиглаж буй байдал

## 1.2. ИЖИЛ ТӨСӨӨТЭЙ СИСТЕМ БҮЛЭГ 1. СЭДВИЙН ЕРӨНХИЙ СУДАЛГАА

1. og:title - Тухайн веб хуудсын гарчиг
2. og:type - Вебсайтын төрөл
3. og:image - Тухайн веб хуудас дээрх агуулгыг илэрхийлсэн png, jpg, jpeg форматтай зураг
4. og:url - Тухайн веб хуудас руу орох холбоос

Эдгээр tag-үүд нь вебийн HTML хуудасны толгой хэсэгт <meta> tag дотор байршил ба сошиал сүлжээнүүдээс гадна вебийн хандалтыг ихэсгэхийн тулд SEO буюу хайлтын оновчлол дээр ашигладаг. Мөн үндсэн tag-үудаас гадна нэмэлт мэдээллүүдийг агуулсан tag-үүд байдаг.

### 1.2 Ижил төсөөтэй систем

#### 1.2.1 AllTop.com

AllTop.com вебсайтыг Гая Кавасаки тэргүүтэй гурван залуу нийлж үүсгэн байгуулсан бөгөөд англи хэл дээр бичигдсэн дэлхийн шилдэг эх сурвалжууд руу үсрэх холбоосыг нэгтгэж, төрөлжүүлэн хэрэглэгчдэд харуулдаг News Aggregation төрлийн вебсайт юм.

##### AllTop's Most Popular Sites

TechCrunch ( <a href="https://techcrunch.com">techcrunch.com</a> )	Wired ( <a href="https://www.wired.com">www.wired.com</a> )	Mashable ( <a href="https://mashable.com">mashable.com</a> )
And the winner of Startup Battlefield at Disrupt S...	Ukraine Could Never Afford to Bet on Starlink	What that 'Black Adam' mid-credit scene means f...
Sources: BeReal raised \$60M in its Series B earlier...	12 Best Portable Battery Chargers (2022): For Pho...	An electric version of the classic '60s Mini Moke is...
EV maker Arrival cutting jobs again in pivot away f...	High-Tech Cars Are Killing the Auto Repair Shop	Webb telescope's Pillars of Creation shows us thi...
Elon Musk reportedly wants to lay off 75% of Twit...	Cars Are Just Software Now	There's a shortage of diabetes drugs because of a ...
Snap stock down 25% as the social network strug...	A New 3,200-Megapixel Camera Has Astronomers...	How to see the Orionid meteor shower without a ...
Watch Draymond Green discuss investing, media ...	Celestron StarSense Dobsonian Telescope: Navig...	
As healthcare goes remote, Equipt Health brings ...	How Vice Society Got Away With A Global Ransom...	
Handoff is creating a more equitable workforce th...	Apple Is an Ad Company Now	
Washington Post Politics ( <a href="https://www.washingtonpost.com">www.washingtonpost.com</a> )	NPR News ( <a href="https://www.npr.org">www.npr.org</a> )	Reddit News ( <a href="https://www.reddit.com">www.reddit.com</a> )
Post Politics Now: Biden to raise money for Fetter...	The Supreme Court won't block the student loan ...	Hans Niemann Files \$100 Million Lawsuit Against ...
Lindsey Graham must testify in 2020 election inve...	The EPA starts a civil rights probe into Mississippi'...	U.S. says Iranian troops "directly engaged" in Cri...
A perfect encapsulation of the partisan divide on ...	Are you voting in the midterm elections? We want...	Schoolboy protester dies in Iran after reportedly ...
King family, veterans demand accountability for U...	The Pentagon will pay for service members to tra...	Diwali will be a public school holiday in New York ...
Okla. governor mocks Democratic challenger for ...	A jury finds that Kevin Spacey didn't molest actor ...	Texas secretary of state to send inspectors to obs...

Зураг 1.2: AllTop.com сайтын харагдах байдал

Манай бүтээх гэж байгаа вебээс ялгаатай тал нь

## *1.2. ИЖИЛ ТӨСӨӨТЭЙ СИСТЕМ БҮЛЭГ 1. СЭДВИЙН ЕРӨНХИЙ СУДАЛГАА*

- Зөвхөн RSS feed-тэй сайтууд дээр ажиллах боломжтой
- Интерфейсийн хувьд хэт энгийн
- Найдвартай эх сурвалжуудаас авна гэсэн учир цаанаасаа тодорхойлж өгсөн хэдэн сайтуудтай

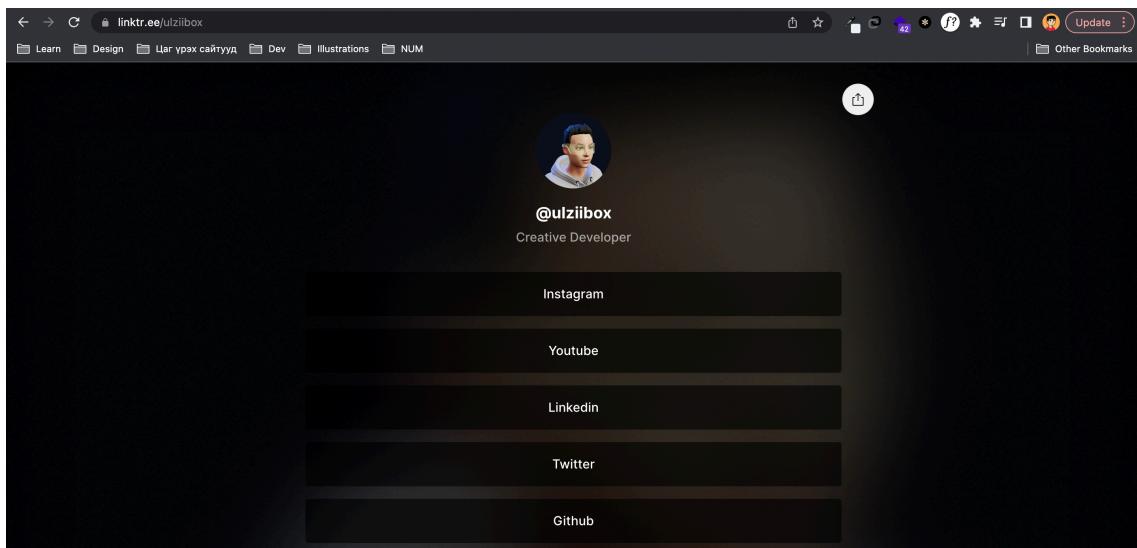
Хөгжүүлэлтийн технологийн хувьд Javascript хэл дээр суурилсан jQuery сан болон Bootstrap CSS фрэймворкийг ашиглан бүтээсэн байна. Одоогоор News Aggregation төрлийн вебсайтууд дундаас хамгийн алдартай, хандалт ихтэй сайтуудын нэг бөгөөд үүний гол хүчин зүйл нь энэхүү сайт дээр байх мэдээг бүлэглэж ангилсан байдал болон олон жил тасралтгүй үйл ажиллагаа явуулсан байдал нь илүү хүмүүсийн итгэлийг хүлээсэн гэж үзэж байна.

### *1.2.2 Linktr.ee*

2016 онд Алекс Закаряя, Антони Закаряя хэмээх ах дүү хоёр дижитал агентийн үйлчилгээ үзүүлдэг компанитай байсан ба захиалагч талдаа зориулж өөрсдийн сошиал хаягуудаа хаяг тус бүрийнхээ bio хэсэгт оруулах шаардлага гарсан байна. Гэвч нэг үйлдлээ олох дахин давтах нь цагийн гарз нэмээд төвөгтэй санагдсан тул өөрсдийн сошиал хаягууд нь нэг дор байрладаг, удирдах боломжтой платформын санаа гаргаж хөгжүүлж эхэлсэн байна. Өдгөө дэлхий даяар нийт 23 сая идэвхтэй хэрэглэгчтэй болж чадсан бөгөөд 2020 онд Reddit-н удаах шилдэг сошиал вебсайтаар шалгаржээ.

Онцлог нь гэвэл хэрэглэгч тус бүр өөр өөрийн веб хуудас үүсгэх ба түүн дээрээ бүхий л сошиал хаягуудаа сонирхолтой байдлаар жагсааж бусад хүмүүст харуулах боломжтой. Мөн таны хуудсыг нийт хэдэн хүн үзсэн, хэдэн удаа даралт авсан гэх мэт тоо баримтыг танд гаргаж өгч чадна. Интерфейс дизайны хувьд шинэлэг байгаа нь хамгийн том давуу тал гэж үзэж байна.

Уг сайтын хөгжүүлэлтийн технологийн хувьд гэвэл харьцангуй сүүлд гарсан веб учир React дээр суурилсан Next.js фрэймворк болон Node.js ашиглажээ. CSS дээрээ styled-component-г ашигласан нь өөрсдийн онцлог интерфейс дизайнаа бүтээхэд илүү хялбар байсан талаар хөгжүүлэгчид нь дурдсан байна.



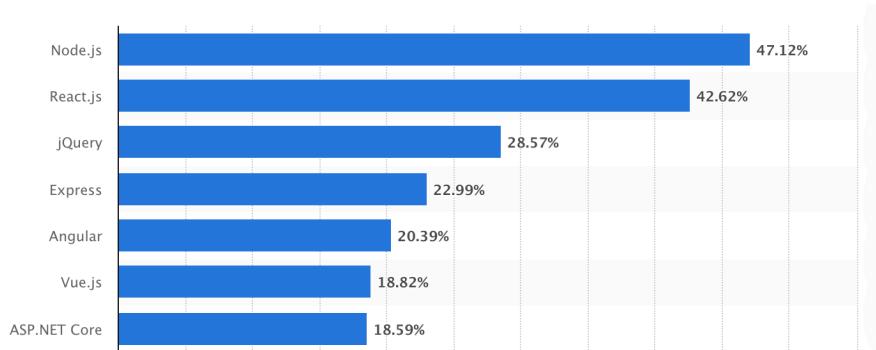
Зураг 1.3: Linktr.ee сайтын харагдах байдал

## 1.3 Ашиглах технологи

### 1.3.1 *React - Javascript* сан

#### Сонгосон шалтгаан

Хэдийгээр олон төрлийн технологи ашиглан уг бүтээгдэхүүнийг хөгжүүлэх боломжтой ч орчин үед хамгийн трэнд болж буй уг технологийн тусгайлан сонгож аван судалж үзлээ. Хамгийн эхлээд 2022 онд хамгийн ихээр ашиглаж буй вэб фрэймворкийн судалгааг<sup>1</sup> танилцуулъя.



Зураг 1.4: 2022 онд хамгийн ихээр ашиглаж буй вэб фрэймворкийн жагсаалт

<sup>1</sup> Вэб фрэймворкийн судалгаа <https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/>

Энэ судалгаанаас та React.js-г вэбийн Front-End хэсэг дээр хамгийн ихээр ашиглаж буйг харж болно.

Мөн Javascript хэл ашиглан хийсэн, хөгжүүлэлт хийхэд хялбар, client болон server-side аппликацийн хийх боломжтой, виртуал DOM гэсэн ойлголтыг оруулж ирснээр бусад сангуудаас илүү хурдан ажиллах боломжийг олгосон нь уг санг сонгосон хамгийн том шалтгаанууд юм.

### Технологийн талаар

Фэйсбуу<sup>2</sup> компани дотооддоо ашиглаж байсан технологио 2013 онд танилцуулсан нь програмчлалын Javascript хэлийг ашиглаж хийсэн Front-end library болох React<sup>2</sup> технологи юм. Declarative UI хөгжүүлэлтийн аргыг хамгийн анх дэлгэрүүлж, өргөн хэрэглээнд нэвтрүүлж чадсан тул Declarative UI-н гол төлөөлөгч гэж явдаг. Уг технологийг ашиглахын тулд үндсэн хэдэн ойлголтууд авах хэрэгтэй. Үүнд component ба түүний lifecycle, javascript-н өргөжүүлсэн хувилбар болох jsx, мөн хамгийн чухал зүйл болох Virtual DOM нар багтана.

Declarative UI гэдэг нь хэрэглэгчийн интерфейсийн кодыг бичихдээ юу зурагдах буюу render хийх үеийн интерфейсийг бүгдийг урьдчилан тодорхойлдог арга барил юм. Imperative програмчлалаас ялгаатай нь хязгаартай нөхцөлд яг юу хийхийг хатуугаар зааж өгөхгүйгээр тухайн state-с хамааруулж хэрэглэгчийн хүссэн зүйлийг гаргаж өгөх боломжтой.

React нь component-based буюу DOM дээр хэвлэж байгаа бүх зүйлс component байна гэсэн дурмийг баримталдаг. Component үүсгэж бичихийн давуу тал нь нэг бичсэн кодоо олон дахин бичигдэхээс зайлсхийж, дахин ашиглах боломжийг олгодог. Тус бур өөрсдийн гэсэн дотоод төлөвтэй мөн гаднаас утга хүлээн авах чадвартай. Үүнийг бид Props гэж нэрлэдэг. Мөн component нь stateless, stateful гэж хоёр хуваагддаг ба stateful component нь өөрийн гэсэн төлөвтэй, түүнийгээ удирддаг, class болон hook ашигласан функцууд байна. React-н давуу тал нь state эсвэл props-н өөрчлөлтийг үргэлж хянаж байдаг тул өөрчлөлт

---

<sup>2</sup>Reactjs official site <https://reactjs.org>

орж ирэхэд бүтэн хуудсыг зурах бус зөвхөн тухайн өөрчлөгдсөн component-г л дахин зурдаг. Ингэснээр энгийн вебүүдээс илүү хурдтай ажилладаг.

```

1  export function Container = ({children}) => {
2
3      return (
4          <div className="container">
5              {children}
6          </div>
7      )
}

```

Код 1.1: JSX ашиглаж 'container' класстай html элемент буцаах

компонент

JSX нь Javascript Extended гэсэн үгний товчлол бөгөөд энгийнээр javascript дотор HTML-н тагуудыг бичиж өгөх мөн кодыг илүү богино болгож хүссэн үр дүндээ хүрэх боломжийг олгодог. Үүний цаана Babel гэсэн transcompiler-г ашиглаж дундын хөрвүүлэлтийг хийдэг ба хэдийгээр HTML таг бичиж байгаа харагддаг ч код дунд цэвэр HTML-г огтоос бичиж өгдөггүй гэсэн үг юм<sup>3</sup>.

### 1.3.2 *Next.js - React дээр суурилсан фрэймворк*

#### **Сонгосон шалтгаан**

Reactjs өөрийг нь ашиглаж төсөл өхлүүлэхэд router, хуудаслалтаас өхлээд олон төрлийн тохиргоо хийх шаардлагатай болдог нь хөгжүүлэлтийн цагийг их үрдэг. Харин Next.js фрэймворк нь тэр бүх тохиргоог нэг л коммандаар хийж, зөвхөн код дээрээ анхаарал хандуулах боломжийг олгодог нь давуу талтай. Тэгэхээр дан Reactjs дээр төслөө өхлүүлсний оронд фрэймворк ашигласан нь хөгжүүлэлтийн хурдад эерэгээр нөлөөлнө гэж үзлээ.

Next.js давуу талуудаас дурьдвал:

- Image Optimization буюу их хэмжээтэй зураг оруулахад автоматаар зургийн чанарыг алдагдалгүйгээр хэмжээг багасгаж өгдөг

---

<sup>3</sup>Дадлагын хугацаандаа хийж байсан технологийн судалгаанаасаа иш татав <https://github.com/ulziibox/intern-report/blob/main/main.pdf>

- Zero config буюу нэг ч тохиргоо хийлгүйгээр төслөө эхлүүлэх боломж
- Static Site Generator болон Server Side Render хийх
- Typescript болон Fast Refresh дэмждэг
- File-system Routing буюу “pages” гэсэн хавтас дотор үүссэн файлуудаас хамаарч вебийн замууд тодорхойлогддог мөн dynamic routing ашиглах боломжтой
- API Routes буюу өөр дээрээ nodejs сервер ашиглаж API endpoint гаргах боломжтой.  
Ингэснээр тусдаа сервер ашиглах шаардлага үүсэхгүй
- SEO буюу хайлтын системийн оновчлолыг SSR ашиглаж тохируулж өгөх гэх мэт маш олон давуу талуудтай

Мөн ердөө ганц “build” командаар статик болон динамик вебийг гарган авч ямар нэгэн веб сервер /apache, nginx гэх мэт/ ашиглалгүйгээр сервер дээрээ шууд байршуулах боломжтой юм.

### Технологийн талаар

Next.js<sup>4</sup> нь React сан дээр суурилж хөгжүүлсэн нээлттэй эхийн фрэймворк бөгөөд Vercel компани 2016 онд албан ёсны танилцуулгаа хийж олон нийтэд зарласан юм. React нь зөвхөн хэрэглэгчийн интерфейсийг зурах үүрэгтэй сан ба бусад веб хөгжүүлэлтэд хэрэгтэй хуудас хооронд шилжих гэх мэт үйлдлийг react-router болон бусад маш олон нэмэлт сангаяас сонголт хийж шийдэх шаардлагатай байсан нь төслийн эхлэх явцыг удаашруулах хандлагатай байдаг. Харин Next.js ашигласнаар нэг ч тохиргоо хийлгүйгээр төслийг эхлүүлж шууд код бичих боломжийг бүрдүүлдэг. Цаана нь хийгдсэн тохиргоо нь нийт веб-сайтуудын 90 хувийн шаардлагыг хангаж чадаг гэж үздэг нь уг фрэймворкын сүүлийн жилүүдэд эрэлттэй болж буй шалтгаануудыг нэг билээ.

---

<sup>4</sup>Next.js official site <https://nextjs.org>

### 1.3.3 *PostgreSQL - Өгөгдлийн сан*

Платформынхоо өгөгдлийн сан дээр ашиглахаар сонгосон PostgreSQL нь 20 жилийн турш идэвхтэйгээр нээлттэй эхээр хөгжүүлэгдэж ирсэн өгөгдлийн сан бөгөөд SQL болон JSON Query дэмждэгээрээ илүү давуу талтай юм. Мөн олон төрлийн ORM дэмжиж ажиллах боломжтой.

Түүхийн хувьд 1986 онд Калифорнийн Их Сургуулийн Компьютерын Ухааны тэнхимд POSTGRES гэдэг нэртэйгээр төсөл нь эхэлжээ. PostgreSQL нь Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) болон Windows үйлдлийн системүүд дээр ажилладгаас гадна Apple, Fujitsu, Red Hat, Cisco, Instagram гэх мэт томоохон компаниуд өөрсдийн зарим бүтээгдэхүүнийхээ технологийн шийдлийг уг өгөгдлийн сан дээр шийдсэн байна.

### 1.3.4 *Prisma - Нээлттэй эхийн ORM*

Prisma нь PostgreSQL, MySQL, SQL Server, SQLite болон MongoDB ашиглаж хурдан хугацаанд, алдаа багатай апп бүтээхэд зориулагдаж гарсан нээлттэй эхийн ORM (Object-relational mapping) юм. Нийт 3 хэсгээс бүрдэх ба үүнд

1. Prisma Client - Node.js болон Typescript хэл ашиглаж query бичих багаж
2. Prisma Migrate - Prisma Cli, Prisma Schema ашиглан migration хийх багаж
3. Prisma Studio - GUI ашиглан өгөгдлийн сангаа харах, засвар боломжтой багажууд багтана.

Уг технологийг сонгох болсон гол шалтгаан нь Next.js дээр back-end хөгжүүлэлтээ хийхэд Prisma ORM нь хамгийн тохиромжтой гэж үзсэнд ба SQL бичихгүйгээр өгөгдлийн сангаа удирдах боломжтой. Энэ нь миний хувьд илүү хурдан хугацаанд хангалттай үр дүнд хүрэх боломжийг өгч байна.

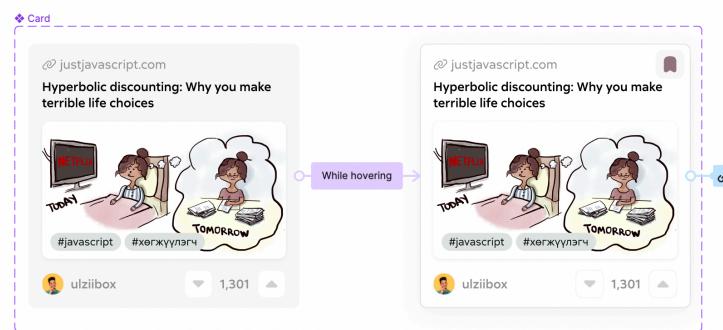


Зураг 1.5: Prisma тойм зураг

### 1.3.5 Figma - интерфейс дизайн, Prototype хувилбар гаргах багаж

Миний хувьд уг платформоо бүтээхдээ хөгжүүлэлтийн үе шатуудыг судлах, түүнийгээ практик дээр хэрэгжүүлэх зорилготой байгаа эхнээс нь бүхий л зүйлсээ чанартайгаар гүйцэтгэхийг оролдож байгаа. Уг үе шатуудад орчин үеийн аргачлалаар хэрэглэгчийн интерфейс дизайн гаргах, түүнийгээ Prototype түвшинд хүргэж туршиж үзэх нь зайлшгүй багтана. Иймд User Experience болон хэрэглэгчийн интерфейсээ гаргахдаа веб дээр сууринсан Figma гэх багажтай танилцан, ашиглаж байна.

Figma нь 2015 онд анх үүсэж байсан бөгөөд веб суурьтай тул ямар ч үйлдлийн систем дээр ажиллах боломжтой. Мөн Vector болон Raster төрлийн зурагтай харьдаг, олон дизайнерууд бодит хугацаанд хамтдаа ашиглах боломжтой, веб болон гар утасны аплын интерфейс хөгжүүлэлтэй ижил системээр (жишээ нь адилхан component гэх ойлголттой) явдаг нь үүнийг сонгох хамгийн том давуу тал болж өгсөн.



Зураг 1.6: Figma ашиглаж Card-н hover эффект дээр prototype хийсэн компонент

Зурсан интерфейсүүдээ хооронд нь холбож хийсвэрээр аппаа ажиллуулан хэрэглэг-

чийн туршилт хийх хэсгийг Prototype гэдэг бөгөөд заавал кодын хэрэгжүүлэлт хийж цаг хугацаа болон мөнгөн зардал гаргалгүйгээр хийж буй аппаа хэрэглэгчээр туршуулах, үр дүнгээ гарган авч түүнийгээ сайжруулах нөхцөлийг уг веб апликешн маань гаргаж өгсөн нь UX/UI дизайнеруудын ашиглах болсон хамгийн том шалтгаануудын нэг юм.

## 2. СИСТЕМИЙН ШААРДЛАГА

Уг бүлэг нь системийн хэрэглэгчийн зүгээс тавигдах шаардлагыг тодорхойлж, тухайн гаргасан шаардлагууд дээрээ үндэслэн UX судалгаа хийсэн талаарх гарах ба хэрэглэгч суурьтай интерфейс дизайн гаргахад тулгарсан асуудлуудыг товч дурдлаа.

### 2.1 Шаардлагын шинжилгээ

#### 2.1.1 Хэрэглэгчид

Интернет сүлжээ ашиглан мэдээлэл авдаг, бусадтай хуваалцдаг бүх төрлийн хэрэглэгчид

#### 2.1.2 Функционал шаардлагууд

Хүснэгт 2.1: Функциональ шаардлага

ФШ 101	Веб нь бусад веб холбоосуудыг дангаар нь болон бүлэглэж оруулах боломжтой байх
ФШ 102	Нийт хэрэглэгчдийн оруулсан веб холбоосууд, хэрэглэгчдийн мэдээллээс түлхүүр үгээр хайлт хийх боломжтой байх
ФШ 103	Хэрэглэгч бүлэглэж оруулсан холбоосуудаа бусад хүмүүстэй хуваалцах боломжтой байх
ФШ 104	Хэрэглэгч платформ дээрх дурын хүнээ дагах, түүний оруулсан веб холбоосуудыг харах боломжтой байх
ФШ 105	Бусад хүмүүсийн оруулсан веб холбоосууд дээр хэрэглэгч үнэлгээ өгдөг боломжтой байх
ФШ 106	Веб нь хэрэглэгчийн дагасан сэдвийн дагуу холбоосуудыг харуулдаг байх
ФШ 107	Веб нь хэрэглэгч бүртгэх боломжтой байх

### 2.1.3 Функционал бус шаардлагууд

Хүснэгт 2.2: Функциональ бус шаардлага

ФБШ 101	Веб нь хэрэглэгч ашиглахад хялбар интерфейстэй байх
ФБШ 102	Интерфейс дизайн нь UI шаардлагын дагуу хөгжүүлэгдсэн байх
ФБШ 103	Веб дээр нийтлэл оруулах үед бусад веб холбоосуудын Open Graph мэдээллийг 500 миллисекундэд багтаан авдаг байх
ФБШ 104	Вебийн эхний хувилбар зөвхөн Desktop төхөөрөмж дээр ажилладаг байх
ФБШ 105	Веб холбоос оруулах үед дээд тал нь хоёр hashtag ашигладаг байх
ФБШ 106	Хэрэглэгч хэдэн ч удаа веб холбоос оруулах боломжтой байх

## 2.2 UX/UI шаардлага

Уг судалгааны ажлын онцлог тал нь шууд хөгжүүлэлтээ хийж эхлэхээс өмнө хэрэглэгч төвтэй дизайн гаргаж түүнийгээ зорилтот хэрэглэгч дээр туршиж, гүйцэтгэл сайтай User Experience болон User Interface дизайн гаргах юм. Ингэснээр сайн бүтээгдэхүүн гаргах том үндэс болох, ирээдүйд гарах хөгжүүлэлтийн зардлыг багасгах давуу талтай.

### 2.2.1 User Experience шаардлага

Хэрэглэгч төвтэй UX/UI дизайн гаргахад дараах үе шатын дагуу ажиллах шаардлагатай.

#### - User Persona гаргах

Манай платформыг ашиглах боломжтой хоёроос дээш хэрэглэгчийг сонгож урьдчилж бэлдсэн асуултуудаас асууж мэдээллийг нэгтгэн тухайн хүнийг тодорхой хэмжээнд дүгнэн, уг платформыг ямар зорилготойгоор ашиглах боломжтой нөхцлүүдийг /Use Case/ гаргана.

#### - Асуудал тодорхойлох

Гаргасан Use Case дээрээ үндэслэж эцсийн хэрэглэгч дээр ямар асуудлууд байгааг су-

далж, хэрхэн шийдэх талаар санаа гарган User Experience-н шаардлагуудыг тодорхойлно.

Энэ нь хэрэглэгч төвтэй дизайн гаргахын үндэс болох тул таамгаар бус судалгаан дээрээ үндэслэж вэбийн хэрэглэгчийн харилцах хэсгийн дизайныг гүйцэтгэнэ.

#### **- Доод түвшинд Prototype хувилбар гаргах**

Дээрх ажлуудыг нэгтгэн User Experience дизайныг доод түвшинд буюу ерөнхий загвартайгаар хийж, ашиглаж буй хэрэгсэл болох Figma дээрээ бүх хуудас, компонентийн логик үйлдлүүдийг холбож бодит ажилладаг мэт загвар гаргана.

#### **- Usability туршилт хийж дизайнаа сайжруулах**

Гаргасан Prototype хувилбараа сонгож авсан хэрэглэгчдээр ашиглиулж UX дээр ямар алдаа байгаа, хэрэглэгчийн асуудлыг шийдвэрлэж чадаж байгаа эсэх, вэб компонентүүдийн байрлал, хоорондын логик үйлдэл зөв байгаа эсэхийг тодорхойлж хэрэглэгчдээс санал хүсэлт авах шаардлагатай. Түүний дараагаас дизайнаа дахин сайжруулж, хөгжүүлэлтийн шатанд ороход бэлэн болгох хэрэгтэй.

### **2.2.2 User Interface дизайны шаардлага**

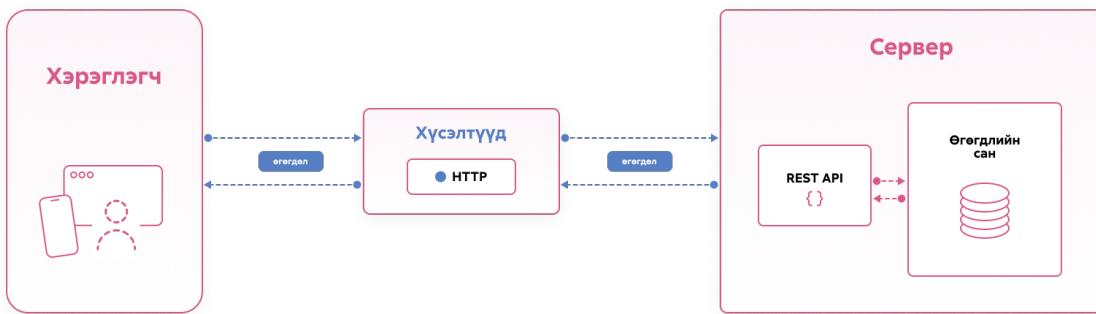
Хүснэгт 2.3: User Interface дизайны шаардлага

ИДШ 101	Шинэлэг дизайнтай байх
ИДШ 102	Гол элементүүд дээр ”D85888”hex кодтой өнгийг ашиглах
ИДШ 103	Компонентуудын micro-interaction ойлгомжтой байх
ИДШ 104	Дэвсгэр өнгө дээр цагаан өнгийг түлхүү ашиглах
ИДШ 105	Contrast буюу өнгөний ялгарлыг бага байлгах
ИДШ 106	Веб фонтын хувьд ”Rubik - Cyrillic Extented”хувилбарыг ашигласан байх
ИДШ 107	Элемент хооронд white-space буюу сул зайд сайн гаргаж өгөх

# 3. СИСТЕМИЙН АРХИТЕКТУР, ЗОХИОМЖ

## 3.1 Системийн архитектур

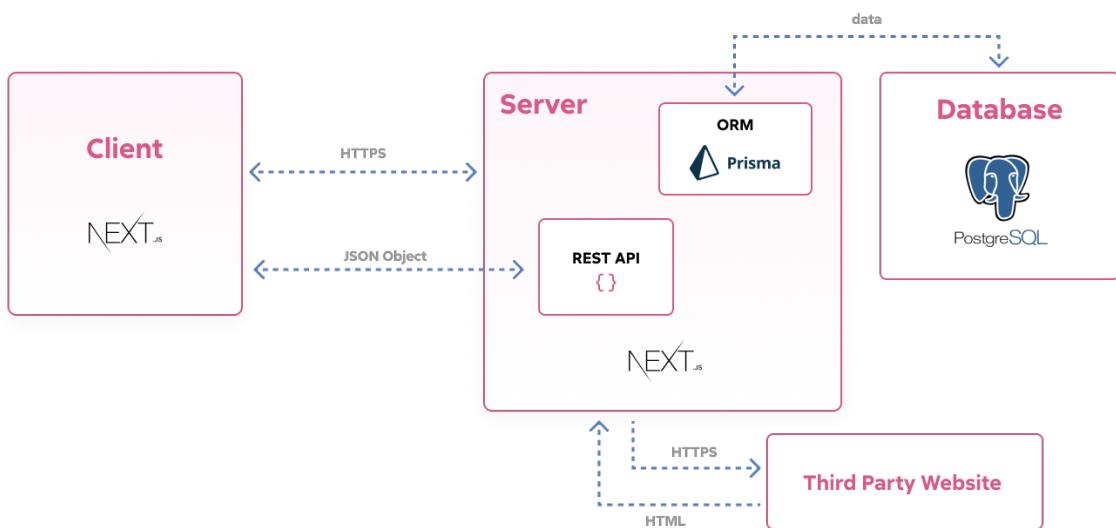
Манай систем нь тийм ч төвөгтэй систем биш мөн энгийн веб хөтчүүдэд зориулж хийгдэх тул системийн архитектурын үлгэр загвар дундаас веб аппд хамгийн өргөн хэрэглэгддэг Хэрэглэгч-Сервер (Client-Server pattern) үлгэр загварыг сонголоо. Хэрэгжүүлэхэд хялбараас гадна Next.js, Prisma ашиглан төслөө хөгжүүлж байгаа учир хамгийн тохиромжтой гэж үзэж байна.



Зураг 3.1: Client-Server Архитектурын үлгэр загварын дүрслэл

Уг архитектур нь хэрэглэгч талаас компьютер, гар утасны веб хөтчийг ашиглаж HTTP холболтоор хүсэлт явуулж, сервер талаас өгөгдлийн санг ашиглан REST API бэлдэн эргээд хэрэглэгч рүү HTTP холболтоор хүсэлтийн хариуг өгөх зарчмаар ажиллана.

### 3.1. СИСТЕМИЙН АРХИТЕКТУРГ 3. СИСТЕМИЙН АРХИТЕКТУР, ЗОХИОМЖ



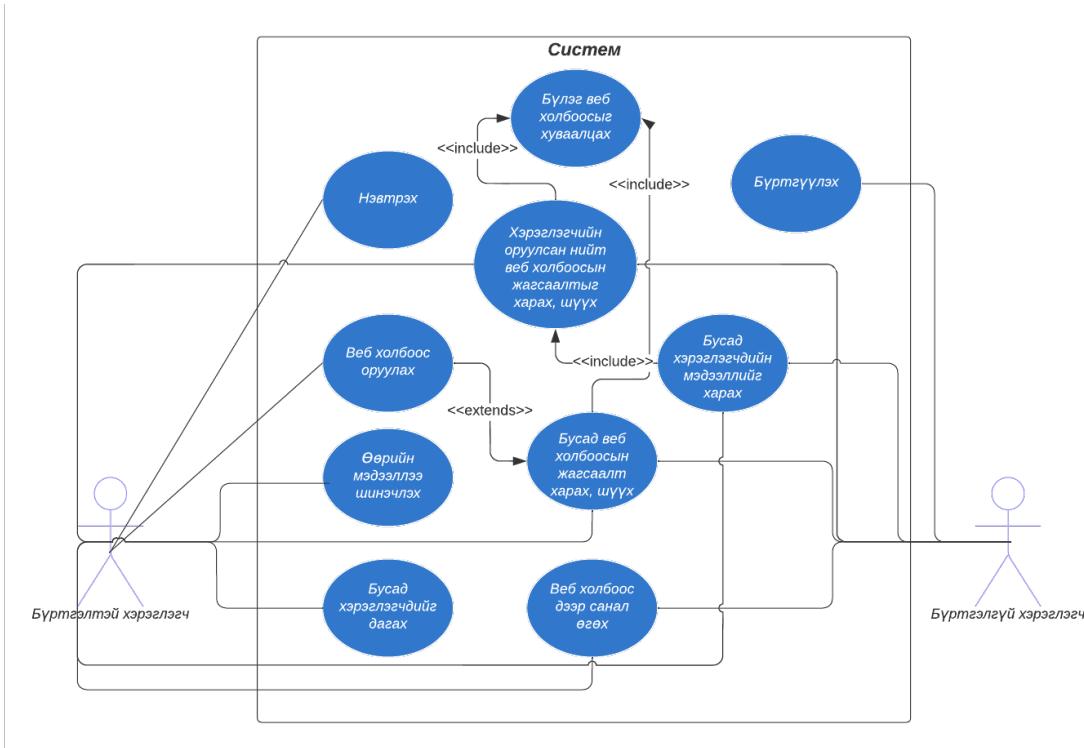
Зураг 3.2: Хэрэгжүүлэх гэж буй системийн архитектур

Front-end болон Back-end хэсгээ Next.js дээрээ хийх буюу fullstack хөгжүүлэлт хийгдэнэ.

- Client талаас сервер рүү интернэт ашиглан HTTPS холболтоор өгөгдлөө дамжуулна
- Сервер дээр Prisma ORM-г ашиглан өгөгдлийн сангаа удирдана
- Хэрэглэгчээс орж ирсэн URL-г авч сервер нь гуравдагч сайтын мэдээллийг авахаар HTTPS холболтоор хүсэлт илгээн хариуд нь тухайн сайтын Open Graph Protocol дахь meta data-г авна.

## 3.2 Ажлын явцын диаграмм

### 3.2.1 Ажлын явцын диаграмм



Зураг 3.3: Ажлын явцын диаграмм

### Ажлын явцын диаграммын тайлбар

- Нэвтрэх** - Бүртгэлтэй хэрэглэгч өөрийн бүртгүүлсэн имэйл хаягаараа нэвтрэх, нууц үгээ сэргээх
- Веб холбоос оруулах** - Өөрийн хүссэн веб холбоосоо дангаар нь болон бүлэглэж оруулах, хэрэв бүлэглэж оруулахаар бол тухайн бүлэг веб холбоосны нэр, тайлбаруудыг заавал авна
- Өөрийн мэдээллээ шинэчлэх** - Өмнөх мэдээлэл хуучирсан эсвэл шинэчлэх шаардлага гарсан тохиолдолд хувийн мэдээлэл, холбооснуудынхаа нууцлалыг шинэчлэх
- Бусад хэрэглэгчийг дагах** - Өөрийн нүүр хуудсаа сонирхлынхoo дагуу хөгжүүлэхийг хүсвэл бусад хэрэглэгчийг дагаж, тэдгээрийн оруулсан холбооснуудыг хамгийн түүхийн тайлбаруудыг заавал авна

### *3.2. АЖЛЫН ЯВЦЫН ДИАГРАММ 3. СИСТЕМИЙН АРХИТЕКТУР, ЗОХИОМЖ*

гийн эхэнд харах

- **Бусад веб холбоосын жагсаалтыг харах** - Систем дээр public байдлаар нийтлэгдсэн бүх веб холбоосыг шууд болон шүүсэн байдлаар харж өөрт хэрэгтэй мэдээллээ авах
- **Веб холбоос дээр санал өгөх** - Бүртгэлтэй болон бүртгэлгүй хэрэглэгчид тухайн веб холбоосны контентыг харсны дараагаар өөрийн хувийн саналаа өгч, чанарын үнэлгээ хийх
- **Бүртгүүлэх** - Хэрэглэгч бүртгэлгүй хэдий ч манай платформыг бүрэн ашиглах боломжтой. Хэрэв өөрөө веб холбоос оруулахыг хүсвэл хувийн мэдээллээ бөглөж бүртгүүлэх
- **Хэрэглэгчийн оруулсан нийт веб холбоосын жагсаалтыг харах** - Зөвхөн нэг хэрэглэгчийн оруулсан private болон public веб холбооснуудыг нэг хуудсан харах, тэдгээрээс шүүх
- **Бусад хэрэглэгчдийн мэдээллийг харах** - Бүртгэлтэй болон бүртгэлгүй хэрэглэгч бусад хэрэглэгчдийн оруулсан хувийн мэдээллийг харах, холбоо барих мэдээллийг олох
- **Бүлэг веб холбоосыг хуваалцах** - Хэрэглэгч нэг сэдвийн хүрээнд олон веб холбоос оруулсан бол платформ дээр тухайн бүлгийн мэдээллийг харуулсан тусдаа нэг хуудас үүснэ. Түүний веб холбоосыг хуулж авах, бусад сошиал сүлжээнд түгээх

### 3.3 UX/UI дизайн

Өмнөх бүлэгт хийсэн UX судалгаан дээрээ үндэслэж User Experience болон User Interface загварыг High Fidelity түвшинд эцэслэж гаргасан ба гаргасан дизайнаа хэрэглэгчээр туршиулж тодорхой үр дүнгүүдэд хүрч чадсан. Үүнд:

- Хэрэглэгчийн шаардлагыг бүрэн ойлгож, хэрэглэгч төвтэй дизайн гаргасан
- Тусдаа бүлэг ажил болгон хийснээр интерфейс загвартая анхаарч шинэлэг, ашиглахад хялбар хэрэглэгчийн интерфейс загвар зурсан
- Хөгжүүлэлтийн шатнаас өмнө бүх хэрэглэгчийн интерфейсүүд дизайн системийн дагуу гарч дууссан тул front-end хөгжүүлэлтийн явцыг ихээр хурдлуулсан
- Интерфейсийн дагуу front-end код явагдах тул back-end код дээр dummy датагаар хөгжүүлж явах, төлөвлөх үе шат хялбар болсон. Front-end хэсэгтэй зөрөх магадлал багассан гэж ойлгож болно.

Одоо UX шаардлагын дагуу ажилласан процессоо богиноор тайлбарлая.

#### 3.3.1 *User Persona тодорхойлж эхний загварыг бүтээсэн нь*

Шаардлагын дагуу хоёр хүнийг сонгон авч тэдгээр хүмүүсээс хэрэгтэй мэдээллүүдээ нэгтгэсэн ба доорхи зургаар илэрхийллээ.

##### **User Persona 1 - Болорчуулун**

**Нэр:** Болорчуулун

**Нас:** 24

**Хүйс:** Эрэгтэй

**Ажил:** Програм хангамжийн инженер

**Байршил:** Монгол улс, Улаанбаатар хот

**Боловсрол:** МУИС, Мэдээллийн технологи -  
2020 он

**Гэр бүл:** Ганц бие

**Илэрхийлэл:** Ганцаараа байх дуртай

Зураг 3.4: Persona 1 - Болорчуулууны мэдээлэл

#### Use Case

- Өглөө ажил дээрээ эхний 20 минут мэргэжилтэйгээ холбоотой нийтлэлүүд унших дуртай
- Веб хөтөч дээрээ таалагдсан эсвэл дараа нь унших ёстой холбоосуудаа bookmark хийж хадгалж авдаг боловч зөвхөн ажлын компьютер дээр л тэдгээр холбоос маань хадгалагддаг. Өөр рүүгээ веб холбоосоо цуглуулж явуулахаас залхуу хүрдэг

### User Persona 2 - Шүрэнцэцгийн мэдээлэл

<b>Нэр:</b> Шүрэнцэцг	<b>Байршил:</b> Монгол улс, Улаанбаатар хот
<b>Нас:</b> 27	<b>Боловсрол:</b> Зохиомж дээд сургууль, Олон улсын сэтгүүлч - 2016
<b>Хүйс:</b> Эмэгтэй	<b>Гэр бүл:</b> Найз залуутай
<b>Ажил:</b> Сэтгүүлч	<b>Илэрхийлэл:</b> Нээлттэй харилцаатай

Зураг 3.5: Persona 2 - Шүрэнцэцг

### Use Case

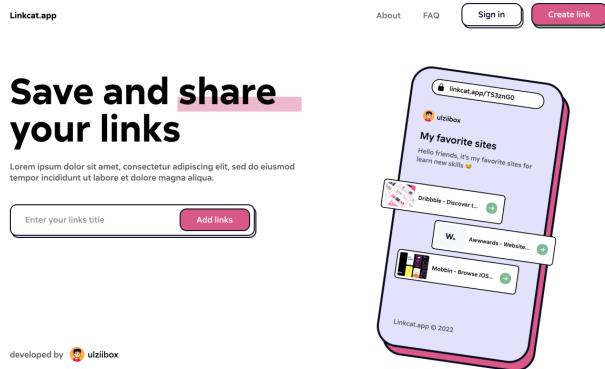
- Миний хувьд мэдээгээ бэлтгэхдээ дийлэнхдээ интернет дэх эх сурвалжуудаас бэлтгэдэг. Нэг асуудал байдаг маань эх сурвалж авдаг хэдхэн вэбсайт дунд л эргэлдэж байгаа.
- Олсон мэдээлүүдээ хамт ажиллаж буй хүмүүс рүүгээ явуулахдаа Telegram ашиглан нэг нэгээр нь хуулж тавин явуулдаг. Хүн болгон руу ингэж явуулах нь надад төвөгтэй байдаг

### Эхний Wireframe загварыг гаргаж Prototype хувилбар бүтээсэн нь

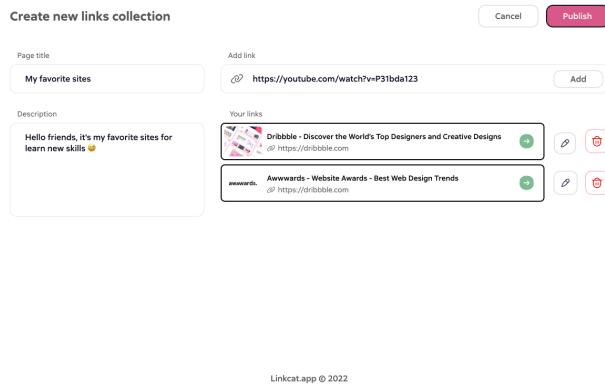
Хамгийн эхэнд нийт 8 хуудсын Wireframe загварыг хэрэглэгчийн Use Case дээрээ тулгуурлан зурж, дизайн дээрх бүх элементүүдээ хооронд нь холбон Prototype<sup>1</sup> хувилбар гаргаж, сонгож авсан хоёр хэрэглэгч дээрээ Usability туршилт хийхэд бэлэн болголоо.

<sup>1</sup>Prototype хувилбарыг туршиж үзэх <https://www.figma.com/proto/EL2nOGmToBRVxHNuZwH661/Draft?>

Нийт зурсан 8 хуудсаас вебийн гол процесийг илэрхийлэх 4 зургийг орууллаа. Бусад хэсгийг хавсралтаас<sup>2</sup> үзэх боломжтой

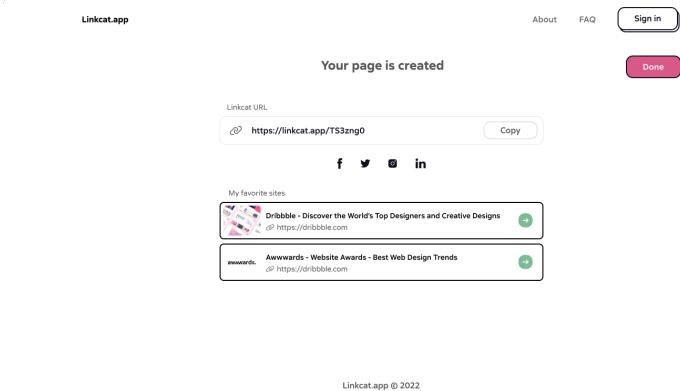


Зураг 3.6: Нүүр хуудас

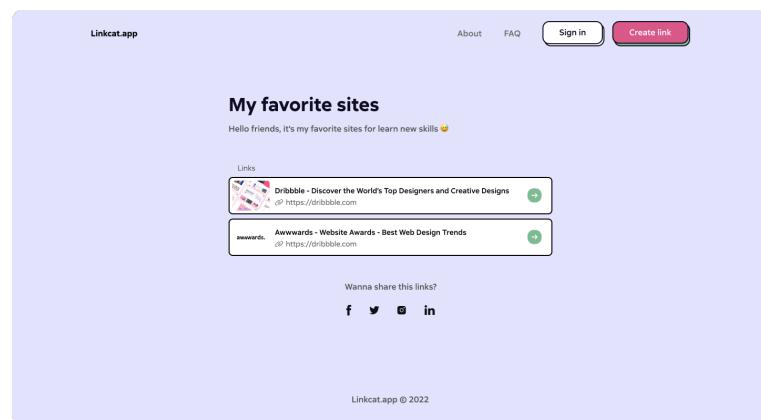


Зураг 3.7: Холбоосуудаа оруулах форм

<sup>2</sup>Зурсан интерфейс дизайн <https://www.figma.com/file/EL2nOGmToBRVxHNuZwH661>



Зураг 3.8: Оруулсан холбоосуудыг нийтлэсний дараах хуудас



Зураг 3.9: Оруулсан нийтлэл бусад хүмүүст харагдах байдал

### **Prototype хувилбар дээрээ Usability туршилт хийсэн нь**

Дараа нь Prototype хувилбар дээрээ Usability буюу хэрэглэхэд тухтай байдлын туршилтыг сонгож авсан хоёр хэрэглэгч дээрээ хийлгэж уг гаргасан дизайн дээрх үүссэн асуудлыг тодорхойлж, хэрэглэгчээс санал хүсэлтийг аван дараагийн гаргах дизайнхаяа ерөнхий шаардлагуудыг тодорхойлов. Үнээс дурдвал:

- UI загварын хувьд концепцийг дахин сайжруулах
- Бусад хүмүүсийн оруулсан холбоосыг нийтэд нь харуулах
- Оруулж буй холбоосыг ангилалтай байлгах
- Платформ дээр бүртгэлтэй бусад хэрэглэгчдийг харуулах

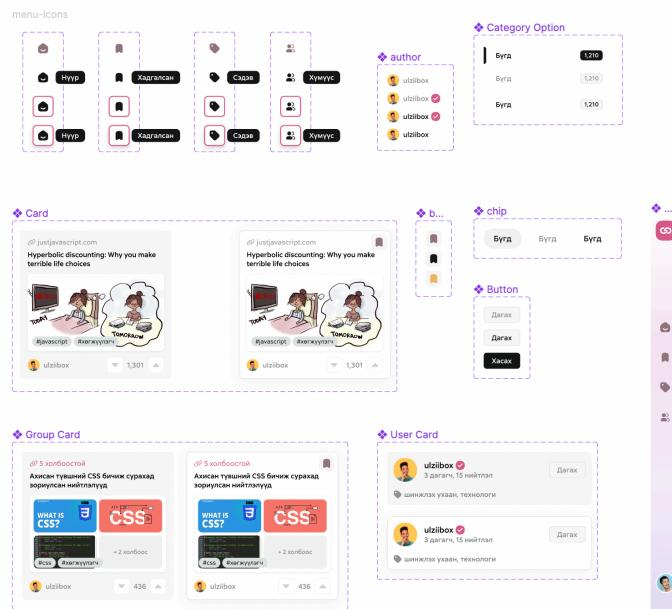
Иймд уг шаардлага дээрээ үндэслэн эцсийн байдлаар вебийнхээ UX/UI дизайнг гаргах шаардлагатай.

### **Redesign буюу зурсан дизайнаа дахин сайжруулсан нь**

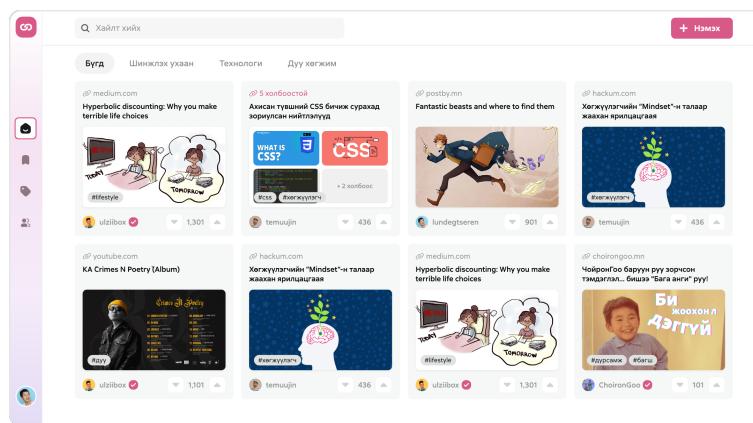
UX/UI гүйцэтгэлийн хамгийн сүүлийн шатанд нийт 12 хуудас дизайныг<sup>3</sup> холбож зурсан ба өмнөх дизайны процесстой адилаар Prototype хувилбар гаргаж эцсийн хэрэглэгчдээрээ Usability туршилтыг хийж дараагийн шат болох код хөгжүүлэлтийн шатыг эхлүүлэхэд бэлэн боллоо. Нийт зурсан хуудсаасаа 6 хуудсыг онцлон харуулж, User Experience дизайнхаяа шийдлийг тайлбарлалаа. Бусад зурсан хуудсыг доор байгаа хавсралтаас харах боломжтой.

---

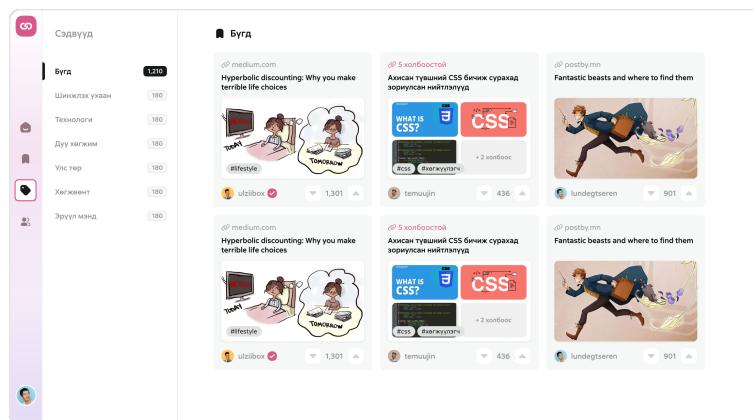
<sup>3</sup>UX/UI дизайнүүдийн хувилбарийн төслийн хувилбар <https://www.figma.com/proto/EL2nOGmT0BRVxHNuZwH661>



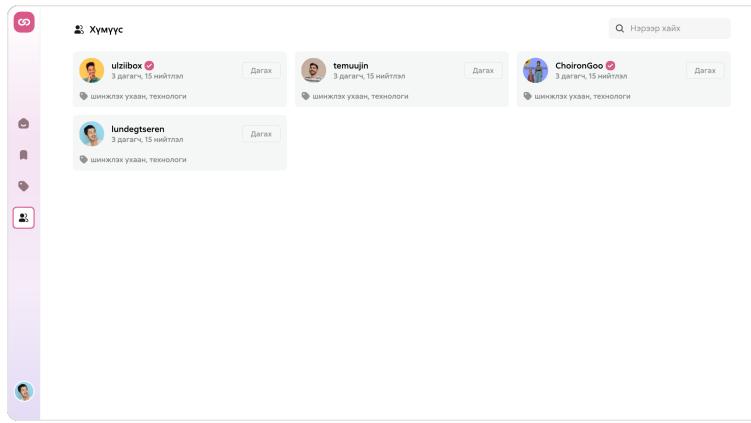
Зураг 3.10: Ашигласан компонентуудын жишээ



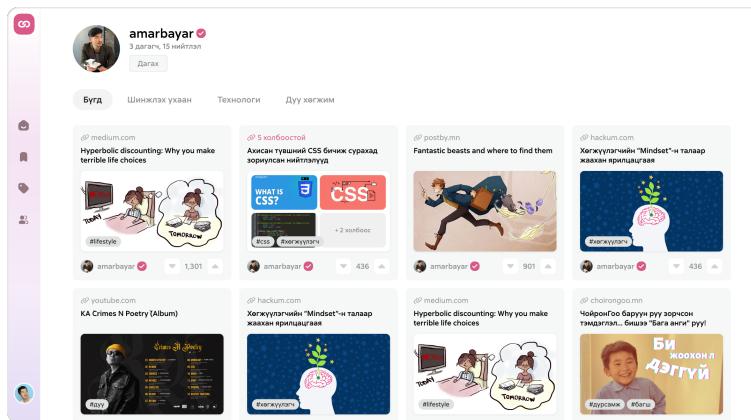
Зураг 3.11: Нэвтэрсний дараах нүүр хуудас



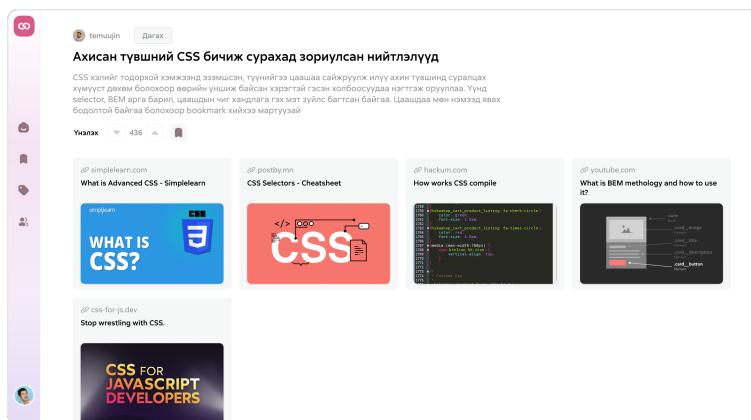
Зураг 3.12: Бүх холбоосуудыг төрлөөр нь шүүж харах хуудас



Зураг 3.13: Хэрэглэгчдийн жагсаалт харагдах хуудас



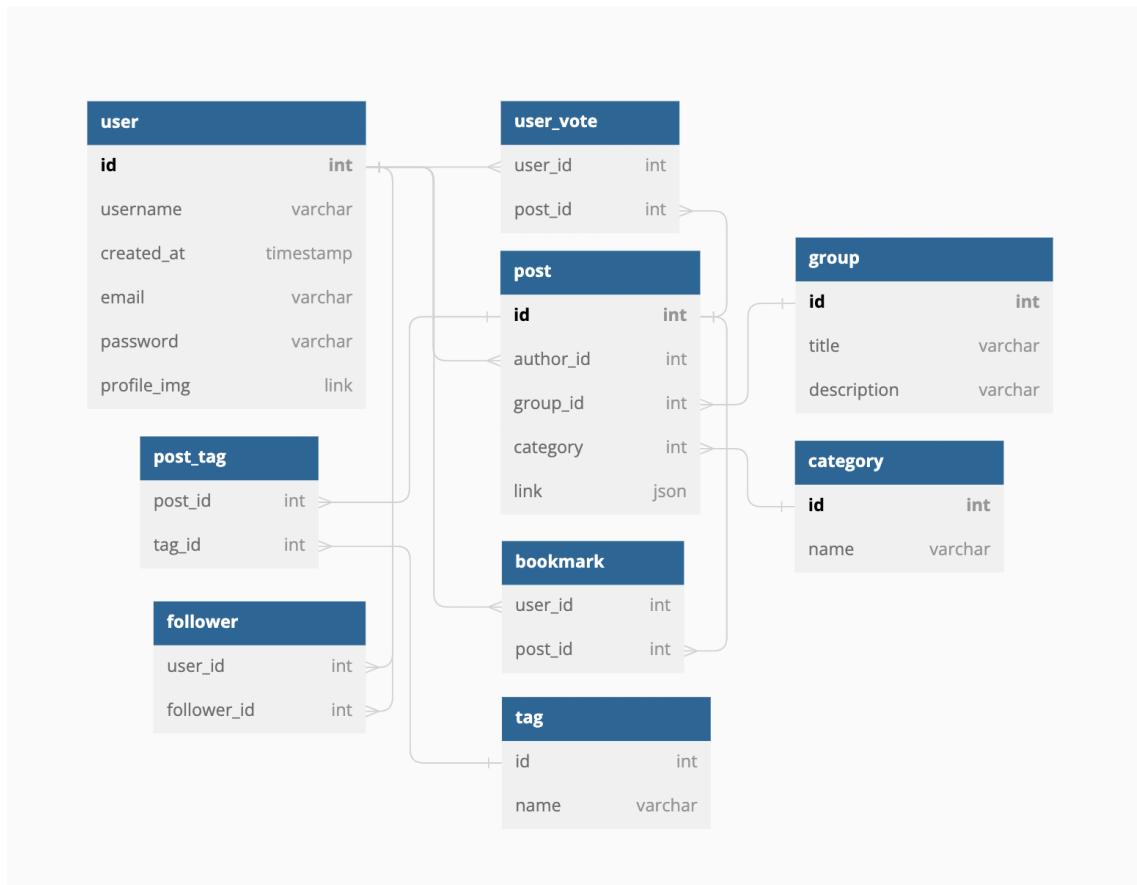
Зураг 3.14: Бусдын профайлыг харах хуудас



Зураг 3.15: Бүлэг холбоос доторх холбоосуудыг харах хуудас

## 3.4 Θгөгдлийн сан

### 3.4.1 Θгөгдлийн сангийн диаграм



Зураг 3.16: Системийн RD Schema

### Θгөгдлийн сангийн хүснэгтүүдийн тайлбар

Хүснэгт 3.1: user хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	int	Хэрэглэгчийн дахин давтагдашгүй ID-г хадгална - Auto Incremented
2	username	varchar	Хэрэглэгчийн гараас оруулж өгсөн нэрийг хадгална. Зөвхөн латин тэмдэгтүүд ашиглах хэрэгтэй
3	created_at	timestamp	Хэрэглэгчийн хаяг үүссэн хугацааг серверээс авч хадгална
4	email	varchar	Хэрэглэгчийн цахим шуудан
5	password	varchar	Хэрэглэгчийн гараас оруулж өгсөн нууц үгийг encrypt-лэж уг хэсэгт хадгална
6	profile_img	link	Хэрэглэгчийн оруулж өгсөн зургийг сервер дээр хадгаж, замыг нь энэ хэсэгт хадгална

Хүснэгт 3.2: post хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	int	Нийтлэлийн дахин давтагдашгүй ID-г хадгална - Auto Incremented
2	author_id	int	Тухайн нийтлэлийг бичсэн хэрэглэгчийн ID-г Foreign Key-р хадгална
3	group_id	int	Хэрэглэгч веб холбоосыг нэг дор олныг оруулах боломжтой ба тухайн тохиолдолд бүлэг веб холбоос гэж үзэн тухайн бүлгийн нэр, тайлбарыг хадгална
4	category	int	Нийтлэлийн төрлийг хадгална. Нийтлэл дор хаяж нэг нийтлэлд харьялагдах шаардлагатай
5	link	json	Нэг болон түүнээс их холбоос хадгалах боломжтой болгож байгаа тул хэрэглэгчийн оруулж өгсөн веб холбоосуудыг уг талбарт json хэлбэрээр хадгална

Хүснэгт 3.3: post\_tag хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	post_id	int	Нэг нийтлэл хэдэн ч tag-тай байж болох ба уг талбарт нийтлэлийн ID-г Foreign Key-р авч ашиглаж байгаа
2	tag_id	int	Хэрэглэгч өөрсдөө Tag-aa үүсгэж өгөх боломжтой учир тухайн үүсгэсэн Tag-н ID-г авна

Хүснэгт 3.4: follower хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	user_id	int	Тухайн хэрэглэгчид хэнийг дагаж байгааг илэрхийлэх талбар
2	follower_id	int	Тухайн хэрэглэгчийг хэн дагаж байгааг илэрхийлэх талбар

Хүснэгт 3.5: user\_vote хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	user_id	int	Хэн тухайн нийтлэл дээр санал өгсныг хадгалах талбар
2	post_id	int	Тухайн хэрэглэгч аль нийтлэл дээр санал өгсныг хадгалах талбар

Хүснэгт 3.6: group хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	int	Primary Key бөгөөд хэрэглэгч олон веб холбоос оруулж өгөх боломжтой. Үүнийг бид бүлэг холбоос гэж нэрлэж байгаа ба энэ тохиолдолд тухайн бүлэгт заавал нэр болон тайлбар утга байх хэрэгтэй
2	title	varchar	Бүлэг холбоосын гарчгийг хадгалах талбар
3	description	varchar	Бүлэн холбоосын тайлбарыг хадгалах талбар

Хүснэгт 3.7: category хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	int	Primary Key бөгөөд хэрэглэгч нийтлэлийн төрөл олон байх боломжтой.
2	name	varchar	Төрлийн нэрийг хадгалах талбар

Хүснэгт 3.8: bookmark хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	user_id	int	Хэрэглэгч өөрт таалагдсан нийтлэлээ хадгалах шаардлагатай ба уг талбарт тухайн нийтлэлийг хадгалсан хэрэглэгчийн ID-г Foreign Key-p хадгална
2	post_id	int	Хэрэглэгчийн хадгалсан нийтлэлийн ID-г хадгална

Хүснэгт 3.9: tag хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	int	Хэрэглэгчийн гараас оруулж өгсөн tag-н ID-г хадгална - Auto Incremented
2	name	int	Хэрэглэгчийн гараас оруулж өгсөн tag-н нэрийг уг талбарт хадгална

# 4. ХЭРЭГЖҮҮЛЭЛТ

Энэхүү бүлэгт өмнө нь гарсан интерфэйс дизайн, системийн архитектур болон диаграммыг хэрэгжүүлж хэрхэн бүтээгдэхүүн болгон гаргасан талаараа бичлээ. Хэрэгжүүлэлт хийх үе шатаа ерөнхийд нь

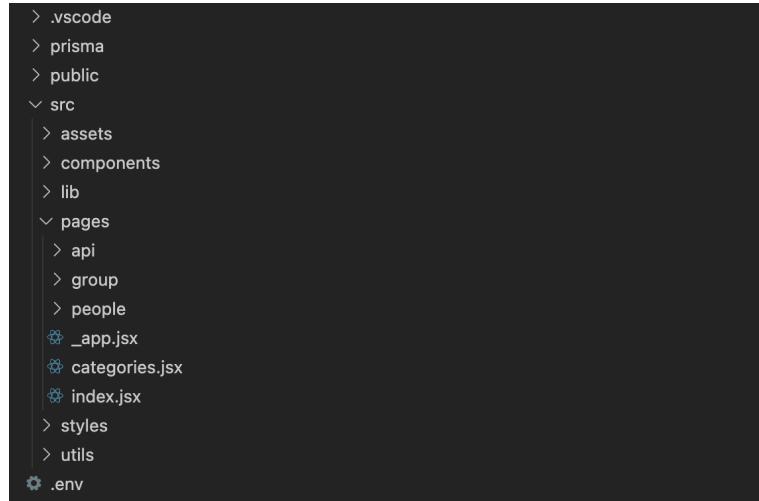
- Хөгжүүлэлтийн орчноо бэлдэх
- Front-end болон Back-end хөгжүүлэлт
- Серверт байршуулах /Deploy/

гэсэн алхмуудад хувааж гүйцэтгэсэн ба доор ажлуудаасаа гол гэсэн зүйлсээ нэгтгэн оруулав.

## 4.1 Хөгжүүлэлтийн орчныг бэлдэх

Ашиглах технологийн хувьд Next.js, Prisma ORM, PostgreSQL ашиглах учир хамгийн эхлээд өгөгдлийн сангаа Local хост дээрээ бэлдэж, дараа нь Next.js фрэймворкоо суулган шаардлагатай тохиргоонуудыг хийнэ. Үүний дараагаас Prisma ORM-ээ суулгаж, урьдчилж бэлдсэн Next.js төсөл дээрээ орууллаа.

Мөн кодын хувилбараа хадгалж, хөгжүүлэлтийн үе шатаа цэгтэй авч явахын тулд Version Control дээрээ Git, Github.com-г сонгож харагдацтай repository үүсгэв.



Зураг 4.1: Төслийн файлын бүтэц

Next.js дээр файлын бүтцээ Зураг 4.1 дээр харагдаж байгаачлан бүтэцлэж хөгжүүлэлтээ эхлэхэд бэлэн болголоо. Хөгжүүлэлтийн гол зорилго маань Fullstack хөгжүүлэлт хийх учир манай платформын back-end, front-end хэсгүүд нэг repository дотор хамт явагдана. Файлын бүтцээ тайлбарлавал

- **prisma** хавтаст цаашдаа ашиглах моделүүд бичигдсэн schema байршина
- **public** хавтаст төслийн хүрээнд ашиглах статик файлууд /зураг, icon/ байна
- **src** хавтаст төслийн бүх компонент, гол эх кодууд байршина
- **src/pages/api** хавтаст төслийн back-end хэсгийн код бичигдэнэ

## 4.2 Код хөгжүүлэлт

Хөгжүүлэлтийн хувьд маш олон компонент, хуудсууд, тэдгээрийн код хийгдсэн тул зөвхөн нүүр хуудас буюу хэрэглэгчийн дагаж буй хүмүүсийн оруулсан холбоосууд харагддаг хэсгийг онцолж авч эхнээс нь эхлээд бүтэн ажиллах үе шатуудыг тайлбарлав.

### 4.2.1 Гаргасан интерфэйсийнхээ дагуу хэрэглэгчид харагдах хэсгийг өрсөн нь

Front-end хөгжүүлэлт дээр урьдчилж интерфэйс дизайныг гаргасан нь вебийн бүх компонент, тэдгээр хаана байрших гээд олон зүйлийг тодорхой болгож өгсөн нь давуу тал

болж өгсөн. Иймд хамгийн эхлээд бэлдсэн хөгжүүлэлтийн орчин дээрээ үндсэн хуудсууд болон дотор нь ашиглаж буй компонентуудыг статик байдлаар өрөв.

```

1  export const Card = ({
2    href,
3    type,
4    site_url,
5    og_title,
6    og_image,
7    author_name,
8    author_profile,
9    vote_count,
10   tags,
11 }) => {
12   return (
13     <Link href={href} target="_blank">
14       <div className="cursor-pointer space-y-2 rounded-lg border border-
15         gray200 bg-gray300 p-3.5 transition-all duration-300 hover:border-
16         gray100 hover:bg-white hover:shadow-sm">
17         <div className="flex items-center gap-1 text-sm font-normal text-
18           description">
19           <LinkIcon />
20           {site_url}
21         </div>
22         ...
23
24         <div className="order-last flex items-center gap-1 text-sm text-
25           description">
26           <VoteBtn type="up" />
27           <VoteBtn type="down" />
28         </div>
29       </div>
30     </Link>
31   )
32 }

```

```
29 );
30 }
```

Код 4.1: Хэрэглэгчийн оруулсан холбоосыг харагдах компонент

#### 4.2.2 Prisma ORM ашиглах өгөгдлийн сангагаа бэлдэх

Хамгийн түрүүнд өгөгдлийн сангийн схемийнхээ дагуу **prisma/schema.prisma** файл дээр өгөгдлийн сан дээр үүсгэх хүснэгт, багануудын бүх нөхцөл, холбоосуудыг бичиж хэрэглэгчийн оруулсан нийтлэлийн өгөгдлийг хадгалах **Post** гэсэн хүснэгтийг үүсгэнэ.

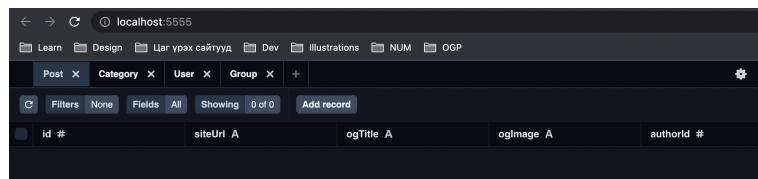
```
1 generator client {
2   provider = "prisma-client-js"
3 }
4
5 datasource db {
6   provider = "postgresql"
7   url      = env("DATABASE_URL")
8 }
9
10 model Post {
11   id       Int      @id @default(autoincrement())
12   siteUrl String
13   ogTitle String
14   ogImage String
15   authorId Int
16   author   User    @relation(fields: [authorId], references: [id],
17                                onDelete: Cascade)
17   groupId Int?
18   group    Group? @relation(fields: [groupId], references: [id])
19   tags     Tag[]
20   categories Category[]
21   createdAt DateTime @default(now())
22   updatedAt DateTime @updatedAt
23 }
```

24

25

Код 4.2: Өгөгдлийн сангийн хүснэгтийг Prisma ашиглан үүсгэх

Уг файлыг **migration** хийсний дараа бичсэн моделийн дагуу өгөгдлийн сан дээр **Post** хүснэгт үүссэн.



Зураг 4.2: Post хүснэгтийн харагдац

#### *4.2.3 Хэрэглэгчийн оруулсан холбоосын жагсаалтыг харах End point гаргах*

Back-end хэсгийн кодыг түрүүн хэлсэнчлэн шууд Next.js фрэймворк дээрээ [/api/posts/index.js](#) гэсэн файл үүсгэж Prisma Client сангаас шаардлагад нийцсэн функцуудийг хэрэглэж хэ-рэгтэй өгөгдлийг тодорхой нөхцлүүдээр авч хариуг буцаана.

```

16             res.status(400).json(e)
17         }
18     } else {
19         console.log(e)
20         res.status(500).json({message: "something wrong try
21             again some time later"})
22     }
23
24     try {
25         const posts = await getPostsByUsers(following.map(follow=>{
26             return follow.following
27         }))
28
29         const curPosts = posts.map(p => {
30             return {
31                 id: p.id,
32                 ogImage: p.ogImage,
33                 ogTitle: p.ogTitle,
34                 ogUrl: p.ogUrl,
35                 groupId: p.groupId,
36                 group: p.group,
37                 tags: p.tags,
38                 authorId: p.authorId,
39                 author: p.author,
40                 categories: p.categories[0].id || ''
41             }
42         })
43
44         res.json({list:curPosts,total:posts.length})
45
46     } catch (e) {
47         if (e instanceof Prisma.PrismaClientKnownRequestError) {
48             if (e.code === 'P2002') {

```

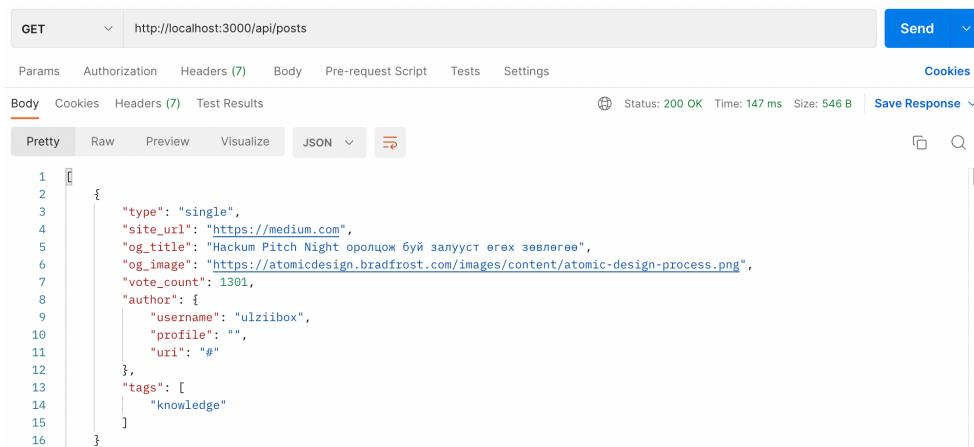
```

49             res.json({list:[],total:0})
50
51         } else{
52
53             res.status(400).json(e)
54
55         } else {
56
57             console.log(e)
58
59             res.status(500).json({message: "something wrong try
60                 again some time later"})
61
62
63 ...

```

Код 4.3: Next.js дээрээ end point гаргах

Уг кодны үр дүнд дараах JSON object үүссэнийг Postman ашиглаж GET хүсэлт тавин харууллаа.



Зураг 4.3: Нийтлэлийн жагсаалтыг агуулсан JSON Object

#### 4.2.4 Front-end хэсэг дээр гаргасан End point-оо ашиглаж бодит өгөгдлийг харуулах

Одоо Front-end хэсэг дээрээ нийтлэлийн жагсаалтуудыг авч нүүр хуудаст зурж харуулах шаардлагатай. Үүний тулд axios хүсэлтийг ашиглаж өгөгдлөө авч өмнө нь үүсгэсэн **Card** компонентдоо шаардлагатай **props**-уудыг дамжуулж, нийтлэлийг **map** функцийг ашиглан харуулав

```

1
2 export async function getServerSideProps() {
3   const { data: categories } = await axios.get(
4     `${process.env.API_URL}/categories`
5   );
6   const { data: posts } = await axios.get(
7     `${process.env.API_URL}/posts/following`
8   );
9   return {
10     props: {
11       categories: categories || [],
12       posts: posts || [],
13     },
14   };
15 }
```

Код 4.4: ServerSideProps болон axios ашиглан хүсэлт илгээж

өгөгдлөө татаж авах

```

1
2 ...
3 const router = useRouter();
4
5 useEffect(() => {
6
7   setCategories([
8     {
9       id: 0,
10      name: "Бүгд",
```

```

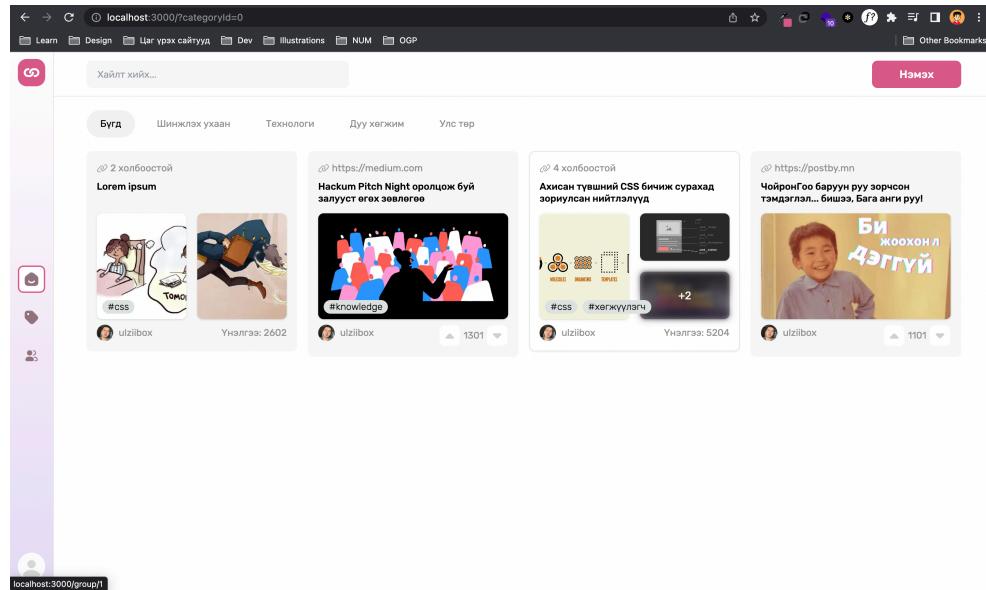
11     },
12     ...CATEGORIES,
13   ]);
14 }, [];
15
16 ...
17
18 {posts.map((item) => {
19   if (item.type === "single") {
20     return (
21       <Card
22         key={item.post.id}
23         site_url={item.post.site_url}
24         og_title={item.post.og_title}
25         og_image={item.post.og_image}
26         author_name={item.post.author.username}
27         author_profile={item.post.author.profile_img}
28         vote_count={item.post.vote_count}
29         tags={item.post.tags}
30         href={item.post.href}
31       />
32     );
33   }
34 })
35
36 ...

```

Код 4.5: Card компонентод ирсэн утгуудыг props-оор дамжуулж DOM  
дээр рендерлэх

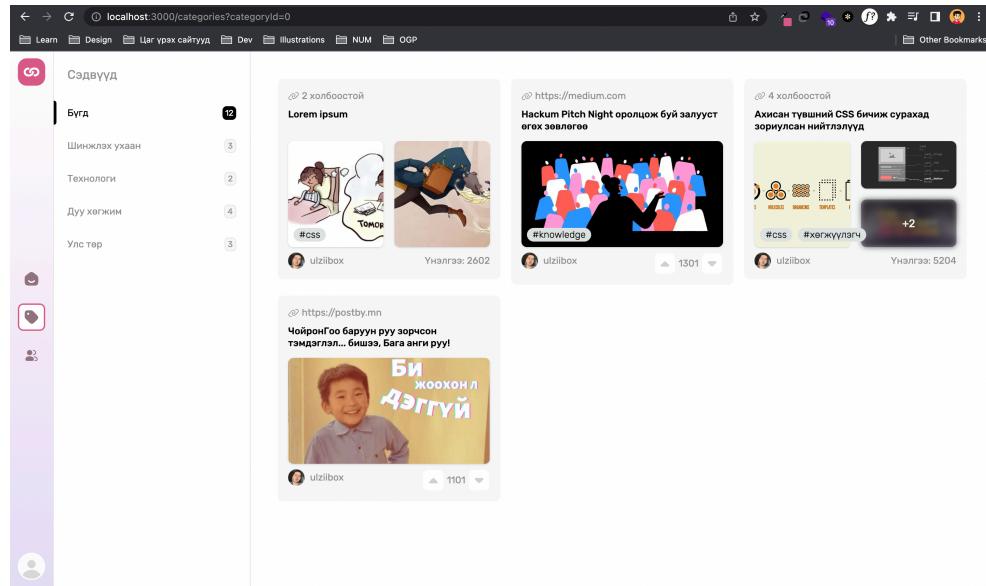
### 4.3 Yp дүн

Дээрх үйлдлийд нь манай төслийг хэрхэн ажиллаж буйг тоймлон харуулсан ба үр дүнд  
нь хэрэглэгчийн дагаж буй хүмүүсийн оруулсан холбоосууд нүүр хуудас дээр харагдана.

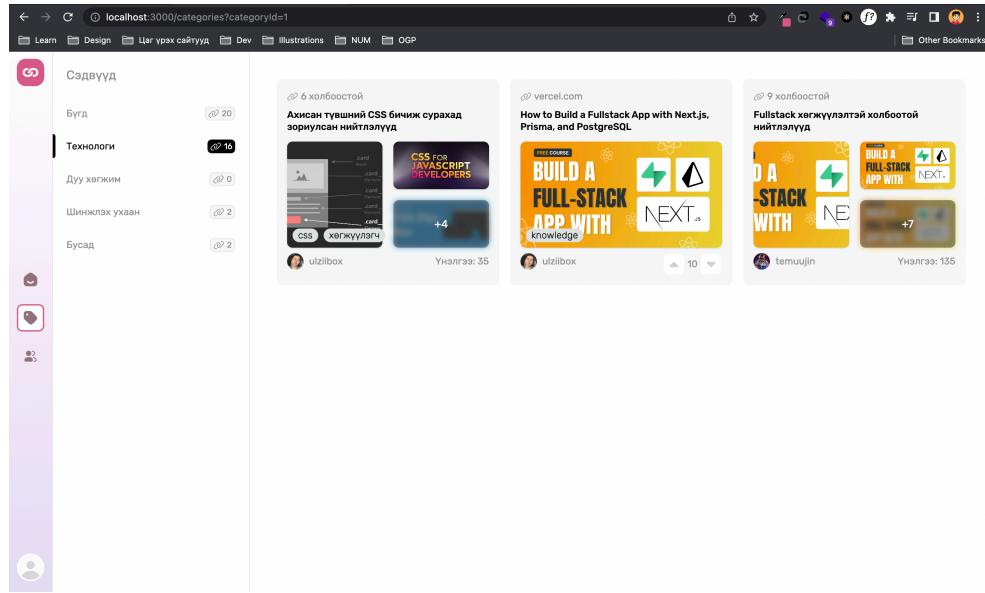


Зураг 4.4: Нүүр хуудас

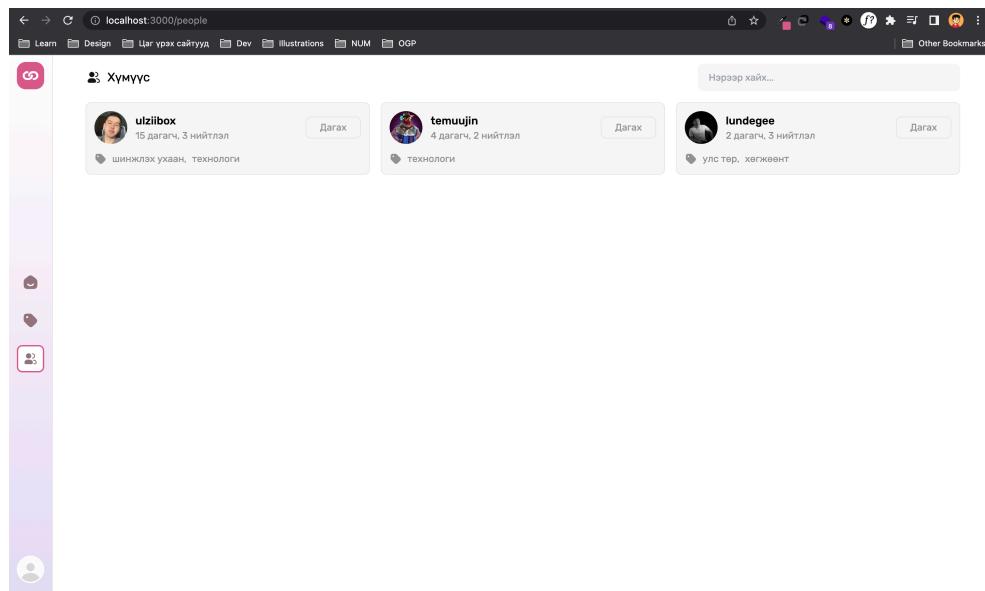
Мөн цаашид дээрх байдлаар шаардлагын дагуу гарсан интерфэйсүүдийг хийж дуусгах ба платформынхoo зарим зургаас оруулав.



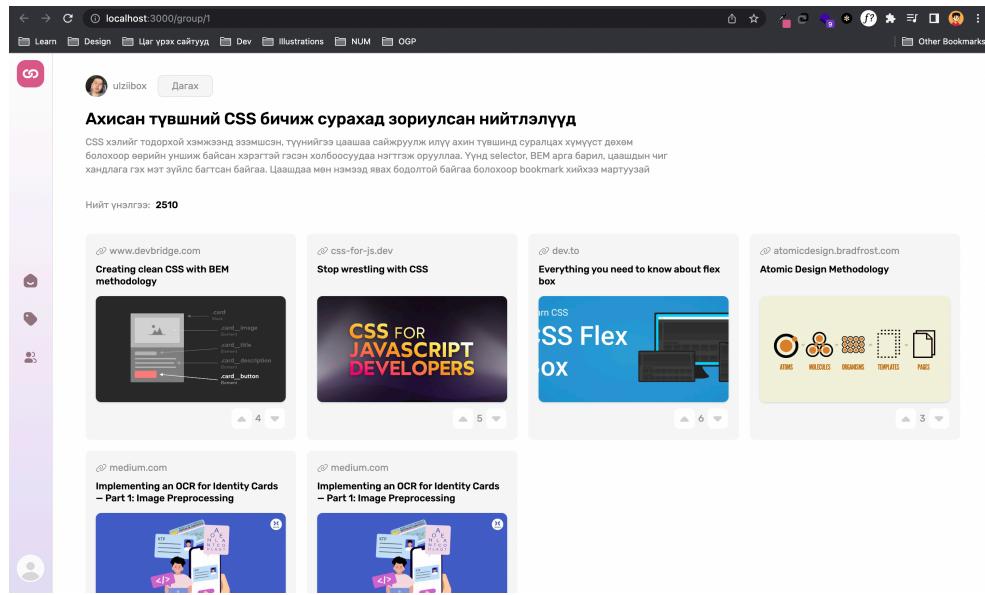
Зураг 4.5: Платформ дээрх сэдвүүдийн жагсаалт болон бүх нийтлэлийг харах



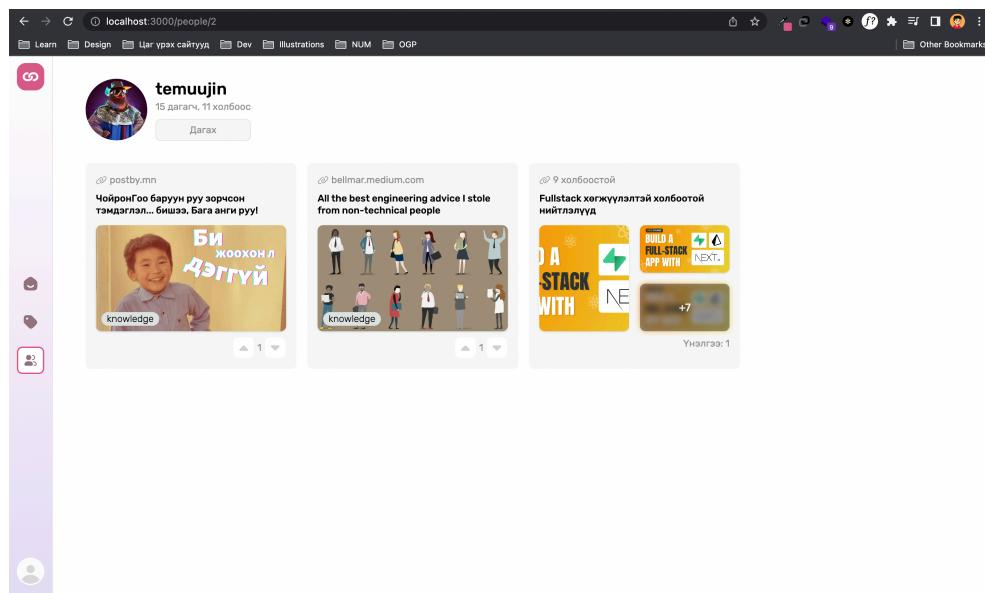
Зураг 4.6: Холбоосуудыг сэдвээр нь шүүж харах



Зураг 4.7: Платформ дээр бүртгүүлсэн хэрэглэгчдийн жагсаалтыг харах



Зураг 4.8: Олон холбоосыг бүлэглэж харуулж буй хуудас



Зураг 4.9: Сонгосон хэрэглэгчийн оруулсан холбоосуудыг нэг дор харах

## 5. ДҮГНЭЛТ

Энэхүү судалгааны ажлаар орчин үеийн дижитал бүтээгдэхүүний хөгжүүлэлтийн арга барилтай танилцаж хэрэглэгчийн шаардлагыг тодорхойлоо ос эхлээд UX судалгаа хийж хэрэглэгч төвтэй интерфэйс дизайн гаргах, шинэлэг технологиудыг ашиглан хөгжүүлэлтийг гүйцэтгэж, өөрийн санаагаа бүрэн ажиллагаатай, хүмүүсийн асуудлыг шийдвэрлэж чадсан сошиал платформ болгож чадсан нь өөрт маань их хэмжээний мэдлэг, туршлага болж үлдсэн гэдэгт итгэлтэй байна.

Мөн цаашлаад уг сошиал платформыг эхний түвшинд Монгол Улсын Их Сургуулийн SISI системийн OAuth-г нэвтрүүлж багш, оюутнууд хоорондоо мэдээллийн эх сурвалжаа хялбар байдлаар хуваалцах, хичээлээс гадуур нэмэлт эх сурвалжийг олох явдлыг хялбар-шуулах боломжтой туслах систем болгон хөгжүүлэх бүрэн боломжтой гэж үзэж байна.

# Ашигласан материал

[1] The Open Graph Protocol Design Decisions

<https://www.scribd.com/doc/30715288/The-Open-Graph-Protocol-Design-Decisions>

[2] Michele Riva - *Real-World Next.js*, (2020)

[3] Pethuru Raj, Anupama Raman, Harihara Subramanian - *Architectural Patterns*, (2017)

[4] Prisma - Documentation

<https://www.prisma.io/docs>

[5] B.Bat-Ulzii - *Reactjs - Intern Report*, (2021) SEAS, NUM.

<https://github.com/ulziibox/intern-report>

# А. PRISMA МОДЕЛ

```
1 generator client {
2   provider = "prisma-client-js"
3   previewFeatures = ["interactiveTransactions"]
4 }
5
6 datasource db {
7   provider = "postgresql"
8   url      = env("DATABASE_URL")
9   shadowDatabaseUrl = env("SHADOW_DATABASE_URL")
10 }
11
12 model User {
13   id        Int    @id @default(autoincrement())
14   username  String
15   email     String
16   password  String
17   profileImg String
18
19   followers Follows[] @relation("following")
20   following Follows[] @relation("follower")
21   votes Votes[] @relation("userVotes")
22
23   posts Post[]
24   createdAt DateTime @default(now())
25   updatedAt DateTime @updatedAt
26 }
27
28 model Post {
29   id        Int    @id @default(autoincrement())
30   ogUrl    String
31   ogTitle  String
32   ogImage  String
33   authorId Int
34   author    User   @relation(fields: [authorId], references: [id],
35                             onDelete: Cascade)
35   groupId  Int?
36   group    Group? @relation(fields: [groupId], references: [id])
37
38   votes Votes[] @relation("postVotes")
39
40   tags     Tag[]
41   categories Category[]
42
43   createdAt DateTime @default(now())
44   updatedAt DateTime @updatedAt
45 }
46
47 model Tag {
48   id    Int    @id @default(autoincrement())
49   name  String
50
51   posts Post[]
52
53   createdAt DateTime @default(now())
54   updatedAt DateTime @updatedAt
55 }
56
57 model Category {
58   id    Int    @id @default(autoincrement())
```

```

59     name String
60
61     posts Post[]
62
63     createdAt DateTime @default(now())
64     updatedAt DateTime @updatedAt
65   }
66
67 model Group {
68     id          Int      @id @default(autoincrement())
69     title       String
70     description String
71
72     createdAt DateTime @default(now())
73     updatedAt DateTime @updatedAt
74     Post       Post[]
75   }
76
77 model Follows {
78     follower   User @relation("follower", fields: [followerId], references
79                   : [id])
80     followerId Int
81     following   User @relation("following", fields: [followingId],
82                           references: [id])
83     followingId Int
84
85     @@id([followerId, followingId])
86
87     createdAt DateTime @default(now())
88   }
89
90 model Votes {
91     id          Int      @id @default(autoincrement())
92     user        User @relation("userVotes", fields: [userId], references: [id])
93     userId     Int
94     post       Post @relation("postVotes", fields: [postId], references: [id])
95     postId     Int
96     vote       String
97
98     createdAt DateTime @default(now())
99     updatedAt DateTime @updatedAt
100    deletedAt DateTime?
101  }

```

## B. PRISMA CLIENT АШИГЛАХ

```
1 import { PrismaClient } from '@prisma/client'
2
3 let prisma: PrismaClient
4
5 if (process.env.NODE_ENV === 'production') {
6   prisma = new PrismaClient()
7 } else {
8   if (!global.prisma) {
9     global.prisma = new PrismaClient()
10   }
11   prisma = global.prisma
12 }
13
14 export default prisma
```

# С. NEXT.JS ДЭЭР REMOTE ДОМАЙН ЗӨВШӨӨРӨХ

```
1  @type {import('next').NextConfig}
2  const nextConfig = {
3    reactStrictMode: true,
4    images: {
5      remotePatterns: [
6        {
7          protocol: "https",
8          hostname: "**",
9        },
10       {
11         protocol: "http",
12         hostname: "**",
13       },
14     ],
15   },
16 };
17
18 module.exports = nextConfig;
```

## D. TAILWIND CUSTOM-CLASS YYCTГЭХ

```
1  @tailwind base;
2  @tailwind components;
3  @tailwind utilities;
4
5  @layer components {
6    .sidebar-icon {
7      @apply flex h-10 w-10 cursor-pointer items-center justify-center
8          rounded-lg transition-all duration-75 hover:border-2 hover:border-primary;
9    }
10
11   .sidebar-icon-active {
12     @apply flex h-10 w-10 cursor-pointer items-center justify-center
13         rounded-lg border-2 border-primary bg-white hover:bg-gray100;
14   }
15
16   .tooltip {
17     @apply absolute left-14 z-10 m-2 w-auto min-w-max origin-left scale-0
18         rounded-md bg-gray-900 p-2 text-xs font-bold text-white shadow-md
19         transition-all duration-100;
20   }
21
22   .vote-btn {
23     @apply grid h-8 w-8 cursor-pointer place-content-center rounded-lg
24         border border-gray200 bg-white hover:border-gray100;
25   }
26
27   .bar {
28     @apply z-0 my-1 flex justify-between px-5 py-3 text-sm text-
29         description transition-all duration-100 hover:text-black;
30   }
31
32   .chip {
33     @apply flex h-5 items-center rounded-md border border-gray100 bg-
34         gray300 px-1 text-xs;
35   }
36
37   .bar-active {
38     @apply relative z-0 my-1 flex justify-between px-5 py-3 text-sm
39         transition-all duration-100 hover:text-black;
40   }
41
42   .chip-active {
43     @apply flex h-5 items-center rounded-md bg-black px-1 text-xs text-
44         white;
45   }
46 }
```

# Е. ХЭРЭГЛЭГЧИЙН МЭДЭЭЛЛИЙГ ГАРГАХ ENDPOINT

```
1 import { NextApiRequest, NextApiResponse } from "next";
2 import prisma from "../../../../../lib/prisma";
3
4 export default async function handle(req: NextApiRequest, res:
5   NextApiResponse) {
6   try {
7     const uId = parseInt(req.query.id as string, 10)
8
9     const resultDB = await prisma.user.findUnique({
10       where: {
11         id: uId
12       },
13       select: {
14         id: true,
15         username: true,
16         profileImg: true,
17         posts: {
18           select: {
19             id: true,
20             ogUrl: true,
21             ogTitle: true,
22             ogImage: true,
23             tags: true,
24             group: true,
25           },
26         }
27       }
28     })
29
30     res.status(200).json(resultDB);
31   } catch (err) {
32     res.status(403).json({ err: "Error occurred" })
33   }
34 }
```

# F. ХОЛБООС ДЭЭР VOTE ӨГӨХ ENDPOINT

```
1 import {NextApiRequest, NextApiResponse} from "next";
2 import {Prisma} from "@prisma/client";
3 import {instanceOf} from "prop-types";
4 import prisma from "../../../../../lib/prisma";
5
6
7 export default async function handle(req: NextApiRequest, res:
8     NextApiResponse) {
9     const requestMethod = req.method
10    switch (requestMethod) {
11        case 'GET':
12            let following;
13            try{
14                let postId = parseInt(req.query.id as string, 10)
15                const totalVote = await VotesTotal(postId)
16                res.status(200).json({totalVote})
17            } catch (e) {
18                if (e instanceof Prisma.PrismaClientKnownRequestError) {
19                    if (e.code === 'P2002') {
20                        res.json({totalVote:0})
21                    } else{
22                        res.status(400).json(e)
23                    }
24                } else {
25                    console.log(e)
26                    res.status(500).json({message: "something wrong try
27                        again some time later"})
28                }
29            }
30            break
31        case 'POST':
32            if (req.body.vote != 'up' && req.body.vote != 'down') {
33                res.status(400)
34            }
35            let postId = parseInt(req.query.id as string, 10)
36            let userId = 1
37            let vote = req.body.vote
38            await Vote(userId, postId, vote)
39            const totalVote = await VotesTotal(postId)
40            res.status(200).json({totalVote})
41            break
42
43        default: res.status(404)
44    }
45
46    async function Vote(userId:number, postId:number, vote:string) {
47        return await prisma.$transaction(async (tx) => {
48            try {
49                const haveVotes = await tx.votes.findFirst({
50                    where:{
51                        userId,
52                        postId,
53                        deletedAt:null
54                    },
55                    orderBy:{
56                        id:'desc'
57                    }
58                })
59            }
60        })
61    }
62}
```

```

58 |     if (haveVotes?.vote==vote) {
59 |         return await tx.votes.updateMany({
60 |             where:{  

61 |                 userId,  

62 |                 postId,  

63 |                 deletedAt:null  

64 |             },  

65 |             data:{  

66 |                 deletedAt: new Date()  

67 |             }  

68 |         })
69 |     }  

70 |     else{  

71 |         await tx.votes.updateMany({  

72 |             where:{  

73 |                 userId,  

74 |                 postId,  

75 |                 deletedAt:null  

76 |             },  

77 |             data:{  

78 |                 deletedAt: new Date()  

79 |             }  

80 |         })
81 |         return await tx.votes.create({  

82 |             data:{  

83 |                 user:{  

84 |                     connect:{  

85 |                         id: userId  

86 |                     }  

87 |                 },  

88 |                 post: {  

89 |                     connect:{  

90 |                         id: postId  

91 |                     }  

92 |                 },  

93 |                 deletedAt: null,  

94 |                 vote,  

95 |             }  

96 |         })
97 |     }  

98 |     catch (e) {  

99 |         if (e instanceof Prisma.PrismaClientKnownRequestError && e.  

100 |             code == 'P2002') {  

101 |             return await tx.votes.create({  

102 |                 data:{  

103 |                     user:{  

104 |                         connect:{  

105 |                             id: userId  

106 |                         }  

107 |                     },  

108 |                     post: {  

109 |                         connect:{  

110 |                             id: postId  

111 |                         }  

112 |                     },  

113 |                     deletedAt: null,  

114 |                     vote,  

115 |                 }  

116 |             })
117 |         }  

118 |         else{  

119 |             throw e
}
return 1

```

```

120 |     })
121 |
122 |
123 | export function VotesTotal(postId:number) {
124 |
125 |     return prisma.$transaction(async (tx) => {
126 |         const totalUp = await tx.votes.count({
127 |             where: {
128 |                 postId,
129 |                 deletedAt:null,
130 |                 vote: 'up'
131 |             }
132 |         })
133 |
134 |         const totalDown = await tx.votes.count({
135 |             where: {
136 |                 postId,
137 |                 deletedAt:null,
138 |                 vote: 'down'
139 |             }
140 |         })
141 |
142 |         return totalUp-totalDown
143 |     })
144 |
145 |
146 |     async function getLastVote(userID, postID) {
147 |         const lastVote = prisma.votes.findFirst({
148 |             where: {
149 |                 userId: userID,
150 |                 post: postID,
151 |                 deletedAt:null
152 |             }
153 |         })
154 |         return lastVote
155 |     }
156 |
157 |     async function DeletePreviousVotes(userId, postId) {
158 |         const res= await prisma.votes.updateMany({
159 |             where: {
160 |                 userId,
161 |                 postId
162 |             },
163 |             data: {
164 |                 deletedAt: new Date()
165 |             }
166 |         })
167 |         return res
168 |     }
169 |
170 |
171 |     async function HaveVotes(userID, postID) {
172 |         try {
173 |             const lastVote = await getLastVote(userID,postID)
174 |         }catch (e) {
175 |             if (e instanceof Prisma.PrismaClientKnownRequestError && e.code
176 | === 'P2002') {
177 |                 return false
178 |             } else {
179 |                 throw e
180 |             }
181 |         }

```

# G. LAYOUT КОМПОНЕНТИЙГ APP КОМПОНЕНТ ДЭЭР АШИГЛАХ

```
1 import "../styles/globals.css";
2 import "../styles/main.css";
3 import { Layout } from "../components/Layout";
4 import { Rubik } from "@next/font/google";
5
6 const rubik = Rubik({
7   subsets: ["cyrillic-ext", "cyrillic", "latin", "latin-ext"],
8   variable: "--font-rubik",
9 });
10
11 function MyApp({ Component, pageProps }) {
12   return (
13     <main className={`${rubik.variable} font-sans`} >
14       <Layout>
15         <Component {...pageProps} />
16       </Layout>
17     </main>
18   );
19 }
20
21 export default MyApp;
```