

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

Лувсандагвын Давааням

Өрсөлдөөнт түргэн бичилтийн вэб апп
(Competitive fast typing web app)

Програм Хангамж (D 061302)
Бакалаврын судалгааны ажил

Улаанбаатар

2023 он

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

Өрсөлдөөнт түргэн бичилтийн вэб апп
(Competitive fast typing web app)

Програм Хангамж (D 061302)
Бакалаврын судалгааны ажил

Удирдагч: _____ Мастер. Р.Жавхлан

Гүйцэтгэсэн: _____ Л. Давааням (20B1NUM0182)

Улаанбаатар

2023 он

Зохиогчийн баталгаа

Миний бие Лувсандагвын Давааням ”Өрсөлдөөнт түргэн бичилтийн вэб апп” 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	2
Ач холбогдол	2
БҮЛГҮҮД	3
1. СЭДВИЙН ЕРӨНХИЙ СУДАЛГАА	3
1.1 Үндсэн ойлголтууд	3
1.2 Ижил төсөөтэй систем	4
1.3 Ашиглах технологи	6
2. СИСТЕМИЙН ШААРДЛАГА	10
2.1 Шаардлагын шинжилгээ	10
2.2 UX/UI шаардлага	11
3. СИСТЕМИЙН АРХИТЕКТУР, ЗОХИОМЖ	13
3.1 Системийн архитектур	13
3.2 Ажлын явцын диаграм	15
3.3 UX/UI дизайн	17
3.4 Өгөгдлийн сан	24
4. ХЭРЭГЖҮҮЛЭЛТ	29
4.1 Хөгжүүлэлтийн орчныг бэлдэх	29
4.2 Код хөгжүүлэлт	30
4.3 Yp дүн	37

5. ДҮГНЭЛТ	41
НОМ ЗҮЙ	42
ХАВСРАЛТ	43
A. PRISMA МОДЕЛ	43
B. PRISMA CLIENT АШИГЛАХ	45
C. NEXT.JS ДЭЭР REMOTE ДОМАЙН ЗӨВШӨӨРӨХ	46
D. TAILWIND CUSTOM-CLASS YYCГЭХ	47
E. ХЭРЭГЛЭГЧИЙН МЭДЭЭЛЛИЙГ ГАРГАХ ENDPOINT	48
F. ХОЛБООС ДЭЭР VOTE ӨГӨХ ENDPOINT	49
G. LAYOUT КОМПОНЕНТИЙГ APP КОМПОНЕНТ ДЭЭР АШИГЛАХ	52

Зургийн жагсаалт

Зураг	Хуудас
1.1 typingtest.com/pro/ сайтын харагдах байдал	4
1.2 play.typeracer.com сайтын харагдах байдал	6
1.3 Figma ашиглаж хийсэн компонент	7
1.4 Firebase Authentication	9
3.1 Client-Server Архитектурын үлгэр загварын дүрслэл	13
3.2 Хэрэгжүүлэх гэж буй системийн архитектур	14
3.3 Ажлын явцын диаграм	15
3.4 Persona 1 - Болорчуууны мэдээлэл	17
3.5 Persona 2 - Шүрэнцэцг	18
3.6 Нүүр хуудас	19
3.7 Холбоосуудаа оруулах форм	19
3.8 Оруулсан холбоосуудыг нийтлэсний дараах хуудас	20
3.9 Оруулсан нийтлэл бусад хүмүүст харагдах байдал	20
3.10 Ашигласан компонентуудын жишээ	22
3.11 Нэвтэрсний дараах нүүр хуудас	22
3.12 Бүх холбоосуудыг төрлөөр нь шүүж харах хуудас	22
3.13 Хэрэглэгчдийн жагсаалт харагдах хуудас	23
3.14 Бусдын профайлыг харах хуудас	23
3.15 Бүлэг холбоос доторх холбоосуудыг харах хуудас	23
3.16 Системийн RD Schema	24
4.1 Төслийн файлын бүтэц	30
4.2 Post хүснэгтийн харагдац	33
4.3 Нийтлэлийн жагсаалтыг агуулсан JSON Object	35
4.4 Нүүр хуудас	38

ЗУРГИЙН ЖАГСААЛТ

ЗУРГИЙН ЖАГСААЛТ

4.5	Платформ дээрх сэдвүүдийн жагсаалт болон бүх нийтлэлийг харах .	38
4.6	Холбоосуудыг сэдвээр нь шүүж харах	39
4.7	Платформ дээр бүртгүүлсэн хэрэглэгчдийн жагсаалтыг харах	39
4.8	Олон холбоосыг бүлэглэж харуулж буй хуудас	40
4.9	Сонгосон хэрэглэгчийн оруулсан холбоосуудыг нэг дор харах	40

Хүснэгтийн жагсаалт

2.1	Функциональ шаардлага	10
2.2	Функциональ бус шаардлага	11
2.3	User Interface дизайны шаардлага	12
3.1	user хүснэгт	25
3.2	post хүснэгт	26
3.3	post_tag хүснэгт	26
3.4	follower хүснэгт	27
3.5	user_vote хүснэгт	27
3.6	group хүснэгт	27
3.7	category хүснэгт	28
3.8	bookmark хүснэгт	28
3.9	tag хүснэгт	28

Кодын жагсаалт

4.1	Хэрэглэгчийн оруулсан холбоосыг харагдах компонент	31
4.2	Өгөгдлийн сангийн хүснэгтийг Prisma ашиглан үүсгэх	32
4.3	Next.js дээрээ end point гаргах	33
4.4	ServerSideProps болон axios ашиглан хүсэлт илгээж өгөгдлөө татаж авах . .	36
4.5	Card компонентод ирсэн утгуудыг props-оор дамжуулж DOM дээр рендерлэх	36

УДИРТГАЛ

Орчин үеийн өртөнц дижитал харилцаа холбоог хөгжүүлж, бичих хурд, нарийвчлал нь янз бүрийн мэргэжлээр ажилладаг хүмүүст зайлшгүй шаардлагатай ур чадвар болж байна. "Duck Racer" нь бичих чадвараа сайжруулах, сурх үйл явцыг тоглоом болгох зорилготой вэбэд суурилсан программ юм. Энэхүү систем нь өрсөлдөөнт уралдааны механизмыг нэгтгэсэн бөгөөд оролцогчид өгөгдсөн догол мөрүүдийг бичиж бодит цагт тоглогчдын эсрэг уралдана. Хэрэглэгчдэд сонирхолтой интерфэйс, бодит цагийн гүйцэтгэлийн хэмжигдэхүүн, текстийн хэсгээс бүрдсэн олон төрлийн мэдээллийг цогцоор нь хослуулж, бичих урлагийг эзэмших өвөрмөц аргыг хэрэглэгчдэд санал болгож байна.

Зорилго

Шивэх дасгалыг үр дүнтэй, тааламжтай болгох системийг тоглоомын аргаар сургалтын үйл явцад нэгтгэж бичих дасгалыг сонирхолтой, хэрэглэгчидийг өрсөлдөх чадвартай болгохыг зорьж байна.

Зорилт

Уг веб аплыг хөгжүүлэхдээ дараах үе шатын дагуу ажиллана.

1. Техникийн болон хэрэглэгчийн шаардлагыг тодорхойлох
2. Хэрэглэгчдийн анхаарлыг татахуйц хэрэглэгчдэд ээлтэй интерфэйсийг дизайн гаргах
3. Ашиглах технологийг онол болон практик дээр суурилж судлах
4. Системийн архитектурын бүтэц, дизайныг зохион байгуулж бэлдэх
5. Гаргасан баримт бичгийн дагуу системийн хөгжүүлэлтээ хийх
6. Бэлэн болсон системд домейн нэр авж, хост хийн байршуулах

Сэдэв сонгох үндэслэл

Энэхүү дипломын ажил нь бичих чадварыг сайжруулах өвөрмөц боловсролын систем бөгөөд. Энэ сэдвийг сонгосон нь хэд хэдэн чухал шалтгаанаас үүдэлтэй:

1. Монголын хүн амын гуравны нэгээс илүү хувь нь 24-өөс доош насныхан байдаг бөгөөд, тоглоомын платформоор энэ залуу хүн амыг татан оролцуулах нь тэднийг дижитал эринд илүү сайн бэлтгэж чадна.
2. COVID-19 тахал гэх мэт нөхцөл байдлаас үүдэн хурдассан онлайн боловсрол руу дэлхий даяар шилжиж байгаа нь бичих чадвар сайтай байх шаардлагатайг улам то-дотгосон.¹
3. Орчин үеийн ажлын байрууд салбараас үл хамааран харилцаа холбоо, хамтын ажиллагаа, бичиг баримт бүрдүүлэхэд дижитал хэрэгслүүд ихээхэн ашигладаг. Бичих ур чадвар нь бүтээмжийг дээшлүүлж, алдааг багасгаж, ажлын урсгалыг илүү хялбар болгоход хувь нэмэр оруулна.²
4. Вэб дээр суурилсан платформ нь дэлхий даяарх хэрэглэгчдийг холбох боломжтой. Энэхүү дэлхийн холболт нь соёлын солилцоог дэмжиж, эрүүл өрсөлдөөнийг дэмжиж, суралцах нийгэмлэгийг бий болгож чадна.³

Ач холбогдол

Уг системийг бүтээснээр тоглоомын сорилтуудаар дамжуулан хэрэглэгчид бичих хурд, нарийвчлалыг цаг хугацааны явцад сайжруулахад түлхэц болох. Сургууль, боловсролын байгууллагууд хичээлд нэмэлт хэрэгсэл болгон ашиглах. Оюутнууд дадлага хийж, ахиц дэвшлээ хянаж, ангийнхантайгаа хөгжилтэй, интерактив байдлаар өрсөлдөж, сургалтын үйл явцыг илүү сонирхолтой байх боломжуудыг бүрдүүлэх юм.

¹Боловсролын талаарх ЮНЕСКО-гийн тайлан: <https://www.unesco.org/en/education>

²LinkedIn ажлын зах зээлийн албан ёсны тайлан: <https://economicgraph.linkedin.com/resources>

³Pew судалгааны төв: <https://www.pewresearch.org/>

1. СЭДВИЙН ЕРӨНХИЙ СУДАЛГАА

Сэдвийн хүрээнд өмнө нь ажиллаж үзээгүй олон шинэ технологиудыг судалж хэд хэдэн шинэ технологи ашиглаж, тэдгээрийг үнэлж, харьцуулав. Энэ бүлэгт би судалгаанаасаа сонгосон технологиудыг танилцуулж, вэб платформыг бий болгох үндсэн ойлголтуудыг танилцуулж байна.

1.1 Үндсэн ойлголтууд

Энэхүү судалгааг амжилттай дуусгахын тулд зарим үндсэн ойлголтуудыг ойлгох нь зайлшгүй чухал юм.

1.1.1 *Web Sockets*

WebSocket нь хэрэглэгчийн хөтөч болон серверийн хооронд хоёр талын интерактив харилцааны session нээх боломжтой дэвшилтэт технологи юм. Энэхүү WebSocket - ийн тусlamжтайгаар сервер рүү мэдээлэл илгээж, серверээс хариулт авах шаардлагагүйгээр үйл явдалд тулгуурласан хариултуудыг хүлээн авах боломжтой.

- Real-time Interaction: Хэрэглэгчдийг бодит цаг хугацаанд бие биетэйгээ уралдуулахыг хүсвэл WebSockets маш чухал. Тэд оролцогч бүрийн мэдээллийг бусад бүх оролцогчдын дэлгэцэн дээр нэн даруй шинэчлэх боломжийг олгоно.
- Dynamic Updates: Хэрэв тоглоомын орчин, дүрэм, зард ямар нэгэн өөрчлөлт орсон бол WebSockets нь бүх идэвхтэй хэрэглэгчдэд шууд мэдэгдэх боломжтой
- Multiplayer Racing: Олон оролцогчтой уралдааны хувьд тоглоомын төлөвийг удирдаж, хэрэглэгчдэд синхрончлолыг хангах нь WebSockets-ийн тусlamжтай илүү удирдах боломжтой болно.

WebSockets нь бодит цагийн интерактив платформын салшгүй технологийн нэг бөгөөд шуурхай шинэчлэлт, харилцан үйлчлэлд шаардагдах хурд, үр ашгийг санал болгодогоороо

1.2. ИЖИЛ ТӨСӨӨТЭЙ СИСТЕМ БҮЛЭГ 1. СЭДВИЙН ЕРӨНХИЙ СУДАЛГАА

давуу талтай.

1.1.2 Web-based Educational Platforms

Вэб дээр суурилсан платформууд нь хэрэглэгчид хүссэн үедээ, хаанаас ч, ихэвчлэн өөрийн хүссэн хэмжээгээр систем руу хандах боломжтойгоороо давуу талтай.

Өрсөлдөх чадвартай эсвэл хамтран ажиллах боломжуудыг агуулж, суралцахыг нийгмийн туршлагыг бий болгож хэрэглэгчид дэлхийн хэмжээнд үе тэнгийнхэнтэйгээ өрсөлдөх эсвэл тэднээс суралцах боломжтой.

1.2 Ижил төсөөтэй систем

1.2.1 TypingTest.com

”TypingTest Pro”нь хэрэглэгчид бичих хурд, нарийвчлалыг хэмжихэд зориулагдсан онлайн платформ юм. Энэ платформ нь хэрэглэгчдэд бодит цагийн горимд бичих янз бүрийн текстүүдийг өгч, гүйцэтгэлийн талаар шууд санал хүсэлтийг санал болгодог.

The screenshot shows the TypingTest Pro interface. At the top, there's a navigation bar with five tabs: 'Test Info' (with a checkmark), 'Personal Info' (with a checkmark), 'Practice' (highlighted with a blue circle and the number 3), 'Typing Test' (with a blue circle and the number 4), and 'Test Summary' (with a blue circle and the number 5). On the left, a vertical sidebar has the text 'Practice Test'. The main content area displays a timer at 0:55 and a text passage: "We know sharks are dangerous but do you know that there is a more dangerous animal we can find on a farm? While sharks kill an average of 5 people a year, cows kill an average of 33 people every year." Below the text is a text input field containing "We know sharks". At the bottom of the main area, there's a 'SKIP PRACTICE' button. At the very bottom of the page, there are links for 'Support', 'Admin login', and '© Typing Master Inc. 2023'.

Зураг 1.1: [typingtest.com/pro/](https://www.typingtest.com/pro/) сайтын харагдах байдал

1.2. ИЖИЛ ТӨСӨӨТЭЙ СИСТЕМ БҮЛЭГ 1. СЭДВИЙН ЕРӨНХИЙ СУДАЛГАА

Манай бүтээх гэж байгаа вебээс ялгаатай тал нь

- 1-ээс 10 минутын хугацаатай туршилтуудыг санал болгож, хэрэглэгчдэд сорилтын түвшингээ сонгох боломжийг олгодог.
- Шууд guest хэлбэрээр орон тест өгөх боломжгүй

Frontend: HTML5, CSS3 болон динамик хэрэглэгчийн интерфэйсүүдэд зориулсан React.js-тэй JavaScript. Backend: Express.js хүрээтэй Node.js нь өргөтгөх боломжтой, үр ашигтай сервер талын програмыг хангадаг. Өгөгдлийн сан: Хэрэглэгчийн профайл, тестийн үр дүн, текстийн хэсгүүдийг хадгалах MongoDB. WebSockets: Бодит цагийн өрсөлдөөн, шууд санал хүсэлтийн функцэд зориулагдсан. Third-party Integrations: Дэлхий даяар тэргүүлэгчдийн самбар, олон нийтийн мэдээллийн хэрэгслээр хуваалцах боломжууд.

Онлайнаар шивэх тестийн олон платформууд байдаг ч ”TypingTest Pro” нь энгийн хэрэглэгчид болон бичих чадвараа сайжруулахад нухацтай ханддаг хүмүүст зориулсан иж бүрэн функцээрээ бусдаас ялгардаг.

1.2.2 play.typeracer.com

TypeRacer бол олон тоглогчийн онлайн хөтөч дээр суурилсан шивэх тоглоом юм. Тоглогчид богино хэсгийг аль болох хурдан шивэх замаар өрсөлддөг бөгөөд тоглоом нь хурдлыг минут тутамд үгээр (WPM) болон нарийвчлалыг хэмждэг. Үүсгэн байгуулагдсан цагаасаа эхлэн энэ нь өрсөлдөөнтэй, хөгжилтэй орчинд бичих чадвараа сайжруулахыг хүсч буй хүмүүсийн дуртай хэрэгсэл болсон.

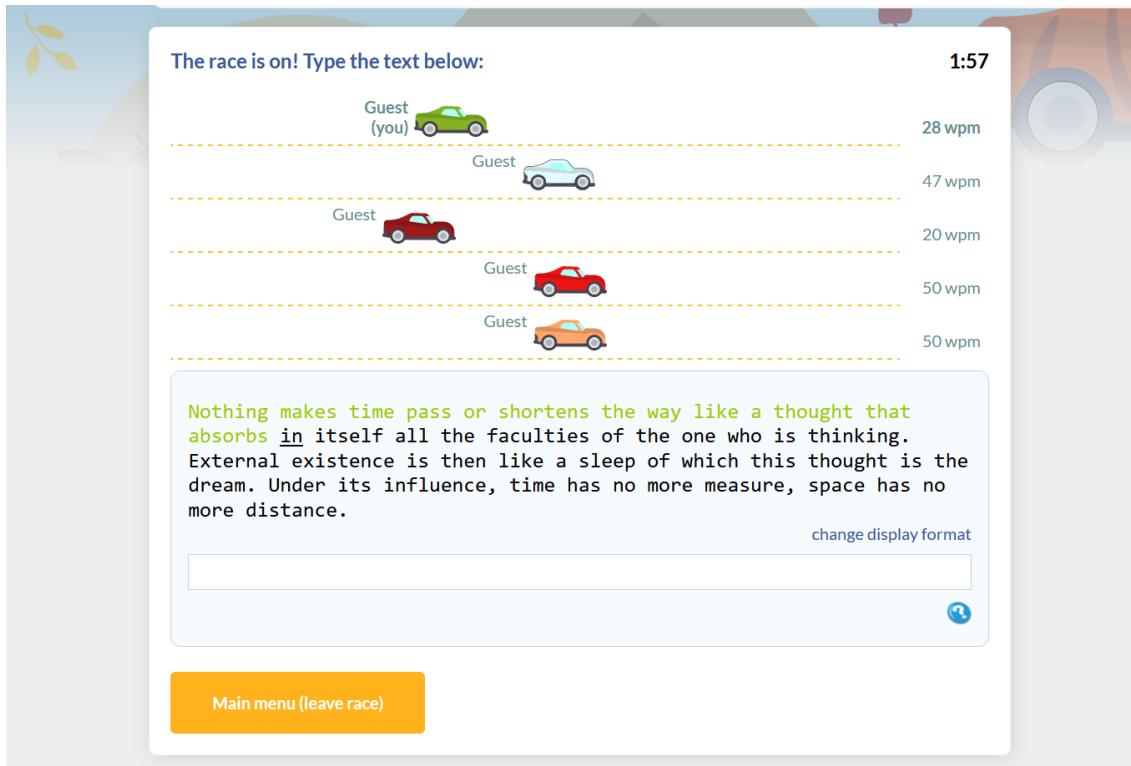
Гол онцлог:

Энэ бол вэб дээрх анхны олон тоглогчийн шивэх тоглоом юм.

2008 оны 3-р сард нээлтээ хийснээс хойш дэлхийн өнцөг булан бүрээс сая сая хүмүүс type racer.com сайт дээр хэдэн зуун сая уралдаанд оролцож, бичих хурдаа минут тутамд 50 үгээр сайжруулсан.

TypeRacer нь 50 өөр хэл дээр байdag.

2010 онд гарсан TypeRacer School Edition нь дэлхийн хамгийн хөгжилтэй боловсрулын бүтээгдэхүүн болох зорилготой юм. K-12 сургуулиудад зориулагдсан бөгөөд бичи-



Зураг 1.2: play.typeracer.com сайтын харагдах байдал

хийг спорт болгон хувиргаж, суралцахыг хөгжилтэй болгодог TypeRacer тоглоомын үзэл баримтлалыг ашигладаг.

1.3 Ашиглах технологи

1.3.1 *Figma - интерфейс дизайн, Prototype хувилбар гаргах бағаж*

Figma нь хэрэглэгчдэд нэг платформ дээр дизайн хийх, загвар гаргах, санал хүсэлийг цуглуулах боломжийг олгодог дизайны хэрэгсэл юм. Дизайнерууд, хөгжүүлэгчид, бүтээгдэхүүний менежерүүд болон оролцогч талууд бодит цаг хугацаанд хамтран ажиллах боломжтой бөгөөд энэ нь дижитал бүтээгдэхүүн дээр ажилладаг багуудад хүчирхэг хэрэгсэл болгодог.

Figma ашиглахын давуу талууд

- Вэб дээр сууринсан бөгөөд энэ нь том хэмжээний програм хангамжийн багц татаж авах шаардлагагүй. Windows, MacOS, Linux зэрэг өөр өөр үйлдлийн системүүд дээр

саадгүй ажилладаг

- Хийсэн өөрчлөлтүүдийг автоматаар хадгалж, шаардлагатай бол хуучин өөрчлөлт рүү буцах боломжийг олгодог
- Figma нь оюутнуудад нэмэлт зардал гаргахгүйгээр иж бүрэн төсөл хэрэгжүүлэх боломжийг үнэ төлбөргүй санал болгодог.



Зураг 1.3: Figma ашиглаж хийсэн компонент

Зурсан интерфейсүүдээ хооронд нь холбож хийсвэрээр аппаа ажиллуулан хэрэглэгчийн 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 аппликашн маань гаргаж өгсөн нь UX/UI дизайнеруудын ашиглах болсон хамгийн том шалтгаануудын нэг юм.

1.3.2 *Next.js - React дээр сууринласан фреймворк*

Сонгосон шалтгаан

Энгийнээр хэлэхэд Next.js нь Javascript програмуудыг хөгжүүлэхэд зориулагдсан React framework юм. Вэб платформыг өндөр гүйцэтгэлтэй, өргөтгөх боломжтой, зөвхөн код дээрээ анхаарал хандуулах боломжийг олгож хурдан ажилладаг.

Next.js давуу талуудаас дурьдвал:

- Вэб програмуудыг бүтээхэд хялбар бүтэцтэй
- Automatic code splitting JS болон CSS шаардлагагүй бол заавал татаж авдаггүй

- Zero config буюу нэг ч тохиргоо хийлгүйгээр төслөө эхлүүлэх боломж
 - Server Side Render хийх (SSR)
 - Typescript болон Fast Refresh дэмждэг
 - HRM болон хөгжүүлэгчдэд ээлтэй хэрэгслүүдтэй
 - 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

Хөгжүүлэлтийг хялбарчлаж хөгжүүлэгчдэд ээлтэй орчинг бүрдүүлэх чадвартай тул Next.js-ийг сонгов.

Технологийн талаар

Next.js¹ нь уян хатан React дээр суурилсан фрэймворк бөгөөд хурдан вэб програмуудыг чанартай үүсгэх боломжийг өгдөг билээ.

1.3.3 Firebase - Өгөгдлийн сан

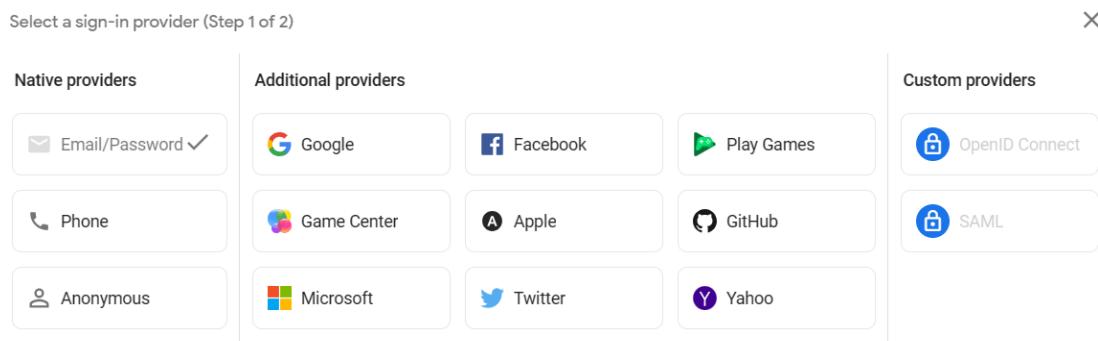
Firebase бол Google-ээс гаргасан гар утас болон вэб програм хөгжүүлэх цогц платформ юм. Энэ нь хөгжүүлэгчдэд програмуудыг хурдан бөгөөд үр дүнтэй бүтээх, сайжруулах, масштаблахад туслах өргөн хүрээний хэрэгсэл, үйлчилгээг санал болгодог. Firebase нь янз бүрийн функц, үйлчилгээг багтаасан бөгөөд үүнийг програм хөгжүүлэхэд түгээмэл сонголт болгодог. Firebase-ийн зарим үндсэн бүрэлдэхүүн хэсэг, боломжуудыг энд харуулав.

Firebase давуу талуудаас дурьдвал:

1. Real-time Database энэ функц нь холбогдсон бүх үйлчлүүлэгчид тоглоомын өгөгдлийг шуурхай шинэчлэх боломжийг олгож, олон тоглогчийн туршлагыг тасралтгүй бий болгодог.

¹Next.js official site <https://nextjs.org>

2. Authentication бүртгэл, нэвтрэх функцийг хэрэгжүүлэхэд хялбар болгодог
3. Serverless Functions эдгээр функцийг тоглоомын session удирдлага, онооны тооцоо зэрэг сервер талын логикийг хэрэгжүүлэхэд ашиглаж болно.
4. Cost efficiency боломжийн үнэ бүхий, төлбөргүй олон боломжуудыг санал болгодог



Зураг 1.4: Firebase Authentication

Firebase-ийг бодит цагийн харилцан үйлчлэл, authentication зэргийг төслийн гол бүрэлдэхүүн хэсэг болгон хэрэглэх болно.

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	Бодит цагт мэдээллүүдийг харуулах (хурд, нарийвчлал, одоогийн байрлал).
ФШ 102	Алдсан, буруу бичсэн үг эсвэл тэмдэгтүүдийг тодруулах.
ФШ 103	Хэрэглэгчид өгөгдсөн текстнээс хамаарч дуусгахад шаардагдах хугацааг өөрчлөх.
ФШ 104	Хэрэглэгчид unique холбоос ашиглан тодорхой найзуудтайгаа өрсөлдөх боломжтой байх
ФШ 105	Уралдааны үеэр бүх оролцогчдын бодит цагийн явцыг харуулах
ФШ 106	Бичих хурд, нарийвчлал дээр үндэслэн тэргүүлэгчдийн самбарыг харуулах
ФШ 107	Веб нь хэрэглэгч бүртгэх боломжтой байх

2.1.3 Функционал бус шаардлагууд

Хүснэгт 2.2: Функциональ бус шаардлага

ФБШ 101	Уралдааны үеэр бодит цагийн хариу үйлдэл үзүүлэг байх.
ФБШ 103	Систем нь дор хаяж 2 хэрэглэгчид хоорондоо өрсөлддөг байх
ФБШ 104	SQL injection, CSRF халдлагууд болон бусад нийтлэг вэб эмзэг байдлаас хамгаалттай байх.
ФБШ 105	Хэрэглэгчийн өгөгдөл болон тоглоомын статистикийг тогтмол нөөцлөдөг байх.
ФБШ 102	Интерфейс UI нь шинэ хэрэглэгчдэд зориулсан ойлгомжтой байх.
ФБШ 106	Төрөл бүрийн төхөөрөмж (компьютер, таблет, гар утас) болон хөтчүүдэд нийцтэй 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 заг-
вартайгаар 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, хэрэглэгчдийн асуудлыг шийдвэрлэж чадаж байгаа эсэх, вэб компонентүү-
дийн байрлал, хоорондын логик үйлдэл зөв байгаа эсэхийг тодорхойлж хэрэглэгчдээс са-
нал хүсэлт авах шаардлагатай. Түүний дараагаас дизайнаа дахин сайжруулж, хөгжүүлэл-
тийн шатанд ороход бэлэн болгох хэрэгтэй.

2.2.2 User Interface дизайны шаардлага

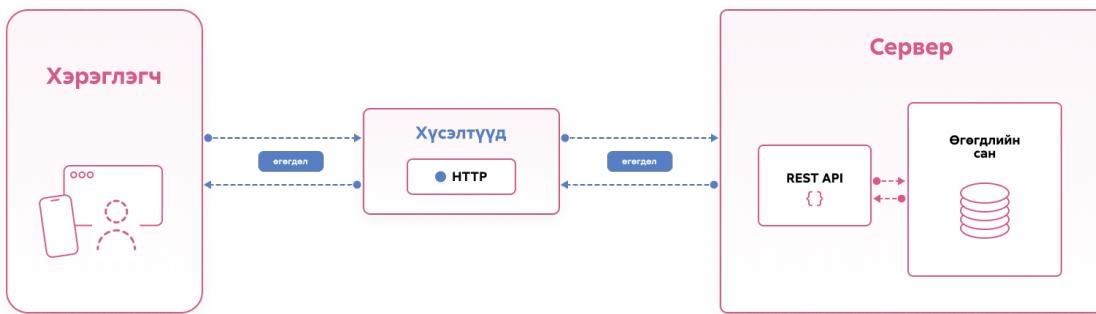
Хүснэгт 2.3: User Interface дизайны шаардлага

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 Extended”хувилбарыг ашигласан байх
ИДШ 107	Элемент хооронд white-space буюу сул зайд сайн гаргаж өгөх

3. СИСТЕМИЙН АРХИТЕКТУР, ЗОХИОМЖ

3.1 Системийн архитектур

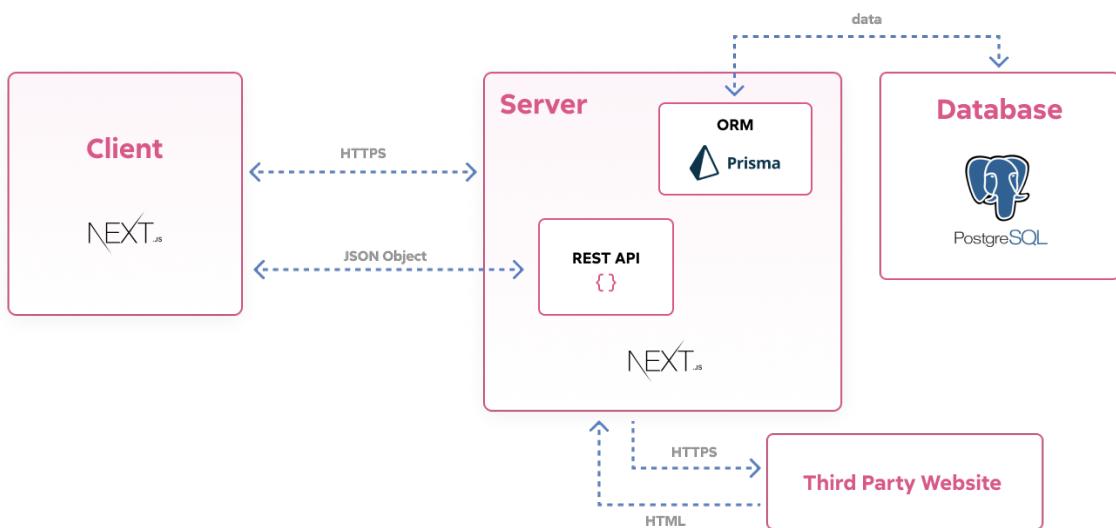
Манай систем нь тийм ч төвөгтэй систем биш мөн энгийн веб хөтчүүдэд зориулж хийгдэх тул системийн архитектурын үлгэр загвар дундаас веб аппд хамгийн өргөн хэрэглэгддэг Хэрэглэгч-Сервер (Client-Server pattern) үлгэр загварыг сонголоо. Хэрэгжүүлэхэд хялбараас гадна Next.js, Prisma ашиглан төслөө хөгжүүлж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



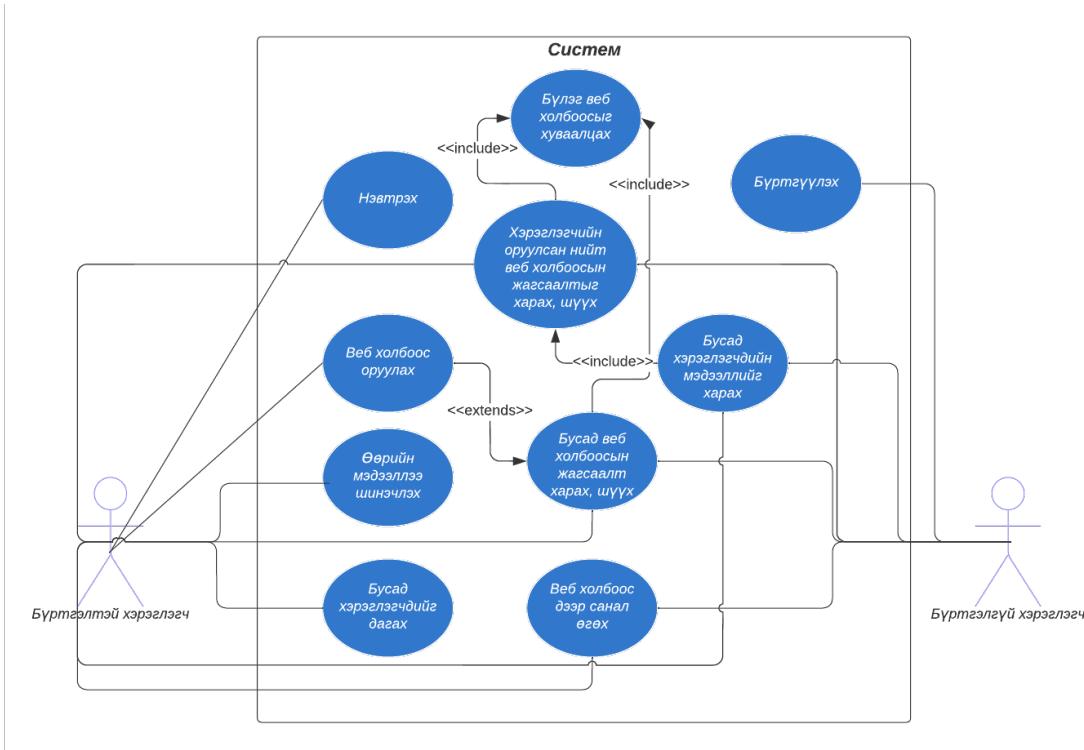
Зураг 3.2: Хэрэгжүүлэх гэж буй системийн архитектур

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 хэрэглэгчдийг дагах** - Өөрийн нүүр хуудсаа сонирхлынхoo дагуу хөгжүүлэхийг хүсвэл бусад хэрэглэгчдийг дагаж, тэдгээрийн оруулсан холбооснуудыг хамгийн ялангуяа 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

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

Зураг 3.5: Persona 2 - Шүрэнцэцг

Use Case

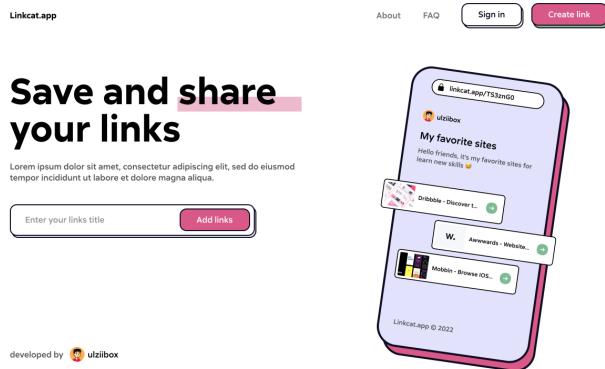
- Миний хувьд мэдээгээ бэлтгэхдээ дийлэнхдээ интернет дэх эх сурвалжуудаас бэлтгэдэг. Нэг асуудал байдаг маань эх сурвалж авдаг хэдхэн вэбсайт дунд л эргэлдэж байгаа.
- Олсон мэдээлүүдээ хамт ажиллаж буй хүмүүс рүүгээ явуулахдаа Telegram ашиглан нэг нэгээр нь хуулж тавин явуулдаг. Хүн болгон руу ингэж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

Эхний Wireframe загварыг гаргаж Prototype хувилбар бүтээсэн нь

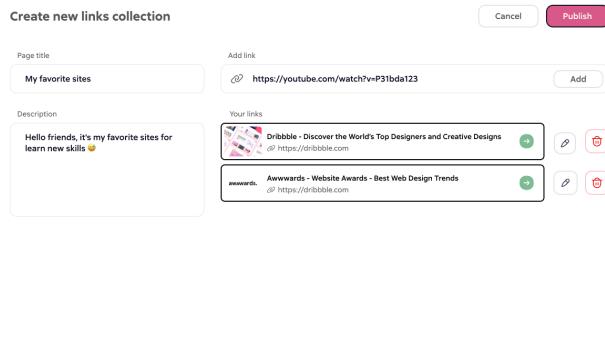
Хамгийн эхэнд нийт 8 хуудсын Wireframe загварыг хэрэглэгчийн Use Case дээрээ тулгуурлан зурж, дизайн дээрх бүх элементүүдээ хооронд нь холбон Prototype¹ хувилбар гаргаж, сонгож авсан хоёр хэрэглэгч дээрээ Usability туршилт хийхэд бэлэн болголоо.

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

Нийт зурсан 8 хуудсаас вебийн гол процесийг илэрхийлэх 4 зургийг орууллаа. Бусад хэсгийг хавсралтаас² үзэх боломжтой

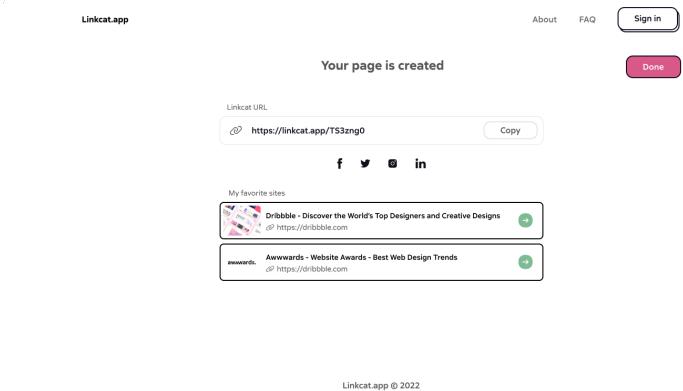


Зураг 3.6: Нүүр хуудас

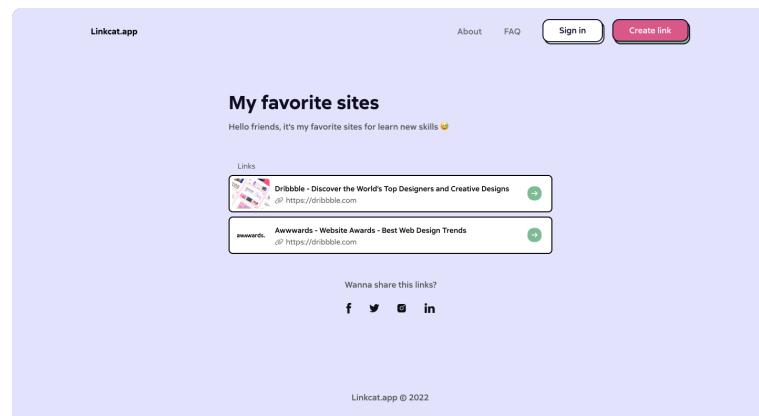


Зураг 3.7: Холбоосуудаа оруулах форм

²Зурсан 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 дизайныхаа ерөнхий шаардлагуудыг тодорхойлов. Үүнээс дурдвал:

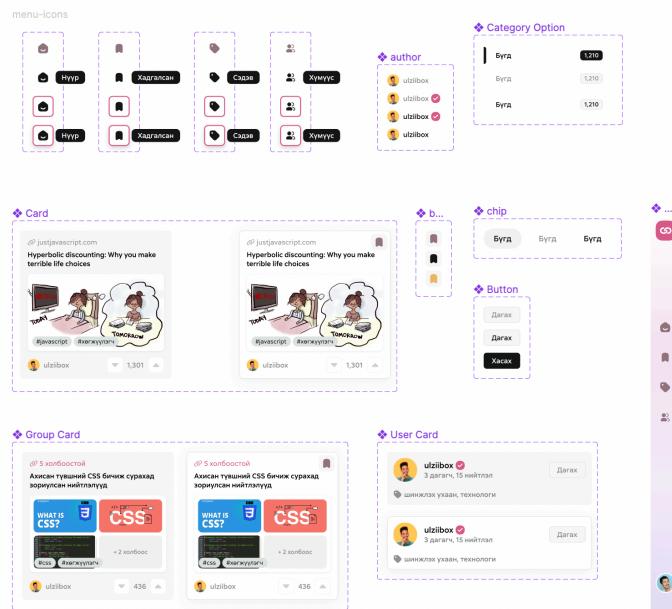
- UI загварын хувьд концепцийг дахин сайжруулах
- Бусад хүмүүсийн оруулсан холбоосыг нийтэд нь харуулах
- Оруулж буй холбоосыг ангилалтай байлгах
- Платформ дээр бүртгэлтэй бусад хэрэглэгчдийг харуулах

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

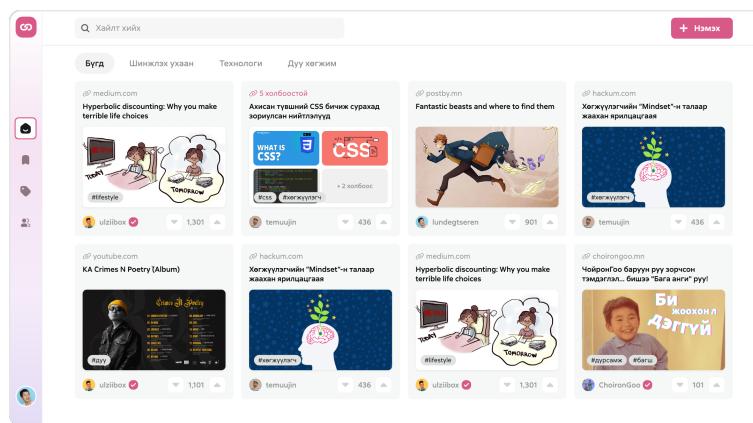
Redesign буюу зурсан дизайнаа дахин сайжруулсан нь

UX/UI гүйцэтгэлийн хамгийн сүүлийн шатанд нийт 12 хуудас дизайныг³ 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 дизайныхаа шийдлийг тайлбарлалаа. Бусад зурсан хуудсыг доор байгаа хавсралтаас харах боломжтой.

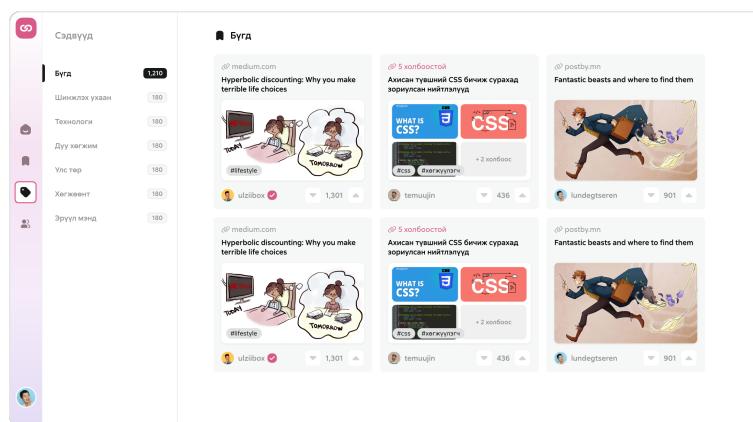
³UX/UI дизайнүүдийн хувилбарыг <https://www.figma.com/proto/EL2nOGmT0BRVxHNuZwH661>



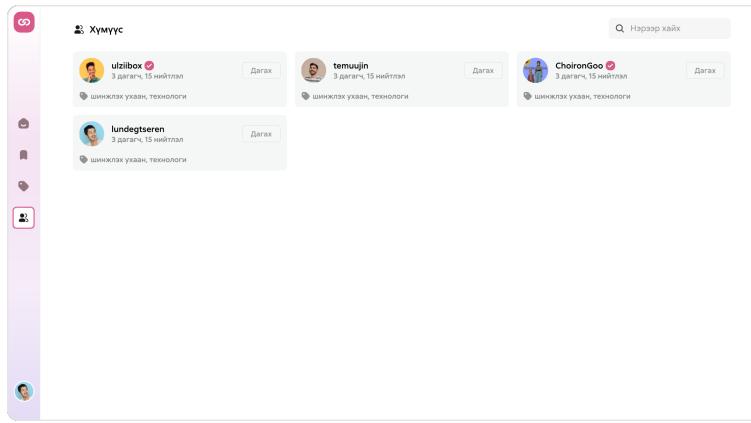
Зураг 3.10: Ашиглсан компонентуудын жишээ



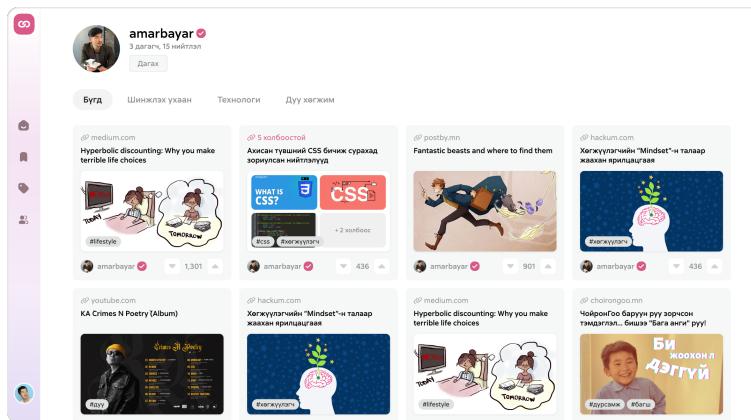
Зураг 3.11: Нэвтэрсний дараах нүүр хуудас



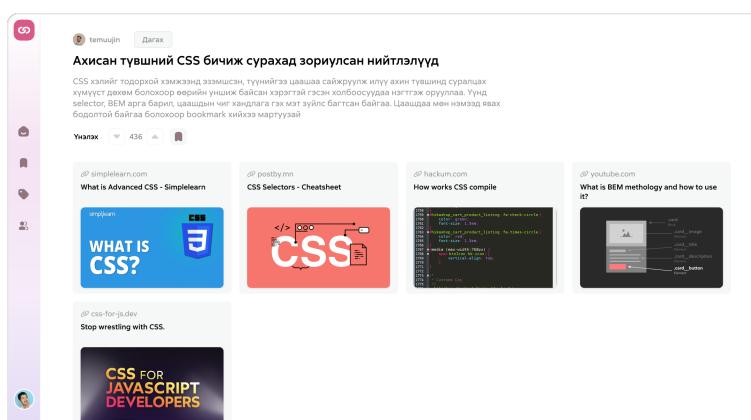
Зураг 3.12: Бүх холбоосуудыг төрлөөр нь шүүж харах хуудас



Зураг 3.13: Хэрэглэгчдийн жагсаалт харагдах хуудас



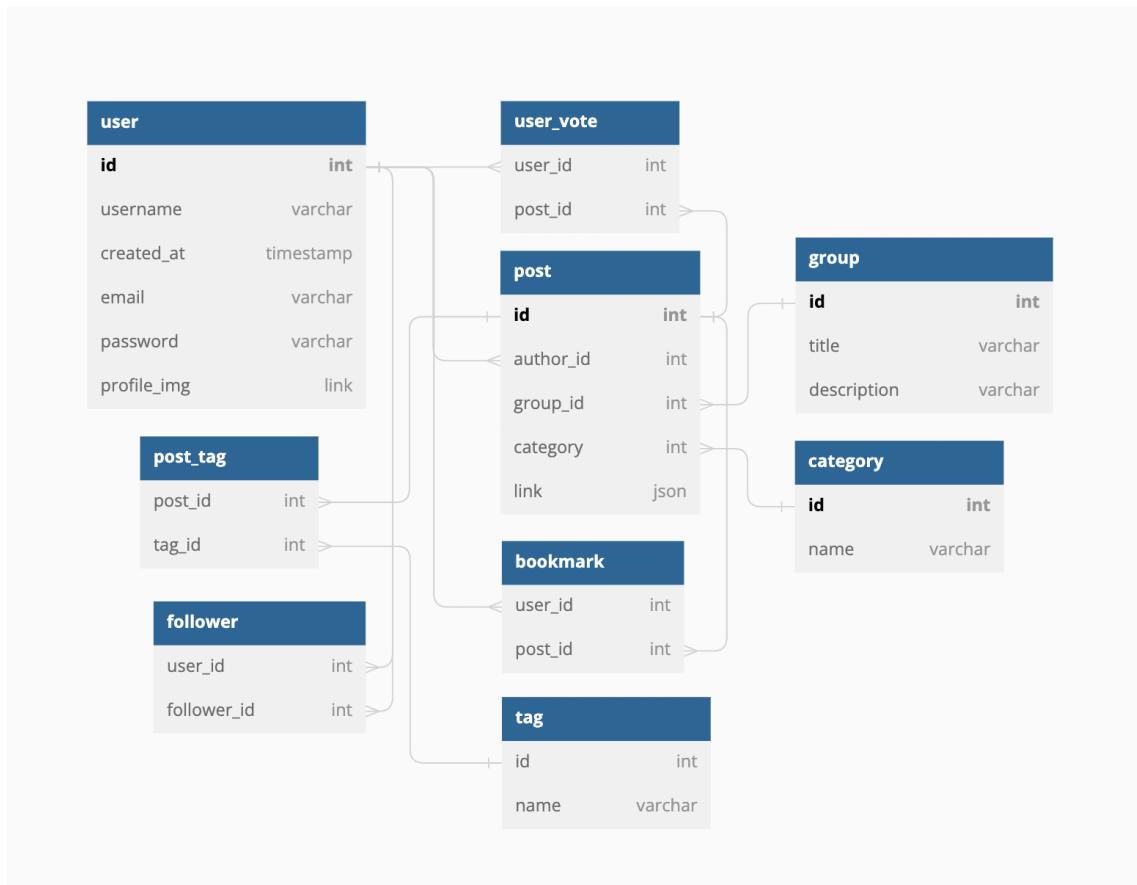
Зураг 3.14: Бусдын профайлыг харах хуудас



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-р 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

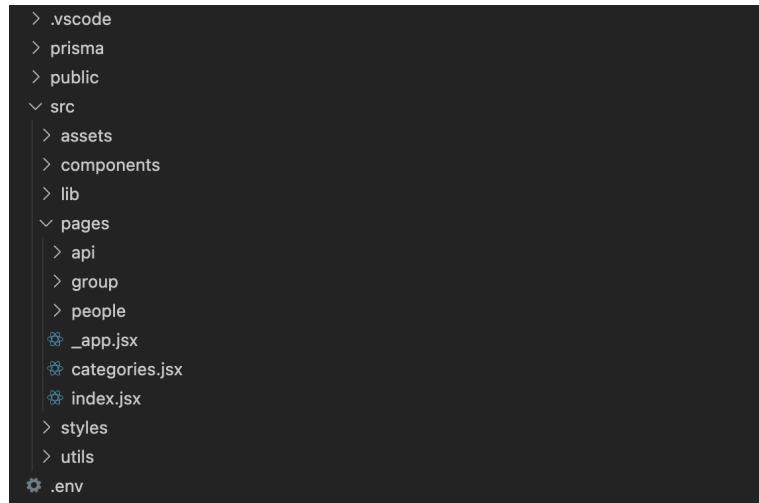
- Хөгжүүлэлтийн орчноо бэлдэх
- Front-end болон Back-end хөгжүүлэлт
- Серверт байршуулах /Deploy/

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 }
```

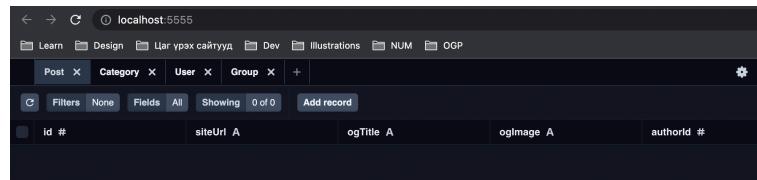
24

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 функциүүдийг хэрэглэж хэрэгтэй өгөгдлийг тодорхой нөхцлүүдээр авч хариуг буцаана.

```

1 import prisma from "../../lib/prisma";
2
3 export default async function handle(req: NextApiRequest, res: NextApiResponse) {
4   const requestMethod = req.method
5   switch (requestMethod) {
6     case 'GET':
7       let following;
8       try{
9         following = await getFollowing(1)
10        console.log(following)
11      } catch (e) {
12        if (e instanceof Prisma.PrismaClientKnownRequestError) {
13          if (e.code === 'P2002') {
14            res.json([])
15          } else{
}

```

```

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') {

```

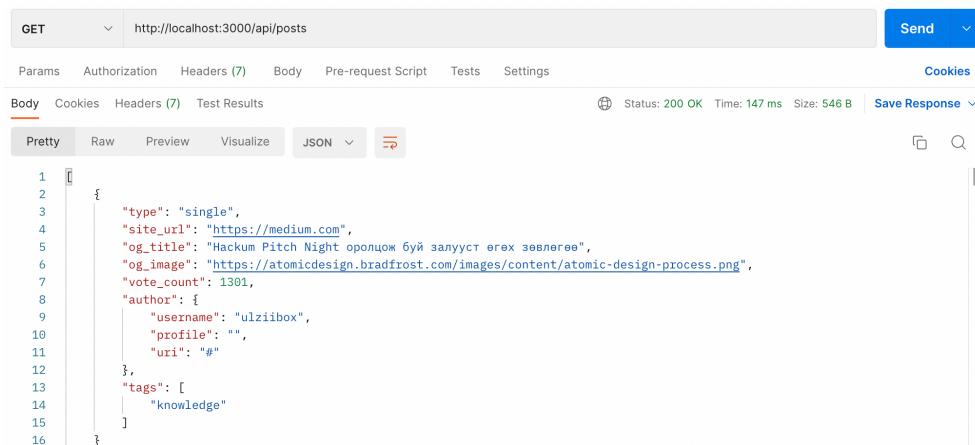
```

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",
```

```

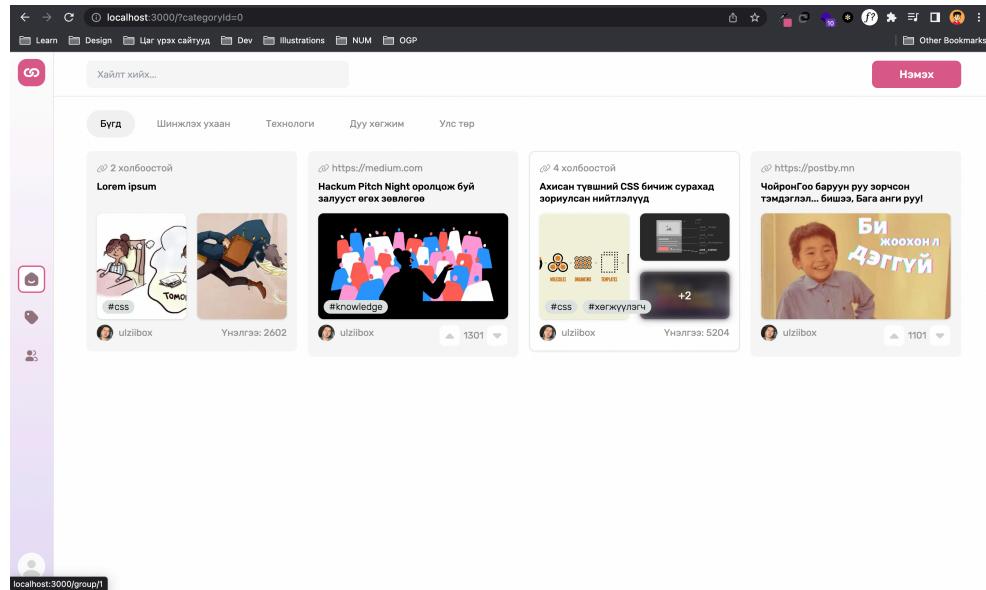
11     },
12     ...CATEGORIES,
13   ]);
14 }, [];
15
16 ...
17
18 {posts.map((item) => {
19   if (item.type === "single") {
20     return (
21       <Card
22         key={item.post.id}
23         site_url={item.post.site_url}
24         og_title={item.post.og_title}
25         og_image={item.post.og_image}
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

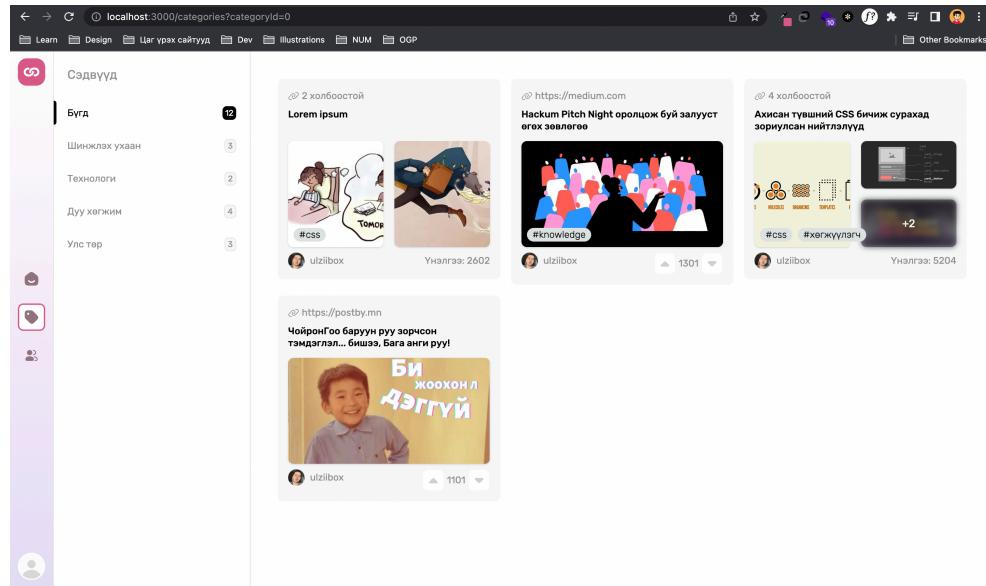
4.3 Yp дүн

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

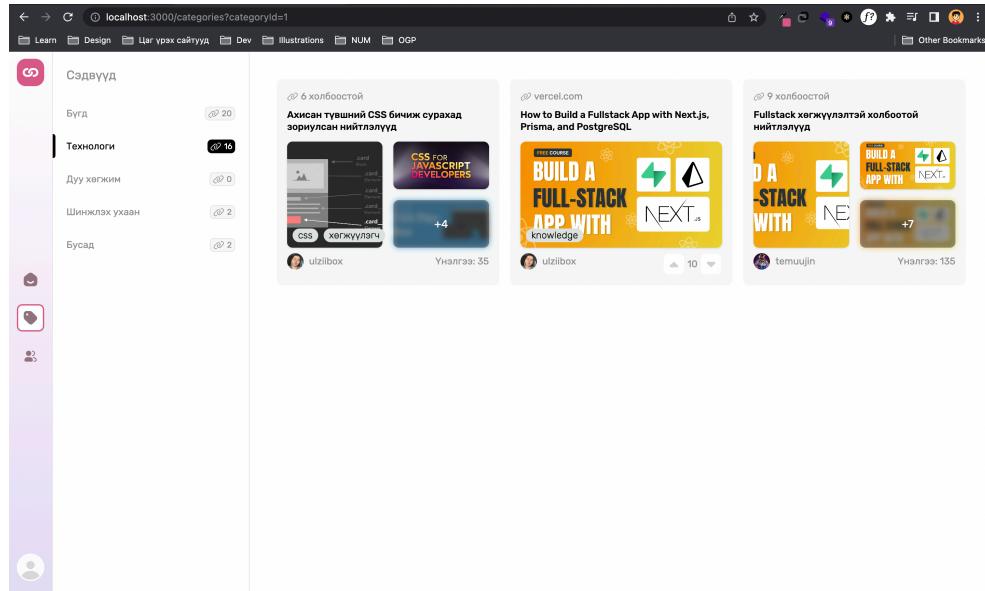


Зураг 4.4: Нүүр хуудас

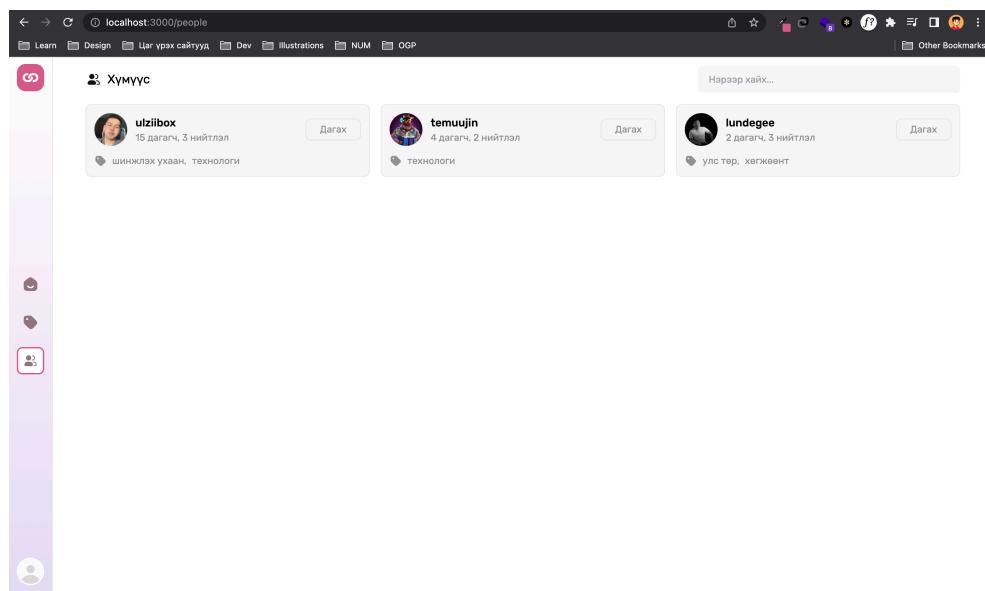
Мөн цаашид дээрх байдлаар шаардлагын дагуу гарсан интерфэйсүүдийг хийж дуусгах ба платформынхoo зарим зургаас оруулав.



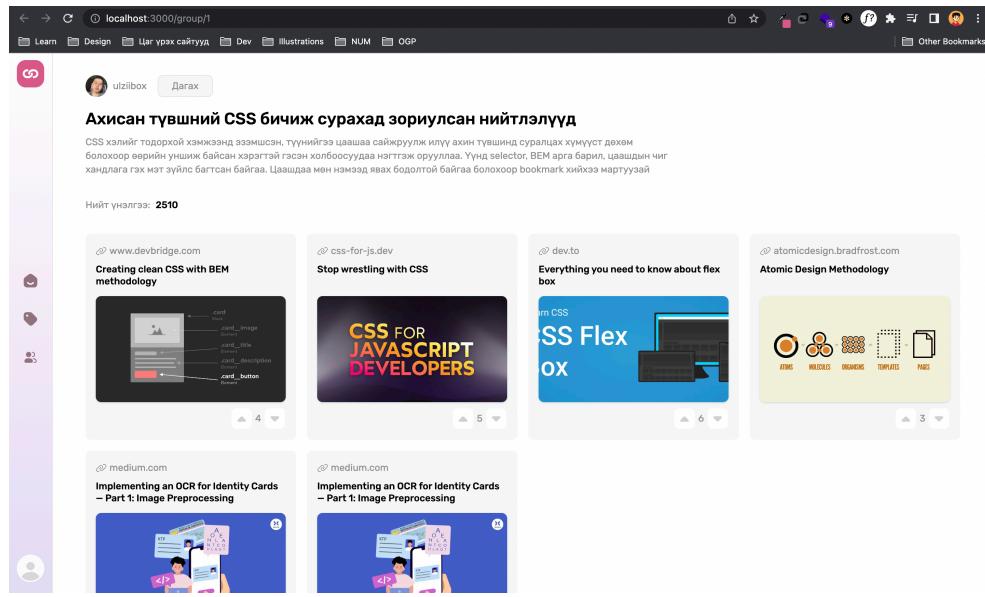
Зураг 4.5: Платформ дээрх сэдвүүдийн жагсаалт болон бүх нийтлэлийг харах



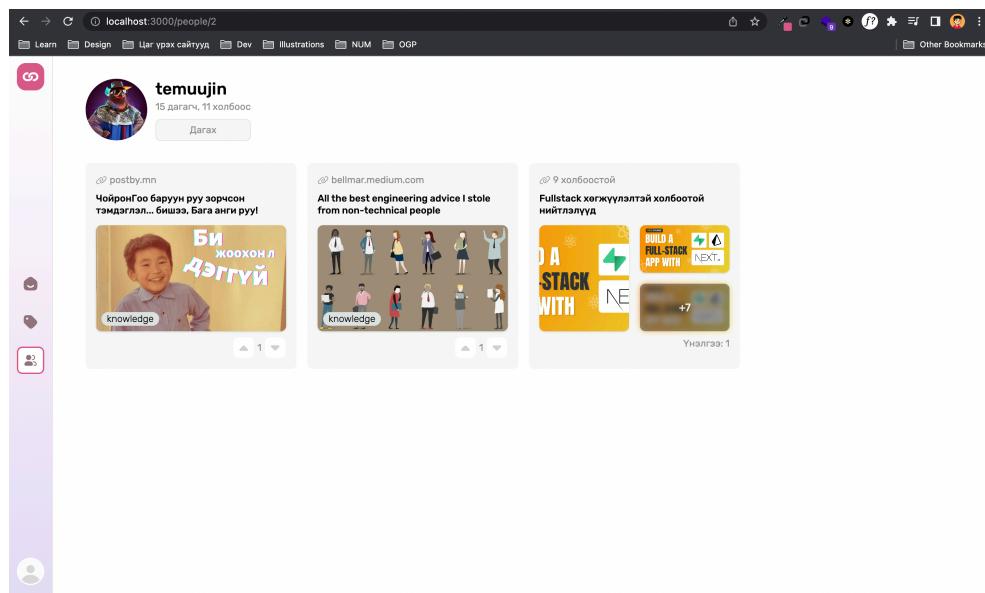
Зураг 4.6: Холбоосуудыг сэдвээр нь шүүж харах



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 |             throw e
118 |         }
119 |     }
120 |     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;
```