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

Энхбаярын Жавхлан

Блокчэйн суурьт лиценз баталгаажуулалт

(Licence validation with blockchain)

Програм хангамж
Бакалаврын судалгааны ажил

Улаанбаатар хот

2024 он

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

Блокчэйн суурьт лиценз баталгаажуулалт
(Licence validation with blockchain)

Програм хангамж
Бакалаврын судалгааны ажил

Удирдагч: _____ Дэд профессор Ч.Алтангэрэл

Гүйцэтгэсэн: _____ Э.Жавхлан (20B1NUM0649)

Улаанбаатар хот

2024 он

Зохиогчийн баталгаа

Миний бие Энхбаярын Жавхлан ”Блокчэйн суурьт лиценз баталгаажуулалт” сэдэвтэй судалгааны ажлыг гүйцэтгэсэн болохыг зарлаж дараах зүйлсийг баталж байна:

- Ажил нь бүхэлдээ эсвэл ихэнхдээ Монгол Улсын Их Сургуулийн зэрэг горилохоор дэвшүүлсэн болно.
- Бусдын хийсэн ажлаас хуулбарлаагүй, ашигласан бол ишлэл, зүүлт хийсэн.
- Ажлыг би өөрөө (хамтарч) хийсэн ба миний хийсэн ажил, үзүүлсэн дэмжлэгийг тайлангийн ажилд тодорхой тусгасан.
- Ажилд тусалсан бүх эх сурвалжид талархаж байна.

Гарын үсэг: _____

Огноо: _____

ГАРЧИГ

ЗУРГИЙН ЖАГСААЛТ

ХҮСНЭГТИЙН ЖАГСААЛТ

Кодын жагсаалт

УДИРТГАЛ

Сэдэв сонгох үндэслэл:

Өнөөгийн цахим орчинд зонхилон тохиолдож буй оюуны өмч болон цахим бүтээгдэхүүний хулгай, өмчлөх эрхийн ил тод байдал, зөвшөөрөлгүй түгээлт зэрэг сорилтуудтай тулгарч байна. Блокчэйн технологийн төвлөрсөн бус, ил тод, хувиршгүй шинж чанарыг ашигласнаар цахим бүтээгдэхүүн эзэмших, лиценз олгоход итгэлцэл, ил тод байдлыг бий болгож, улмаар оюуны өмчийн зөвшөөрөлгүй хулгайн гэмт хэргийг бууруулах зорилготой уг сэдвийг сонгосон.

Зорилго:

Энэхүү судалгааны ажлаар хэрэглэгчид блокчэйн технологиор дамжуулан цахим бүтээлийг хамгаалах, хуваалцах, хандах зөвшөөрөл олгох цахим бүтээлийн лицензийн систем хөгжүүлэх зорилго тавьсан.

Зорилт:

1. Блокчэйн технологийн талаар судлах
2. Системийг хэрэгжүүлэх технологийн талаар судлах
3. Системийн зохиомж, архитектурыг боловсруулах
4. Блокчэйн дээр суурилсан цахим бүтээлийн лицензийн систем хөгжүүлэх

1. ОНОЛЫН СУДАЛГАА

1.1 Блокчэйн технологи

Хамгийн анх блокчэйн технологийн талаар 2008 онд Сатоши Накамото гэдэг этгээдийн нийтэлсэн “Биткойн: Peer-to-Peer Электрон Мөнгөний Тогтолцоо” судалгааны ажлын нийтлэлд дурдагдсан байдаг.

Блокчэйн гэдэг нь өгөгдөл буюу дата мэдээллүүдийг хадгалдаг нэгэн төрлийн мэдээллийн бааз гэж хэлж болно. Бааз доторх дата мэдээллийг Блок гэж нэрлэгдэх хэсгүүдэд багцлан хадгалж уг сүлжээнд холбогдсон бүх компьютерт ижил хуулбар болгон тархмал хэлбэрээр хадгална. Тэдгээр блокуудыг өөр хоорондоо гинжин хэлхээ буюу математик тооцоолол, цахим нууцлалын аргаар хэлхэн холбосноор бидний ярьж буй Блокчэйн үүсэх юм.

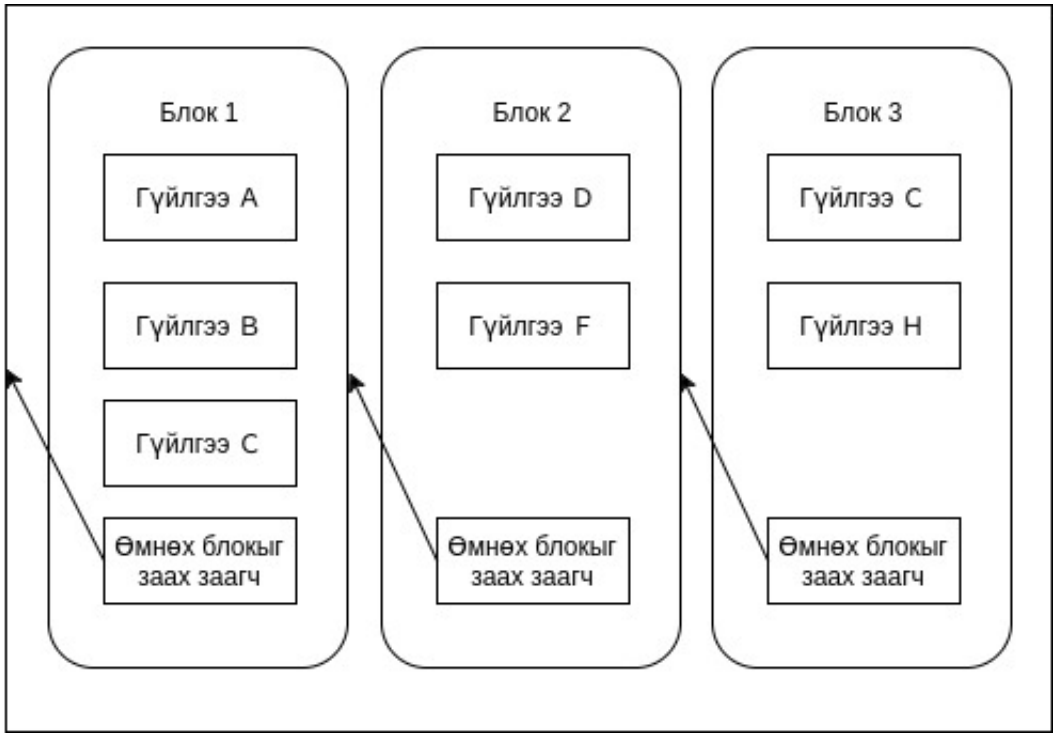
1.2 Блокчэйний онцлог

Тархсан Peer-to-Peer (P2P) сүлжээ гэдэг нь сүлжээнд оролцогч буюу зангилаанууд нь газарзүйн хувьд тархсан байдаг ба аль ч хоёр зангилаа хоорондоо байршлаас үл хамааран ямар нэг серверээр дамжилгүйгээр өөр хоорондоо шууд холбогддог сүлжээ юм.

Тархсан бүртгэлийн дэвтэр буюу Distributed ledger technology (DLT) технологи нь мэдээллийг тархсан P2P сүлжээний оролцогч нар дээр хадгалдаг технологи бөгөөд уламжлалт өгөгдлийн сангийн системээс ялгарах гол ялгаа нь төвлөрсөн өгөгдлийн сан болон төвлөрсөн удирдлагын функц байхгүйд оршино.

Блокчэйн нь DLT технологийн гол төлөөлөгч бөгөөд мэдээллийн гинжин хэлхээ юм. Блокчэйнд тогтсон хэмжээтэй блок үүсгэж, үүн дотроо мэдээллийг хадгалах ба эхний блок дүүрэхэд дараагийн шинэ блок үүсгэдэг. Эдгээр блок нь хэш функцээр кодлогдсон байх ба блокийг цаг хугацааны дагуу жагсааж, блок тус бүр яг өөрийн өмнөх блокийн мэдээллийг өөр дотроо хадгалах байдлаар гинжин бүтцийг үүсгэнэ.

Блокчэйн технологийн хамгийн чухал, онцлох давуу тал нь төвлөрсөн бус тархсан бүтэцтэй бөгөөд сүлжээнд байгаа бүх компьютер блокчэйний халдашгүй чанарыг үргэлж баталгаажуулж байдаг ба хэн нэгэн, эсвэл аль нэг компани үүн доторх өгөгдөл түүний бүрэн бүтэн байдлыг удирдах боломжгүй байдагт байгаа юм. Блокчэйний бүх зангилаа ижил мэдээллийг агуулж байдаг болохоор "А" зангилаан дахь өгөгдөл эвдэрч гэмтвэл блокчэйний хэсэг болж чадахгүй, учир нь өгөгдөл нь бусад "В" болон "С" зангилааны өгөгдөлтэй ижил байж чадахгүй болно.



Зураг 1.1: Блокчэйний өгөгдөлийн бүтэц

1.3 Блокчэйний нууцлалын технологи

1.3.1 Криптограф хэш

Криптограф хэш функц нь оруулсан өгөгдлийн уртаас үл хамааран тогтсон урттай хэш утгуудыг буцаадаг. Оролтын зөвхөн нэг тэмдэгт өөрчлөгдөхөд гаралтын хэш утгууд нь эрс ялгаатай байна. Энэ шинж чанарыг ашиглан, гүйлгээний өгөгдөл болон бусад бүх өгөгдлийн хувьд засвар ороогүй болохыг баталгаажуулах боломжийг олгодог. Жишээ нь, та Мас-ын

командлайн-аар дараах командыг оруулбал, SHA-256 hash функцийн утгыг хялбархан олох болно.

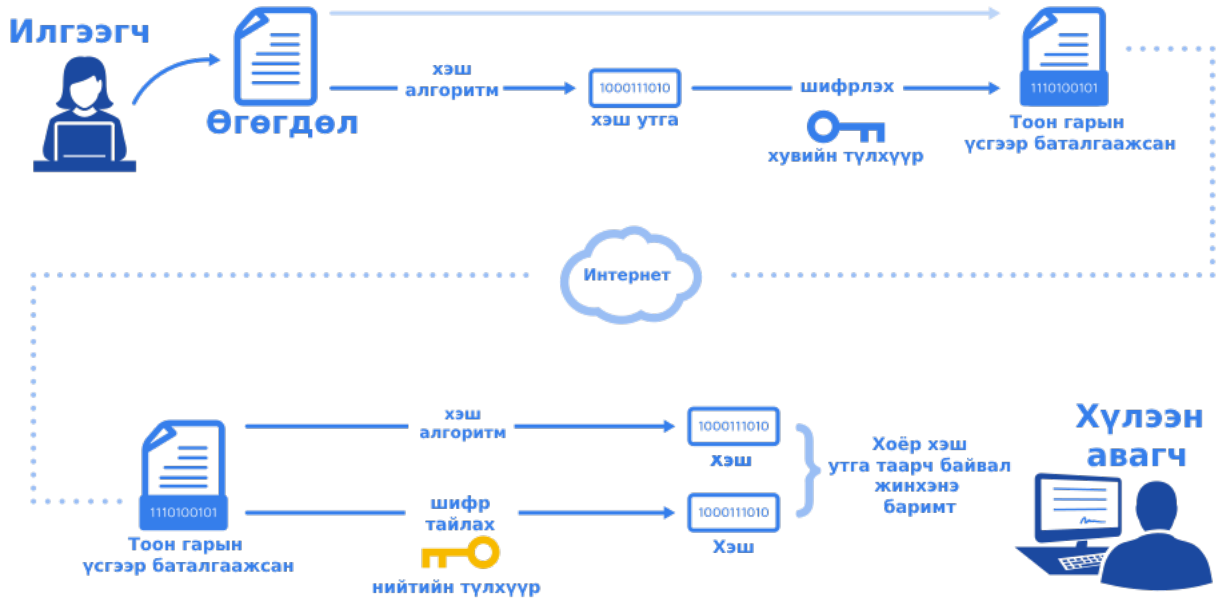
```
❯ echo "Hello World" | openssl sha256
SHA2-256(stdin)= d2a84f4b8b650937ec8f73cd8be2c74add5a911ba64df27458ed8229da804a26
❯ echo "Hallo World" | openssl sha256
SHA2-256(stdin)= e6c1396639c0b79bebc94e4448cfe2700b871d45d0d38d98df6ee9da3f09d35c
```

Зураг 1.2: ”Hello World”, ”Hallo World” гэсэн үгнүүдийн хэшийг бодсон байдал

”Hello World”, ”Hallo World” гэсэн үгнүүдийн sha256 хэшийг бодсон байдал жишээн дээр, ”Hello World” гэсэн үгний SHA-256 hash утга болон, ”e”-г ”a”-гээр сольсон үгийн SHA-256 hash утгыг харуулсан байна. Зөвхөн нэг үсгээр ялгаатай боловч, тэдгээрийн hash утгууд нь эрс ялгаатай байна. Энэ мэтчилэн hash функц нь оролт нь 1 байтаар л ялгаатай байхад, эрс өөр үр дүн гаргадаг шинж чанарыг агуулж байдаг. Энэ шинж чанарыг ашиглан, гүйлгээний өгөгдөл болон тэдгээрийг хадгалах блокийн бүх өгөгдлийн хувьд гарах hash утгыг тухайн өгөгдлийн бүтцэд оруулснаар, засвар ороогүй болохыг баталгаажуулах боломжтой болно.

1.3.2 Тоон гарын үсэг

Тоон гарын үсэг нь дижитал мессеж эсвэл баримт бичгийн жинхэнэ эсэхийг шалгах математик аргачлал юм. Энэ нь хос түлхүүр үүсгэх замаар ажилладаг: өргөн тархсан нийтийн түлхүүр, нууцлагдсан хувийн түлхүүр. Гарын үсэг зурахдаа баримт бичгийн өвөрмөц хэшийг үүсгэж, хувийн түлхүүрээр шифрлэж, тоон гарын үсгийг бүрдүүлдэг. Хүлээн авсны дараа хэшийг илгээгчийн нийтийн түлхүүрээр тайлж, хүлээн авсан баримтаас шинэ хэш үүсгэнэ. Хэрэв хоёулаа таарч байвал энэ нь тухайн баримт бичиг нь жинхэнэ бөгөөд ямар нэгэн өөрчлөлт ороогүй гэсэн үг юм.



Зураг 1.3: Цахим гарын үсгийн ажиллах зарчим

Тоон гарын үсэгт хаш функц болон хос түлхүүрийг нийлүүлж ашигласнаар, өгөгдөл илгээгчийг болон агуулгын засагдаагүй гэдгийг баталгаажуулах ажлыг зэрэг гүйцэтгэдэг юм. Блокчэйнд өмнөх хэсгийн хаш функц болон дээр өгүүлсэн цахим гарын үсгийг аль алийг нь ашигладаг бөгөөд гүйлгээ тус бүрийн үнэн зөв байдал, нийцтэй байдлын талаарх мэдээллийн илгээгч, агуулгын бүрэн бүтэн(засагдаагүй) байдлын баталгаа зэрэг төрөл бүрийн зорилгоор ашигладаг.

1.4 Ухаалаг гэрээ

Ухаалаг гэрээ гэдэг нь дундын зуучлагч буюу хуульч, нотариатгүйгээр хоёр этгээд гэрээ байгуулсныг баталсан компьютерын код бөгөөд тухайн гэрээний нөхцөл, үүрэг, хариуцлагыг багтаасан байна. Анх этереум нь ухаалаг гэрээг оруулсан блокчэйн гэдгийг гаргаж, түүний дараагаар олон тооны блокчэйнд ухаалаг гэрээг оруулж ирсэн. Ухаалаг гэрээ нь зөвхөн нөхцөл, үүргийг заахаас гадна автоматаар биелэх боломжтой байдаг.

Анх 1996 онд Nick Szabo ухаалаг гэрээний санааг нь гаргаж ирсэн. Гол санаа нь хүнээс хамааралгүйгээр урьдчилан тодорхойлсон ямар нэг нөхцөлийн биелэх үед автоматаар үйлдэл

хийгдэнэ.

1. Хийгдсэн үйлдэл/гүйлгээ нь олон нийтэд ил байх ч, хэн хийсэн бэ гэдэг нь нууц байж болдог.

2. Блокчэйн сүлжээний бүх зангилаанууд Ухаалаг гэрээг ажиллуулдаг.

3. Цаг хугацаа хэмнэхээс гадна гарч болох олон асуудлыг шийдэх боломжтой. (3-дагч этгээдийг оролцоо хэрэггүй)

1.5 Блокчэйн зарим хэрэглээ

Олон улсын хэмжээнд стартап компаниуд блокчэйн технологийг ашигласан шинэ системийг эрүүл мэнд, даатгал, татвар зэрэг олон салбарт санал болгож байна.

Жишээлбэл, эрүүл мэндийн салбарт блокчэйн рүү иргэний эрүүл мэндийн болон эмчилгээний түүхийг оруулдаг болгох систем юм. Энэ тохиолдолд эмчлэгч эмч тухайн иргэний мэдээллийг харах судалгаа, шинжилгээний зорилгоор авч ашиглахаар бол системд хүсэлт гаргахад зөвхөн тухайн иргэний зөвшөөрлөөр системээс мэдээлэл нь харагдана. Хүний эрүүл мэндийн мэдээлэл блокчэйнд хадгалагдсанаар тухайн хүн дэлхийн аль ч улс оронд эмчилгээнд хамрагдахад асуудалгүй болж байгаа юм. Мөн блокчэйнд хүн өөрийн итгэмжлэгдсэн төлөөллийг нэмж өгөх боломжтой бөгөөд тухайн хүн өөрөө блокчэйнээс мэдээллээ гаргаж өгөх боломжгүй нөхцөлд ашиглагдах юм. Хэрэв блокчэйн ашиглагдаж эхэлбэл зайнаас эмчлэх, эмчилгээний зөвлөгөө өгөх зэрэг шинэ төрлийн үйлчилгээнүүд хүчээ авах юм.

Нэгдсэн Үндэстний Байгууллага 2017 онд блокчэйн технологи ашигласан олон төрлийн санал, санаачлагыг хэрэгжүүлснээс үүний нэг болох тусламж түгээлтийн бүртгэлийн систем амжилттай хэрэгжсэн байна. НҮБ-аас гаргасан судалгаагаар, нийт тусламжийн 30 орчим хувь нь очих ёстой хүлээн авагчдаа хүрдэггүй гэж гарсан байна. 2017 оны тавдугаар сараас НҮБ-ын Дэлхийн хүнсний хөтөлбөрт хэрэгжсэн хүрээнд Сирийн дүрвэгчдэд үзүүлж байгаа тусламжийг этереум блокчэйн ашиглаж түгээжээ. Тодруулбал, Иордан улсын дүрвэгчдийн

хуаранд байрлаж байгаа Сири улсын 10500 дүрвэгчид хүнсний бүтээгдэхүүн (1.4 сая ам.доллар) түгээхэд криптовалютад суурилсан ваучер тарааж, уг ваучераа ашиглан хуаранд байрлах дэлгүүрээс хүнсний бүтээгдэхүүн авах боломжийг хангажээ. НҮБ-аас уг төслийг өмнөх тусламжтай харьцуулахад маш амжилттай хэрэгжсэн гэж үзэж байгаа бөгөөд 2018 оны хоёрдугаар улиралд тусламжинд хамрагдах хүний тоог 500,000-д хүргэхээр төлөвлөж байна гэж мэдээлж байна.

НҮБ-аас хамгийн сүүлд эхлүүлсэн нэг ажил нь хүүхдийг блокчэйнд бүртгэлжүүлэх систем юм. Хуурамч бичиг баримт үйлдэн хүүхэд хил дамнуулахыг зогсооход хамгийн ээдрээтэй зүйл нь жинхэнэ юм шиг бүрдүүлсэн хуурамч бичиг баримтыг таних ажил байдаг. Хүний наймаа ихээр явагддаг бүс нутагт хүүхдүүдийг шат дараатайгаар албан ёсны бүртгэлтэй болгож, түүнийг нь НҮБ-ын блокчэйн системд хадгална. Энэ төрлийн гэмт хэрэг хамгийн их явагддаг Молдав улсад хэрэгжүүлж эхэлсэн ажээ. НҮБ-ийн судалгаагаар 5-аас доош насны хүүхэд бүртгэлжээгүй байх тохиолдол зарим бүс нутагт их байдаг байна.

Швейцарийн Зуг (ZUG) хот нь крипто хот болохоор ажиллаж байгаа бөгөөд ийм уриа гаргасан бусад хот болох Сан-Франциско, Лондон, Токио, Сингапур, Нью-Иорк, Амстердамаас ялгагдах зүйл нь санхүү болон технологийн гарааны бизнесээ эхэлж буй компаниудад хууль эрх зүйн орчин нь маш тааламжтай юм. Зуг хотын удирдлага крипто хөндий байгуулж, иргэдээ блокчэйнд бүртгэж эхэлсэн ба 2017 оны арваннэгдүгээр сараас иргэддээ зориулж цахим ID авах вебийн үйлчилгээг нээсэн нь этереум блокчэйнд суурилсан ба хэрэглэгч хаанаас ч өөрийн мэдээллийг оруулан цахим ID-гаа авах боломжтой бөгөөд хотын зүгээс уг мэдээллийг зөвхөн шалгаж баталгаажуулах эрхтэй. Энэхүү цахим ID-гаа ашиглаад иргэд зөвхөн хотын үйлчилгээг (хэрэглээний төлбөр, түрээсийн төлбөр) авахаар хязгаарлагдахгүй ба 2018 оны хавар сонгуулийн санал өгөхөд (e-vote) ашиглахаар бэлдэж байна.

1.6 Дижитал эрхийн менежмент (DRM)

Дижитал эрхийн менежмент (DRM) нь цахим баримт бичиг, дуу хөгжим, видео, цахим ном, программ хангамж болон бусад дижитал медиа зэрэг дижитал контентыг хамгаалах,

удирдахад ашигладаг технологи, процессыг хэлнэ. DRM системүүд нь цахим контентын хандалтыг хянах, ашиглалтын хязгаарлалтыг хэрэгжүүлэх, оюуны өмчийн эрхийг хамгаалах зорилготой юм.

1.6.1 DRM-ийн үндсэн ойлголт ба бүрэлдэхүүн хэсгүүд:

- **Шифрлэлт:** Шифрлэлт нь криптограф алгоритмыг ашиглан цахим контентыг унших боломжгүй формат руу хөрвүүлэх явдал юм. Шифрлэгдсэн контентод зөвхөн шаардлагатай код тайлах түлхүүрийг эзэмшсэн эрх бүхий хэрэглэгчид хандах буюу тайлж болно.

- **Хандалтын хяналт:** DRM систем нь дижитал контент руу хэн хандах, үзэх, өөрчлөх, түгээх боломжтойг зохицуулах хандалтын хяналтын механизмыг хэрэгжүүлдэг. Хандалтын эрхийг ихэвчлэн хэрэглэгчийн үүрэг, лиценз эсвэл контент эзэмшигчээс олгосон зөвшөөрөл дээр үндэслэн тодорхойлдог.

- **Лицензийн менежмент:** DRM шийдлүүд нь хэрэглэгчдэд дижитал контент руу нэвтрэх, ашиглах зөвшөөрөл олгохын тулд лицензэд суурилсан загваруудыг ашигладаг. Лицензүүд нь ашиглалтын хугацаа, зөвшөөрөгдсөн төхөөрөмж, нэгэн зэрэг хэрэглэгчдийн тоо зэрэг ашиглалтын нөхцөл, нөхцөлийг тодорхойлдог.

- **Тоон усан тэмдэг:** Тоон усан тэмдэг нь үл үзэгдэх танигч эсвэл гарын үсгийг цахим контентод оруулахад ашигладаг техник юм. Усан тэмдэглэгээг контентын зөвшөөрөлгүй хуулбарыг эх сурвалж руу нь буцаах эсвэл контентын жинхэнэ эсэхийг шалгахад ашиглаж болно.

- **Хуулбарлах хамгаалалт:** DRM системүүд нь цахим контентыг зөвшөөрөлгүй хуулбарлах, хуулбарлахаас сэргийлэхийн тулд хуулбарлах хамгаалалтын механизмыг хэрэгжүүлдэг. Хулгайлах, зөвшөөрөлгүй түгээхээс урьдчилан сэргийлэхийн тулд хуулбарлахаас урьдчилан сэргийлэх, хуулбарлах хяналт, хуулбар илрүүлэх зэрэг арга техникийг ашигладаг.

2. СИСТЕМИЙН СУДАЛГАА, ЗОХИОМЖ

2.1 Системийн шаардлага

2.1.1 Функционал шаардлагууд

/ФШ 10/ Систем нь цахим бүтээл болон лицензийн талаарх мэдээллийн найдвартай байдлыг хадгалахын тулд блокчэйнтэй харилцах ёстой.

/ФШ 20/ Системд хэрэглэгчид крипто хэтэвчээ ашиглан өөрийгөө баталгаажуулах боломжтой байх ёстой (жишээ нь, MetaMask).

/ФШ 30/ Системд зөвхөн холбогдсон түрийвчтэй, баталгаажуулсан хэрэглэгчид системийн функцэд хандах эрхтэй байх ёстой.

/ФШ 40/ Систем нь хэрэглэгчдэд янз бүрийн төрлийн цахим бүтээлийг (жишээ нь, видео, аудио, PDF, зураг) аюулгүйгээр байршуулах боломжтой байх ёстой.

/ФШ 50/ Систем нь цахим бүтээлийг байршуулахдаа бүтээлийн мэдээллийг бүртгэх ёстой.

/ФШ 60/ Систем нь байршуулсан файлуудыг IPFS (InterPlanetary File System) дээр найдвартай хадгалах ёстой.

/ФШ 70/ Систем нь өгөгдлийн нууцлал, аюулгүй байдлыг хангах үүднээс байршуулахаас өмнө файлуудыг шифрлэх ёстой.

/ФШ 80/ Систем нь шифрлэгдсэн файлуудыг зохих зөвшөөрөлтэй эрх бүхий хэрэглэгчдэд зориулан тайлах ёстой.

/ФШ 90/ Системд хэрэглэгчдэд цахим бүтээлийн эзэмшигчдээс тухайн цахим бүтээлд хандах лицензийн хүсэлт илгээх боломжтой байх ёстой.

/ФШ 100 Системд цахим бүтээл эзэмшигчид лицензийн хүсэлтийг зөвшөөрөх эсвэл татгалзах боломжтой байх ёстой.

/ФШ 110 Систем нь цахим бүтээл эзэмших, түүнд лиценз олгох асуудлыг зохицуулахын тулд ухаалаг гэрээг блокчэйн дээр байршуулж, удирдах ёстой.

/ФШ 120 Системд хэрэглэгчид файл үүсгэх, лицензийг удирдах, өмчлөлийг баталгаажуулахын тулд ухаалаг гэрээнүүдтэй харилцах ёстой.

/ФШ 130 Системд хэрэглэгчид систем дээр байрлуулсан цахим бүтээлийн дэлгэрэнгүй мэдээллийг үзэх боломжтой байх ёстой.

/ФШ 140 Цахим бүтээлийн лиценз авахад лицензэд өвөрмөц дугаар олгож, блокчэйн дээр хадгалах ёстой.

/ФШ 150 Систем нь хэрэглэгчид лиценз авсны дараа лицензийн дугаар, файлын мэдээлэл зэрэг лицензийнхээ дэлгэрэнгүй мэдээллийг агуулсан цахим гэрчилгээ олгох ёстой.

2.1.2 Функционал бус шаардлагууд

/ФБШ 10/ Блокчэйн технологи нь өгөгдлийн бүрэн бүтэн байдлыг хангаж, лицензийн мэдээллийг зөвшөөрөлгүй өөрчлөхөөс сэргийлнэ.

/ФБШ 20/ Систем нь гүйцэтгэлийн бууралтгүйгээр олон тооны хэрэглэгчид болон лицензүүдийг зохицуулах чадвартай байх ёстой.

/ФБШ 30/ Ухаалаг гэрээ нь модульчлагдсан байх ёстой бөгөөд шинэчлэгдэхэд хялбар байх ёстой.

/ФБШ 40/ Систем нь хүлээн зөвшөөрөгдсөн тодорхой хугацааны дотор баталгаажуулах хүсэлтийг хурдан боловсруулах чадвартай байх ёстой.

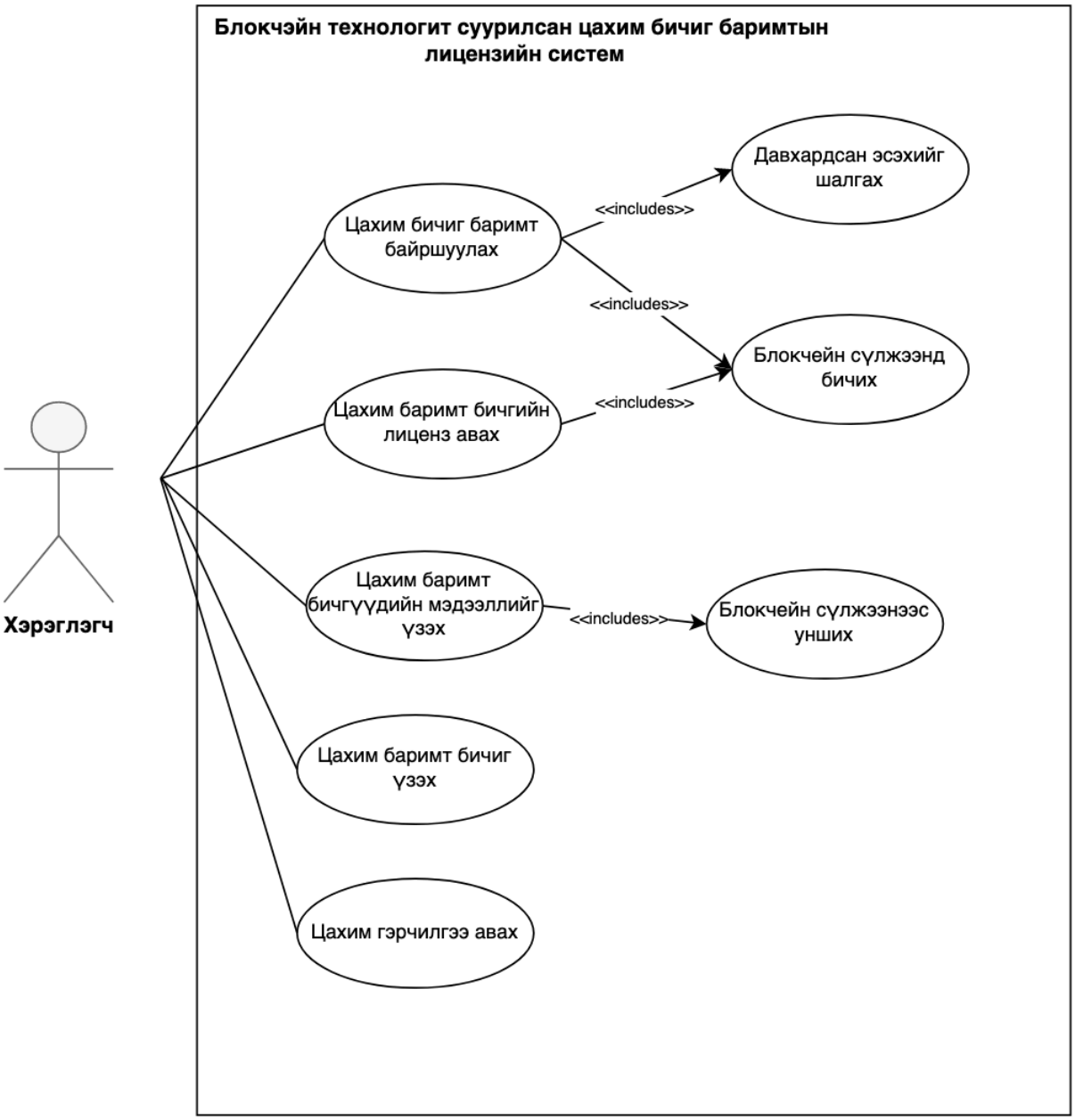
/ФБШ 50/ Систем нь янз бүрийн техникийн чадвартай хэрэглэгчдэд үүнийг үр дүнтэй ашиглах боломжийг олгодог хэрэглэгчдэд ээлтэй интерфейстэй байх ёстой.

/ФБШ 50/ Систем нь янз бүрийн үйлдлийн систем, хөтөч, төхөөрөмжтэй нийцтэй байх ёстой.

/ФБШ 60/ Систем нь лиценз олгох, дижитал гүйлгээ, блокчэйн технологитой холбоотой аливаа зохицуулалтын шаардлагад нийцэж байх ёстой.

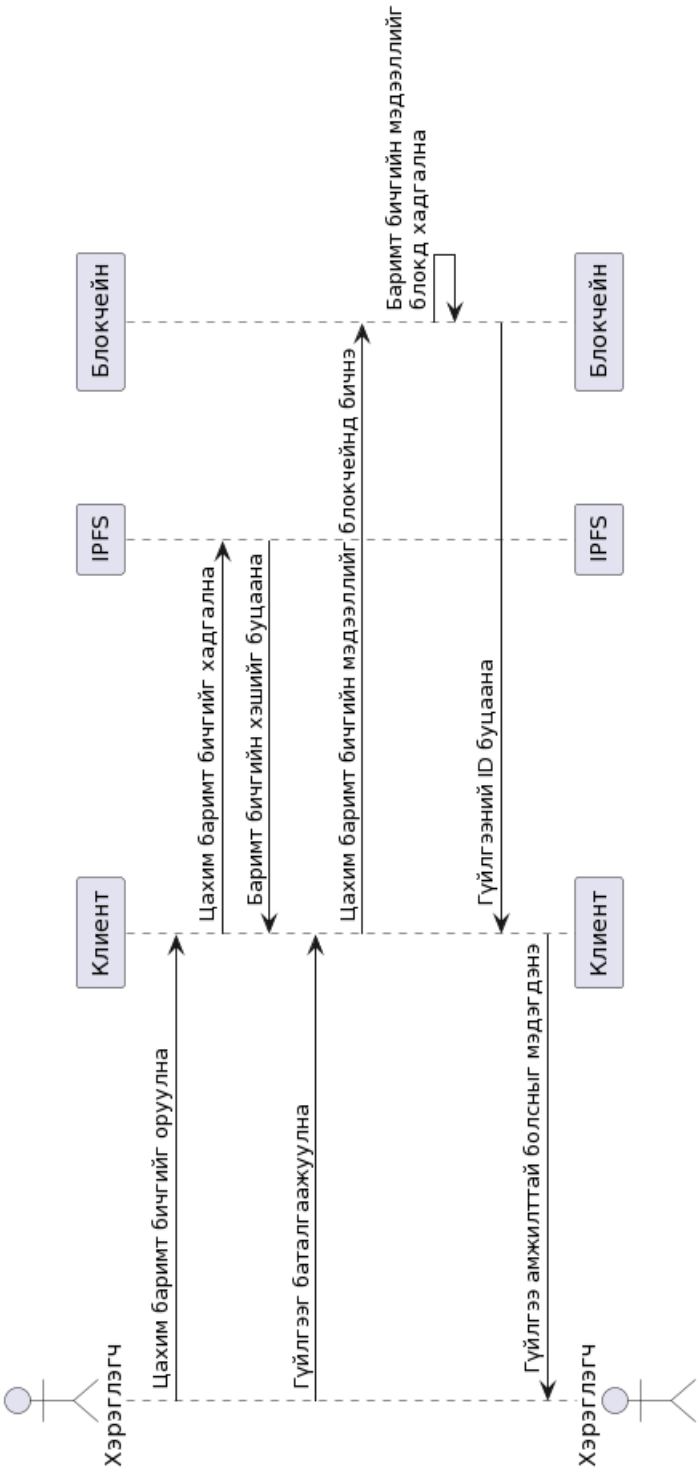
/ФБШ 70/ Энэ систем нь гамшгийн үед өгөгдөл алдагдахгүй байхын тулд найдвартай нөөцлөх, сэргээх механизмтай байх ёстой.

2.2 Use case диаграмм



Зураг 2.1: Use-case диаграм

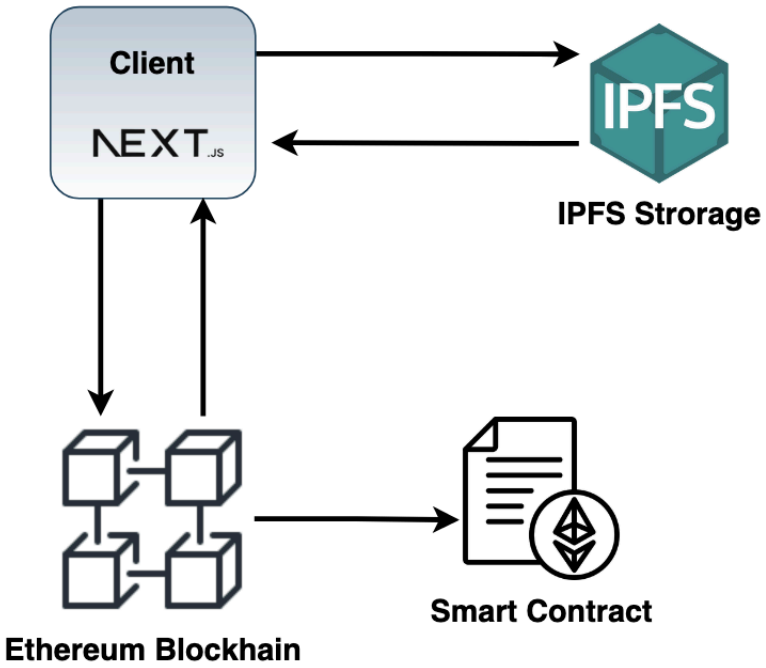
2.3 Хэрэглэгч цахим баримт бичиг оруулах sequence диаграмм



Зураг 2.2: Sequence диаграмм

2.4 Архитектур

Энэхүү төслийн фронт-энд хэсэг нь NextJS-н ашигласан тул сервер талын рендер хийж байгаа ба хэрэглэгчийн оруулсан цахим баримт бичиг болон лицензийн мэдээллийг этереум блокчэйн сүлжээнд бичих болон унших үйлдлийг хийх юм. Мөн хэрэглэгчийн оруулсан цахим баримт бичгийг IPFS сүлжээнд шифрлэлт хийн хадгална.



Зураг 2.3: Архитектурын зураг

3. СИСТЕМИЙН ХЭРЭГЖҮҮЛЭЛТ

3.1 Сонгосон технологи

3.1.1 React & Next.js

Declarative

React нь хэрэглэгчийн интерактив интерфейс бүтээхийг хялбарчилдаг. Аппликейшны state бүрд зориулсан энгийн бүтэц зохион байгуулахаас гадна, React нь өгөгдөл өөрчлөгдөхөд яг зөв компонентоо өөрчлөн рендер хийдэг. Declarative бүтэц нь кодыг тань debug хийхэд хялбар болгохоос гадна, ажиллагаа нь илүү тодорхой болдог.

Компонент-д тулгуурласан

Бие даан state-ээ удирддаг маш энгийн компонент бичиж, эдгээрийг хольж найруулан нарийн бүтэцтэй хэрэглэгчийн интерфейс бүтээх боломжтой. Компонентийн логик нь тэмплэйтээр бус JavaScript-ээр бичигддэг учраас өгөгдлийг апп хооронд хялбар дамжуулж, DOM-оос state-ээ тусад нь байлгаж чадна.

Next.js

Netflix, TikTok, Hulu, Twitch, Nike гэсэн орчин үеийн аваргууд ашигладаг энэхүү орчин үеийн фрэймворк нь React технологи дээр үндэслэгдсэн бөгөөд Frontend, Backend хоёр талд хоёуланд нь ажилладаг веб аппуудыг хийх чадвартайгаараа бусдаасаа давуу юм. Next.js-ийн үндсэн дизайн нь клиент болон сервер талын аль алиных давуу талыг ашиглаж чаддаг, ямар нэг дутагдалгүй веб сайтыг яаж хамгийн хурдан хялбар бүтээх вэ гэдгийг бодож тусгасан байдаг. Next.js нь сервер талд react компонентуудыг рендерлэн энгийн html, css, json файл болгон хувиргах замаар ажилладаг бөгөөд 2020 оноос олон нийтэд танигдсан JAMStack технологи

болон статик сайт, автоматаар статик хуудас үүсгэх, CDN deployment, сервергүй функц, тэг тохиргоо, файлын системийн рүүтинг (PHP-ээс санаа авсан), SWR (stale while revalidate), сервер талд рендерлэх зэрэг асар олон орчин үеийн шинэхэн технологиудыг бүгдийг хийж чаддаг анхны бүрэн веб фрэймворк гэж хэлж болно.

3.1.2 Ethereum блокчэйн

Төвлөрсөн бус, блокчэйн дээр суурилсан программуудыг хангамжийн платформ анх Ethereum-ийг 2013 онд программист Vitalik Buterin бичсэн бөгөөд 2015 онд олон нийтэд анх танилцуулагдсан юм. Ethereum нь бусад койныг бодвол зөвхөн арилжааны бус тус платформыг ашиглан smart contract буюу ухаалаг гэрээ үүсгэх боломжтой. Энэ нь энгийнээр хөгжүүлэгчдэд төвлөрсөн бус хэрэглээний программуудыг бүтээх, ажиллуулах боломжийг олгодог.

3.1.3 Hardhat

Hardhat нь ухаалаг гэрээг хөгжүүлэх орчин юм. Энэ нь Ethereum ухаалаг гэрээг бичих, туршихаас эхлээд байршуулах, дибаг хийх хүртэлх бүх амьдралын мөчлөгийг хөнгөвчлөх зорилготой юм. Hardhat нь Ethereum Virtual Machine (EVM) дээр бүтээгдсэн бөгөөд Ethereum, Polygon, Avalanche болон бусад EVM-тэй нийцтэй блокчэйнүүдийг дэмждэг.

3.1.4 Wagmi

Wagmi нь блокчэйнтэй ажиллахад шаардлагатай бүх зүйлийг агуулсан React Hook-ийн цуглуулга юм. Wagmi нь крифто түрийвч холбох, мэдээллийг авах, ухаалаг гэрээтэй харилцах гэх мэт үйлдлүүдийг хөнгөвчлөх боломжийг олгодог.

3.1.5 IPFS & Pinata

IPFS буюу Interplanetary File System нь peer-to-peer сүлжээн дэх файлуудыг хадгалах, хуваалцахад зориулагдсан төвлөрсөн бус протокол юм. Үндсэндээ IPFS нь файлуудыг жижиг

хэсгүүдэд хувааж, сүлжээний олон зангилаанд хадгалдаг. Энэ нь файлуудыг нэг байршилд хадгалдаггүй, харин сүлжээгээр тарааж байршуулдаг. Pinata нь төвлөрсөн бус бичиг баримт хадгалалтын сүлжээ болох Interplanetary File System (IPFS) дээр бүтээгдсэн үйлчилгээ юм. Pinata нь хөгжүүлэгчид болон хэрэглэгчдэд IPFS сүлжээнд өгөгдөл хадгалах, уншихад хялбар болгодог. Энэ нь IPFS дээр хадгалагдсан файлуудыг байршуулах, удирдах, хандахад зориулсан API болон бусад хэрэгслээр хангаснаар IPFS-тэй харилцах үйл явцыг хялбаршуулдаг.

3.1.6 Lit Protocol

Lit Protocol нь өөр өөр талууд эсвэл программуудын хооронд аюулгүй, хувийн харилцаа холбоо, өгөгдөл дамжуулах боломжийг олгодог төвлөрсөн бус сүлжээний протокол юм. Энэ нь одоо байгаа блокчэйн болон төвлөрсөн бус хадгалалтын шийдлүүдийн дээр нууцлал, хандалтын хяналтын давхаргыг хангах зорилготой юм.

Lit Protocol-н хэд хэдэн гол технологи, ойлголтууд:

- End-to-End Шифрлэлт: Протокол нь талуудын хооронд хуваалцсан өгөгдөл нь нууц хэвээр үлдэж, зөвхөн хүссэн хүлээн авагчид хандах боломжтой байхын тулд төгсгөлөөс төгсгөл хүртэл шифрлэлтийг ашигладаг.

- Хандалтын хяналтын нөхцөлүүд: Lit Protocol нь хандалтын хяналтын нөхцөлийн тухай ойлголтыг танилцуулсан бөгөөд энэ нь хэн тодорхой өгөгдөлд хандах эсвэл тодорхой үйлдлийг гүйцэтгэх боломжтой болохыг тодорхойлдог криптограф нотолгоо юм. Эдгээр нөхцөлүүд нь төвлөрсөн бус байдлаар хэрэгжиж, төвлөрсөн эрх мэдлийн хэрэгцээг арилгадаг.

- Төвлөрсөн бус таних тэмдэг: Протокол нь аюулгүй харилцаа холбоо, мэдээлэл солилцох үйл ажиллагаанд оролцож буй талуудыг төлөөлөхийн тулд Ethereum хаяг эсвэл блокчэйнд суурилсан бусад таних тэмдэг зэрэг төвлөрсөн бус таних тэмдгийг ашигладаг.

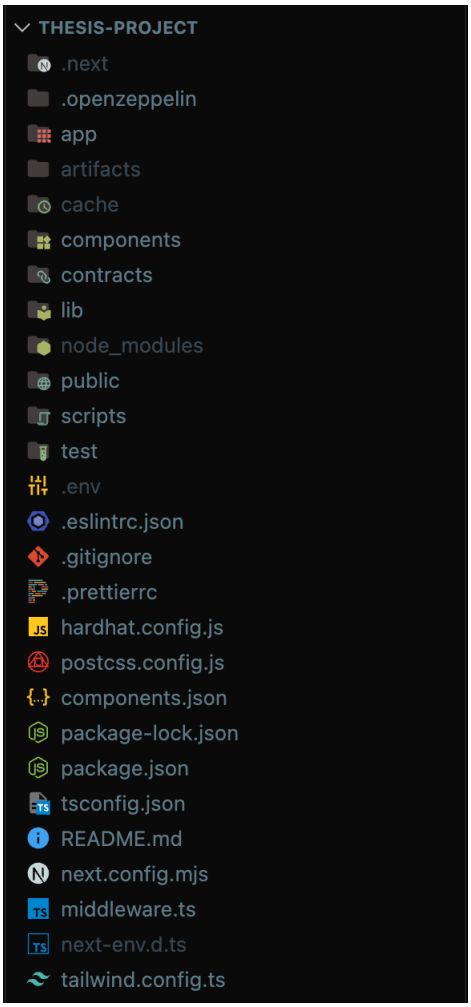
Энэхүү судалгааны ажлын хүрээнд хэрэглэгчийн байршуулсан файлуудыг шифрлэх, файлуудад

хандах хандалтыг хянахад найдвартай, төвлөрсөн бус шийдлээр хангах зорилгоор Lit Protocol-ийг сонгосон. Lit Protocol-тэй нэгтгэснээр систем нь файлуудыг найдвартай хадгалж, хандалтын хяналтын тодорхой нөхцөөл дээр үндэслэн зөвхөн эрх бүхий этгээдэд хандах боломжтой.

3.2 Хөгжүүлэлт

3.2.1 Хөгжүүлэлтийн орчныг бэлдэх

Энэхүү судалгааны ажлын практик хэсэгт би NextJS, Hardhat, Pinata, Wagmi, Tailwind CSS зэргийг ашиглан хөгжүүлэлт хийх билээ. NextJS нь монолитик төсөл хийхэд тохиромжтой ба төслийн ухаалаг гэрээ хөгжүүлэлт, клайнт талуудыг нэг repository-д хадгалж байгаа. Version Control System-ээр Github-г сонгосон юм. Кодын фолдер бүтэц нь дараах байдлаар байна.



Зураг 3.1: Фолдерийн бүтэц

- **components** - React компонентууд
- **lib** - Хэрэглэгчийн талын шаардлагатай код туслах функцууд

- **app** - NextJS дээрх хуудаснууд
- **public** - Статик зураг, файлууд
- **scripts** - Ухаалаг гэрээний хөгжүүлэлтийн холбоотой javascript файлууд
- **contracts** - Ухаалаг гэрээний файлууд

3.2.2 Ухаалаг гэрээн хөгжүүлэлт

Миний төсөл нэг ухаалаг гэрээнээс бүтнэ. Уг ухаалаг гэрээ нь цахим файлууд болон тэдгээртэй холбоотой лицензүүдийг төлөөлдөг Файл ба Лиценз гэсэн хоёр бүтцийг тодорхойлсон. Файл бүтэц нь id, эзэмшигчийн хаяг, файлын нэр, тайлбар, ангилал, файлын хэш, үүсгэсэн хугацааны зэрэг атрибутуудыг агуулна. Лиценз бүтэц нь лицензийн дугаар, эзэмшигчийн хаяг, файлын нэр, тайлбар, ангилал, файлын хэш, үүсгэсэн хугацааны зэрэг атрибутуудыг агуулна. Мөн дараах функцүүдтэй:

- **createFile**: Цахим баримт бичгийн мэдээллийг бичих
- **issueLicense**: Лицензийн мэдээллийг бичих
- **getAllPublicFiles**: Оруулсан бүх цахим баримт бичгийн авах
- **getAllUserFiles**: Хэрэглэгчийн оруулсан цахим баримт бичгүүдийг авах
- **getAllUserLicenses**: Хэрэглэгчийн эзэмшиж буй лицензүүдийг авах
- **validateLicense**: Лицензийн дугаараар лицензийг шалгах
- **getPublicFileById**: Цахим баримт бичгийг авах id-гаар нь авах
- **getMarketplaceFiles**: Лиценз авах боломжтой цахим баримт бичгүүдийг авах
- **generateUniqueLicense**: Лицензд өвөрмөц дугаар бий болгох

3.2.3 Ухаалаг гэрээг блокчэйнд байршуулах

```
const { ethers } = require('hardhat');

async function deployContract() {
  let contract;

  try {
    contract = await ethers.deployContract('LicenseMarketplace');
    await contract.waitForDeployment();

    console.log('Contracts deployed successfully.');
    return contract;
  } catch (error) {
    console.error('Error deploying contracts:', error);
    throw error;
  }
}

async function main() {
  let contract;

  try {
    contract = await deployContract();
    await saveContractAddress(contract);

    console.log('Contract deployment completed successfully.');
  } catch (error) {
    console.error('Unhandled error:', error);
  }
}

main().catch((error) => {
  console.error('Unhandled error:', error);
  process.exitCode = 1;
});
```

Код 3.1: deploy

3.2.4 Хэрэглэгч талын хөгжүүлэлт (Front-end)

Уг код нь хэрэглэгчийн оруулах цахим баримт бичгийн мэдээллийг блокчэйнд бичнэ.

```
const [file, setFile] = useState<File | null>(null);
const { connect } = useConnect();
const { toast } = useToast();
const fileInputRef = useRef<HTMLInputElement>(null);

const { writeContract, isPending, error, data: hash, isError:
   issueError } = useWriteContract();
const { isLoading, isSuccess, isError } =
   useWaitForTransactionReceipt({
```

```
  hash,
});
const { isConnected } = useAccount();

async function onSubmit(data: z.infer<typeof formSchema>) {
  if (!isConnected) {
    connect({ connector: injected() });
  }
  try {
    if (!file) {
      return;
    }
    const res = await pinFileToIPFS(file);

    if (!res.isDuplicate) {
      writeContract({
        abi: licenseValidationAbi.abi,
        address: licenseValidationContract.contractAddress as `0
           x${string}`,
        functionName: 'createFile',
        args: [data.fileName, data.description, 'PDF', res.
           IpfsHash, data.isPublic],
      });

      if (isSuccess) {
        form.reset();
      }
    } else {
      if (fileInputRef.current) {
        fileInputRef.current.value = '';
      }

      toast({
        variant: 'destructive',
        description: 'This␣file␣has␣already␣been␣uploaded.',
      });
    }
  } catch (error) {
    console.error(error);
  }
}
```

Код 3.2: Блокчэйнд бичих

Энэ функц нь хэрэглэгчийн оруулсан баримт бичгийг IPFS-д байршуулна.

```
async function pinFileToIPFS(file: File): Promise<any> {
   const formData = new FormData();
   formData.append('file', file);

   const res = await fetch('https://api.pinata.cloud/pinning/
      pinFileToIPFS', {
      method: 'POST',
      headers: {
```

```
            pinata_api_key: process.env.NEXT_PUBLIC_PINATA_API_KEY!,
            pinata_secret_api_key: process.env.
               NEXT_PUBLIC_PINATA_API_SECRET!,
        },
        body: formData,
    });
    return res.json();
}
```

Код 3.3: Файл IPFS-д байршуулах

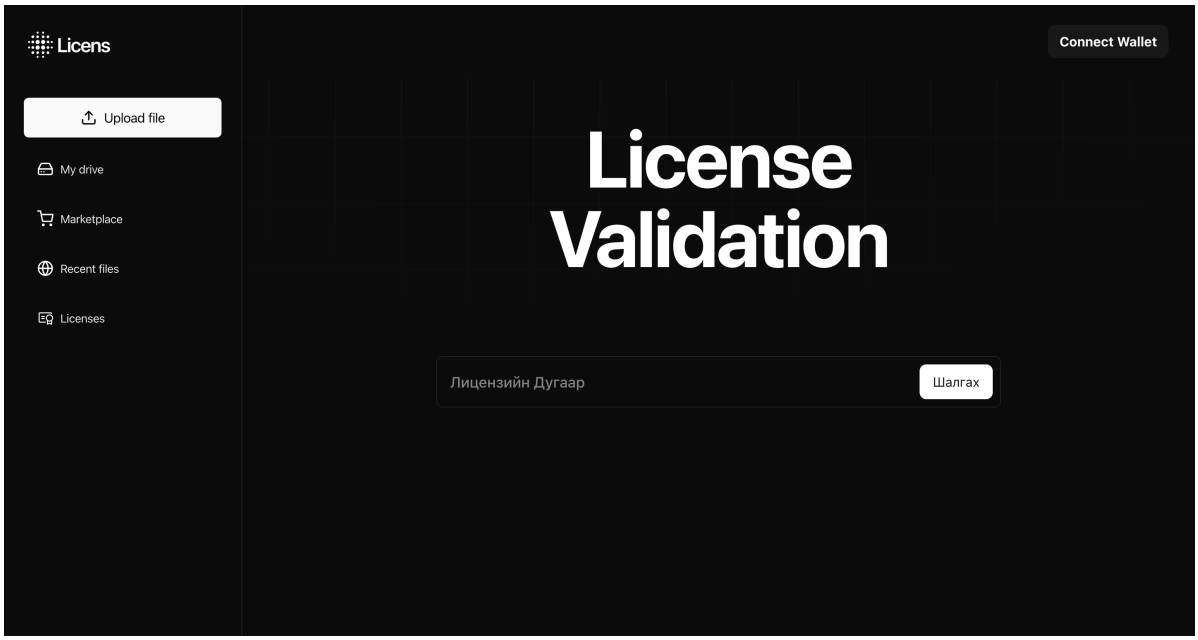
Уг код нь хэрэглэгчийн оруулсан цахим баримт бичгүүдийн мэдээллийг блокчэйнээс уншина.

```
const { address } = useAccount();
const {
  data: userFiles,
  isLoading,
  error,
} = useReadContract({
  address: licenseValidationContract.contractAddress as `0x${
     string}`,
  abi: licenseValidationAbi.abi,
  functionName: 'getAllUserFiles',
  account: address,
}) as { data: UploadedFile[]; isLoading: boolean; error: any };
```

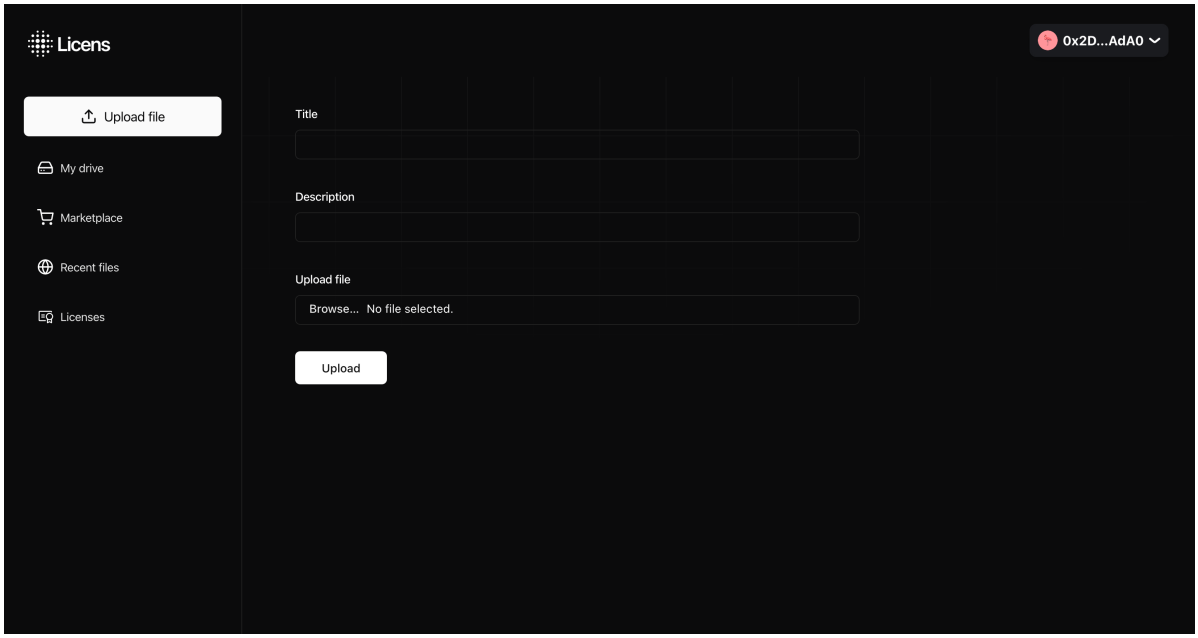
Код 3.4: Блокчэйнээс унших

3.2.5 Үр дүн

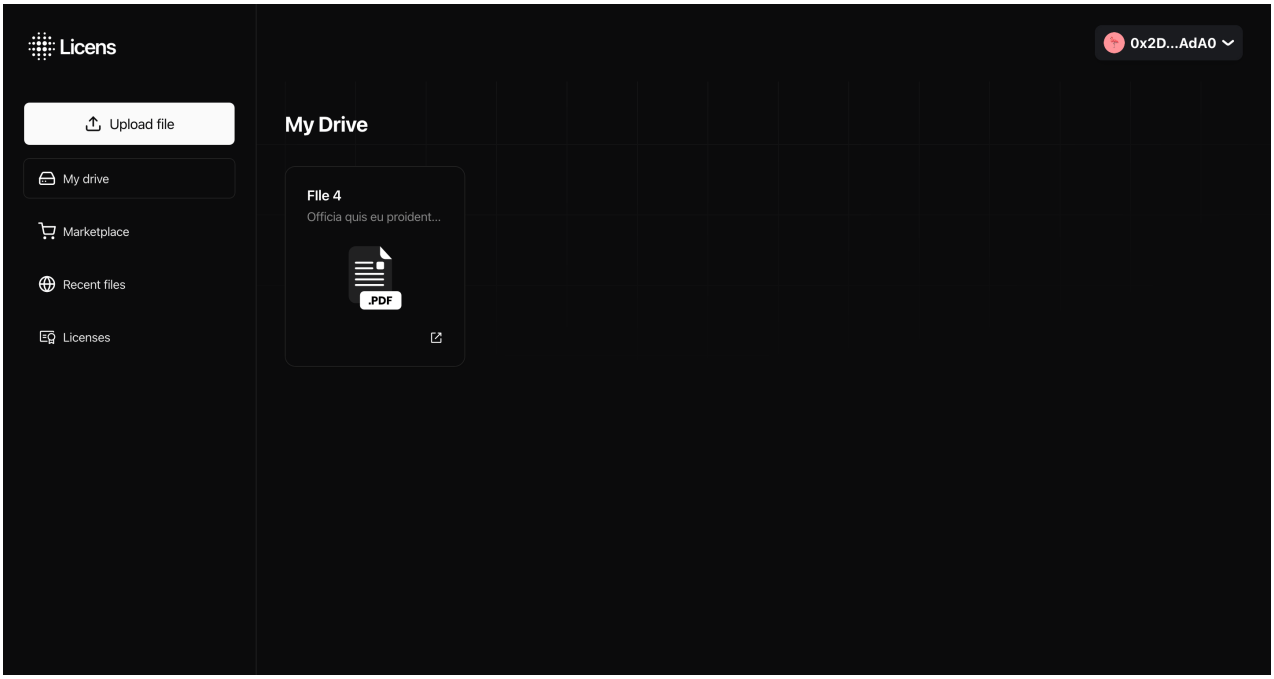
Төслийн практик ажлын үр дүнд бүтээгдсэн системийн интерфейс дараах байдлаар харагдана.



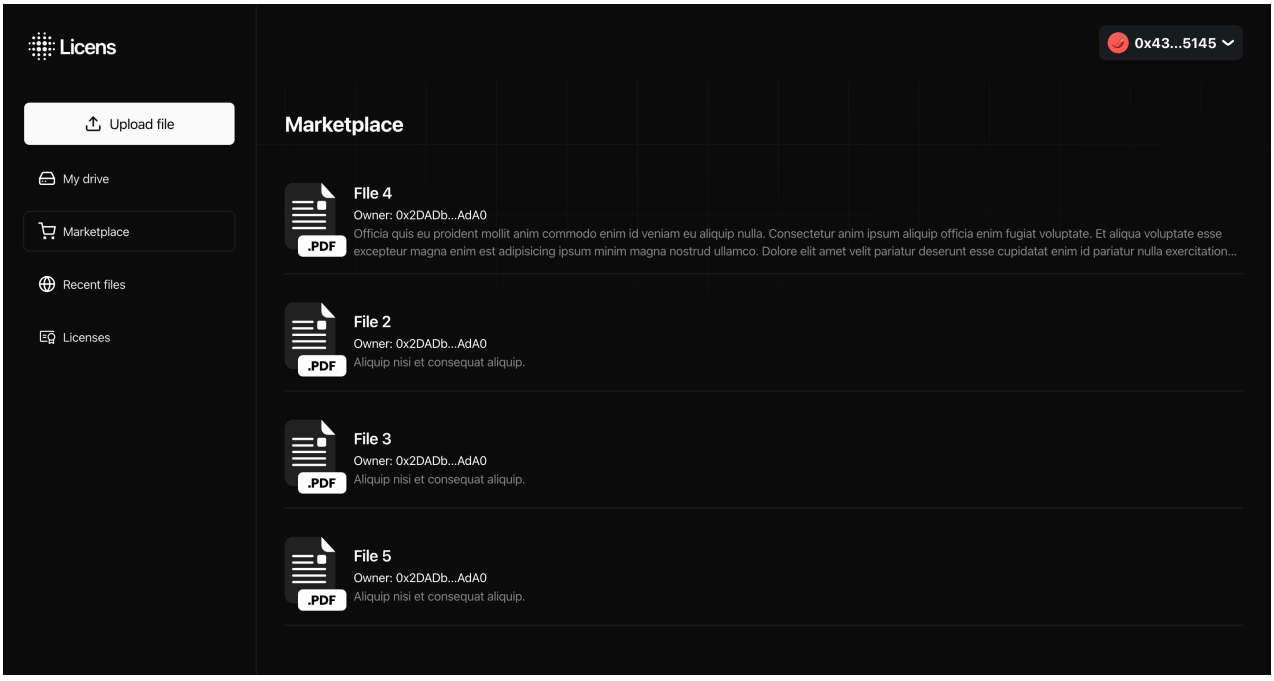
Зураг 3.2: Нүүр хуудас



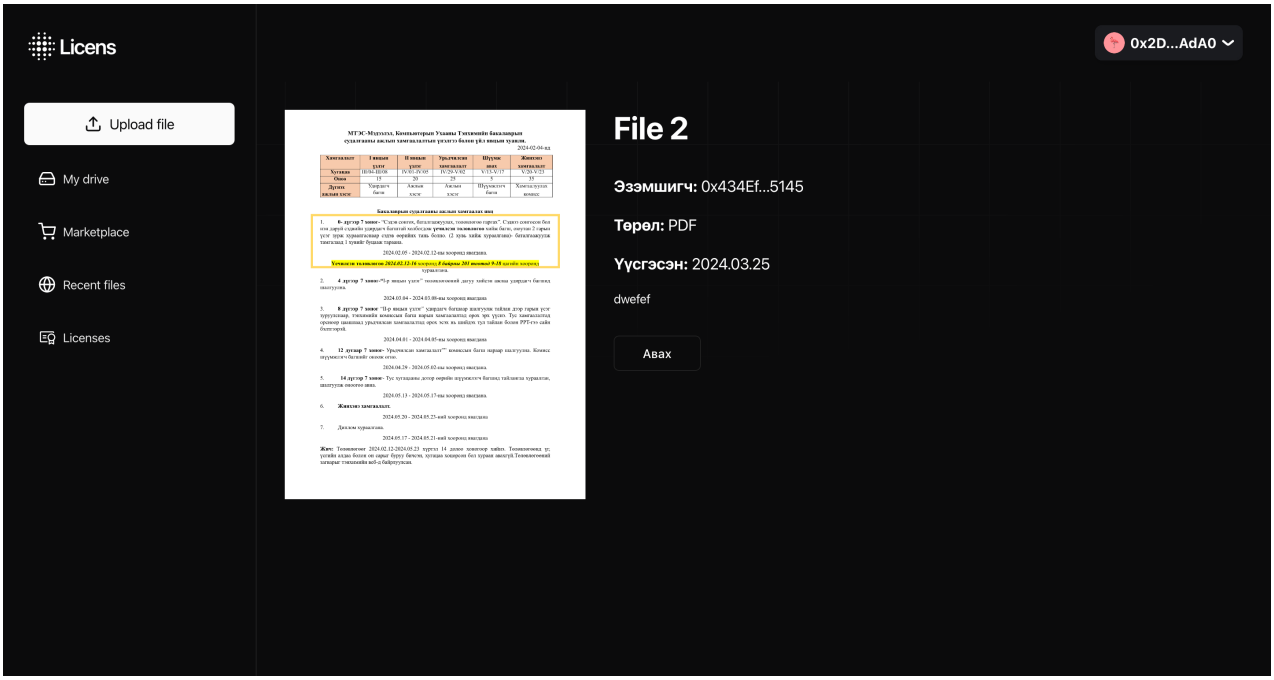
Зураг 3.3: Цахим бичиг баримт оруулах



Зураг 3.4



Зураг 3.5


Licens
0x2D...AdA0
Upload file
My drive
Marketplace
Recent files
Licenses
File 2
Эзэмшигч: 0x434Ef...5145
Төрөл: PDF
Үүсгэсэн: 2024.03.25
dwefef
Авах

Зураг 3.6

4. ДҮГНЭЛТ

Энэхүү судалгааны ажлаар блокчэйн технологи болон дижитал эрхийн менежментийн талаар судласан. Энэхүү судалж суралцсан мэдлэгээ ашиглан практикт цахим баримт бичгийн лицензийн систем бүтээхийг зорилоо. Хөгжүүлэлтийн явцад блокчэйн сүлжээнд цахим баримт бичгийг байршуулах, лиценз олгох, лицензийн баталгаажуулалт зэрэг янз бүрийн функцүүдыг хэрэгжүүлж, туршиж үзсэн. Үр дүнд нь орчин үеийн шинэлэг блокчэйн технологиудтай танилцсан ба бүтээгдэхүүний шаардлагыг гаргаж ухаалаг гэрээ бичихээс эхлээд эцсийн хэрэглэгчид хүрэх чанарын шаардлагыг хангаж блокчэйн технологийг ашиглан найдвартай, ил тод, төвлөрсөн бус системийг бүтээлээ. Цаашид өөрийн бичсэн ухаалаг гэрээ болон системээ хөгжүүлэн зөвхөн баримт бичиг бус дуу хөгжим, видео, цахим ном зэргийн цахим бүтээлийн лицензийн систем болгохыг зорино.

Bibliography

[1] Adam Hayes, Blockchain Facts: What Is It, How It Works, and How It Can Be Used. (December 15, 2023) https://www.investopedia.com/terms/b/blockchain.asp

[2] Scott Nevil, Distributed Ledger Technology (DLT): Definition and How It Works. (May 31, 2023) https://www.investopedia.com/terms/d/distributed-ledger-technology-dlt.asp

[3] Sundararajan S. UN Agencies Turn to Blockchain In Fight Against Child Trafficking. (Nov 13, 2017) https://www.coindesk.com/markets/2017/11/13/un-agencies-turn-to-blockchain-in-fight-against-child-trafficking/

[4] Zug Digital ID: Blockchain Case Study for Government Issued Identity. https://www.investopedia.com/terms/b/blockchain.asp

[5] What is digital rights management (DRM)?. https://business.adobe.com/blog/basics/digital-rights-management

А. ҮЕЧИЛСЭН ТӨЛӨВЛӨГӨӨ

Баглав.

МКУГ-ийн эрхлэгч:...../Дэд профессор. Ч.Алтангэрэл/ 2024 оны 02 сарын 12 нд

Монгол нэр: **Блокчэйн сууръ лицэнз баталгаажуулалт**
Англи нэр: **Licence validation with blockchain**
Сэдэвт бакалаврын судалгааны ажлын **7 хоногийн үечилсэн төлөвлөгөө**

Хугацаа: **2024.02.13**-аас **2024.05.23** хүртэл **14** долоо хоног

№	Долоо хоног / Хийх ажил		0	1	2	3	4 Явц I	5	6	7	8 Явц II	9	10	11	12 Урьдчилсан хамгаалалт	13 Засаж сайжруулах	14 Шүүмж	Жишээ хамгаалалт
1	Судалгаа	Сэдвийн	2 сарын 05-08 хооронд үечилсэн төлөвлөгөөө															
		Технологийн																
2	Системийн шинжилгээ зохиомж	Системийн шинжилгээ																
		Шаардлагын тодорхойлолт																
		Системийн UI/UX дизайн																
3	Системийн хэрэгжүүлэлт	Үндсэн код																
		Нэмэлтүүд																
4	Туршилт	Тест																
		Сайжруулалт																
5	Тайлан	Явцын																
		Эцсийн																

Тайлбар: Төслийг хэрэгжүүлэх төлөвлөгөөг 7 хоногийн давтамжтайгаар хийж тод харагдуулж будаж тэмдэглэнэ. Хийх ажил дээ хэсэгтэй байвал уг ажилд зарцуулах хугацаа хуваан төлөвлөж болно. Ажлын эхлэх төгсөх хугацаа хооронд давхцаж болно. Ажлын гүйцэтгэлийг дүгнэж тэмдэглэсэн хийх боломжтой байхаар "Үхнөг" баганыг үүсгэнэ.

Зөвшөөрсөн: Удирдагч багш ...../Дэд профессор Ч.Алтангэрэл/

Боловсруулсан: Оюутан ...../Программ хангамж-4, Э.Жавхлан/
Оюутны ID: 20b1num0649
Холбогдох утас: 88242310

Зураг А.1: Удирдагчийн үнэлгээ дүгнэлт

В. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract LicenseMarketplace {
   address public owner;

   struct File {
        uint256 id;
        address fileOwner;
        string fileName;
        string description;
        string category;
        string fileCid;
        string imgUrl;
        uint256 fileSize;
        uint256 createdAt;
   }

   struct License {
        uint256 licenseNumber;
        uint256 fileId;
        address fileOwner;
        address buyer;
        string fileName;
        string description;
        string category;
        string fileCid;
        string imgUrl;
        uint256 fileSize;
        uint256 createdAt;
   }

   struct LicenseRequest {
        uint256 requestId;
        uint256 fileId;
        address requester;
        address fileOwner;
        bool isApproved;
   }

      mapping(address => File[]) private userFiles;
      mapping (address => License[]) private fileLicenses;
      mapping (uint256 => LicenseRequest) private licenseRequests
         ;

      mapping(uint256 => bool) public usedLicenses;

      File[] public publicFiles;

      uint256 public fileId;
```

```
50        uint256 public requestId;
51
52        event FileShared(address indexed  owner, string fileName);
53        event FileLicense(address indexed  owner, address indexed
              buyer, string fileName);
54        event LicenseRequestCreated(uint256 indexed requestId,
              uint256 fileId, address requester, address fileOwner);
55        event LicenseRequestApproved(uint256 indexed requestId,
              uint256 fileId, address requester, address fileOwner);
56        event LicenseRequestRejected(uint256 indexed requestId,
              uint256 fileId, address requester, address fileOwner);
57
58        modifier onlyOwner(){
59            require(msg.sender == owner,  "Only␣the␣owner␣can␣call␣
                  this␣function");
60            _;
61        }
62
63        constructor() {
64            owner = msg.sender;
65        }
66
67        function createFile(string memory _fileName, string memory
              _description, string memory _category,  string memory
              _fileCid, uint256  _fileSize, string memory _imgUrl)
              external{
68            fileId++;
69
70            File memory newFile = File({
71                id: fileId,
72                fileOwner: msg.sender,
73                fileName: _fileName,
74                description: _description,
75                category: _category,
76                fileCid: _fileCid,
77                imgUrl: _imgUrl,
78                fileSize: _fileSize,
79                createdAt: block.timestamp
80            });
81
82            userFiles[msg.sender].push(newFile);
83            publicFiles.push(newFile);
84
85            emit FileShared(msg.sender, _fileName);
86        }
87
88        function requestLicense(uint256 _fileId, address _fileOwner
              ) external {
89            require(!isFileOwnedOrLicensed(msg.sender, _fileId), "
                  User␣already␣owns␣or␣has␣a␣license␣for␣this␣file");
90
91            requestId++;
92            LicenseRequest memory newRequest = LicenseRequest({
93                requestId: requestId,
```

```
94              fileId: _fileId,
95              requester: msg.sender,
96              fileOwner: _fileOwner,
97              isApproved: false
98          });
99
100         licenseRequests[requestId] = newRequest;
101         emit LicenseRequestCreated(requestId, _fileId, msg.
                sender, _fileOwner);
102     }
103
104
105     function approveLicenseRequest(uint256 _requestId) external
            {
106         LicenseRequest storage request = licenseRequests[
                _requestId];
107         require(request.fileOwner == msg.sender, "Only␣the␣file␣
                owner␣can␣approve␣the␣request");
108         require(!request.isApproved, "Request␣already␣approved")
                ;
109
110         request.isApproved = true;
111
112         File memory file = getPublicFileById(request.fileId);
113         uint256 licNum = generateUniqueLicense();
114
115         License memory newLicense = License({
116             licenseNumber: licNum,
117             fileId: file.id,
118             fileOwner: file.fileOwner,
119             buyer: request.requester,
120             fileName: file.fileName,
121             description: file.description,
122             category: file.category,
123             fileCid: file.fileCid,
124             imgUrl: file.imgUrl,
125             fileSize: file.fileSize,
126             createdAt: block.timestamp
127         });
128
129         fileLicenses[request.requester].push(newLicense);
130
131         emit FileLicense(file.fileOwner, request.requester, file
                .fileName);
132         emit LicenseRequestApproved(_requestId, request.fileId,
                request.requester, request.fileOwner);
133     }
134
135     function rejectLicenseRequest(uint256 _requestId) external
            {
136         LicenseRequest storage request = licenseRequests[
                _requestId];
137         require(request.fileOwner == msg.sender, "Only␣the␣file␣
                owner␣can␣reject␣the␣request");
```

```
138             require(!request.isApproved, "Request already approved");
139
140             delete licenseRequests[_requestId];
141             emit LicenseRequestRejected(_requestId, request.fileId,
                    request.requester, request.fileOwner);
142         }
143
144         function getAllPublicFiles() external view returns(File[]
                memory) {
145             return publicFiles;
146         }
147
148         function getAllUserFiles() external view returns(File[]
                memory) {
149             return userFiles[msg.sender];
150         }
151
152         function getAllUserLicenses() external view returns(License
                [] memory) {
153             return fileLicenses[msg.sender];
154         }
155
156      function getPublicFiles() external view returns (File[]
            memory) {
157           return publicFiles;
158      }
159
160         function generateUniqueLicense() internal returns (uint256)
                {
161             uint256 randomNumber = uint256(keccak256(abi.
                    encodePacked(block.timestamp, blockhash(block.number)
                    , msg.sender)));
162             uint256 license = randomNumber % 10000000000;
163
164             while (usedLicenses[license]) {
165                 randomNumber = uint256(keccak256(abi.encodePacked(
                        randomNumber, block.timestamp)));
166                 license = randomNumber % 10000000000;
167             }
168
169             usedLicenses[license] = true;
170
171             return license;
172         }
173
174     function validateLicense(uint256 licenseNumber) external view
            returns (bool) {
175             return usedLicenses[licenseNumber];
176     }
177
178     function getPublicFileById(uint256 _id) public view returns (
            File memory) {
179             for (uint256 i = 0; i < publicFiles.length; i++) {
```

```
180             if (publicFiles[i].id == _id) {
181                 return publicFiles[i];
182             }
183         }
184         revert("Public file not found");
185     }
186 
187     function getUserLicenseRequests() external view returns (
        LicenseRequest[] memory) {
188         uint256 count = 0;
189         for (uint256 i = 1; i <= requestId; i++) {
190             if (licenseRequests[i].requester == msg.sender) {
191                 count++;
192             }
193         }
194 
195         LicenseRequest[] memory userRequests = new
            LicenseRequest[](count);
196         uint256 index = 0;
197         for (uint256 i = 1; i <= requestId; i++) {
198             if (licenseRequests[i].requester == msg.sender) {
199                 userRequests[index] = licenseRequests[i];
200                 index++;
201             }
202         }
203 
204         return userRequests;
205     }
206 
207     function getFileOwnerLicenseRequests() external view returns (
        LicenseRequest[] memory) {
208         uint256 count = 0;
209         for (uint256 i = 1; i <= requestId; i++) {
210             if (licenseRequests[i].fileOwner == msg.sender) {
211                 count++;
212             }
213         }
214 
215         LicenseRequest[] memory ownerRequests = new
            LicenseRequest[](count);
216         uint256 index = 0;
217         for (uint256 i = 1; i <= requestId; i++) {
218             if (licenseRequests[i].fileOwner == msg.sender) {
219                 ownerRequests[index] = licenseRequests[i];
220                 index++;
221             }
222         }
223 
224         return ownerRequests;
225     }
226 
227     function isFileOwnedOrLicensed(address _user, uint256 _fileId)
        public view returns (bool) {
228         for (uint256 i = 0; i < userFiles[_user].length; i++) {
```

```
229             if (userFiles[_user][i].id == _fileId) {
230                 return true;
231             }
232         }
233 
234         for (uint256 j = 0; j < fileLicenses[_user].length; j++)
                {
235             if (fileLicenses[_user][j].fileId == _fileId) {
236                 return true;
237             }
238         }
239 
240         return false;
241     }
242 
243 
244     function getFileId() external view returns (uint256) {
245         return fileId + 1;
246     }
247 }
```

Код B.1: Ухаалаг гэрээ