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:

Орчин үеийн цахим орчинд оюуны өмч болон цахим бүтээгдэхүүнийг хамгаалах, баталгаажуулах нь нэн чухал болсон.

Цахим хөрөнгийн менежментийн хүрээнд дижитал контентын бүрэн бүтэн байдал, аюулгүй байдал, зөв хуваарилалтыг хангахын тулд лицензийн баталгаажуулалтын найдвартай механизм шаардлагатай байна. Лицензийн баталгаажуулалтын уламжлалт арга барил нь төвлөрсөн хяналт, залилан мэхлэлтэд өртөмтгий байх, ил тод бус байх зэрэг бэрхшээлтэй тулгардаг. Эдгээр дутагдалтай талуудын хариуд блокчэйн технологийг нэгтгэх нь анхаарал татахуйц шийдэл болж байна.

Энэхүү судалгааны сэдэл нь дижиталчлал болон онлайн контентын хэрэглээ нэмэгдэж байгаа нөхцөлд дижитал хөрөнгийн менежментийн найдвартай шийдлүүдийн хэрэгцээ нэмэгдэж байгаатай холбоотой юм. Блокчэйн технологийг DRM зарчимтай хослуулснаар бид уламжлалт дижитал хөрөнгийн удирдлагын системийн дутагдлыг арилгах, илүү бат бөх, өргөтгөх боломжтой шийдлийг санал болгохыг зорьж байна. Энэхүү аргын цаад учир шалтгаан нь дижитал хөр өнгийн бүрэн бүтэн байдал, үнэн зөв, мөрдөх чадварыг сайжруулж болох төвлөрлийг сааруулах, ил тод байдал, криптографийн аюулгүй байдал зэрэг блокчэйний салшгүй давуу талуудад оршдог.

Зорилго:

Энэхүү судалгааны ажлаар хэрэглэгчид блокчейн технологиор дамжуулан цахим бичиг баримтыг хамгаалах, хуваалцах, хандах зөвшөөрөл олгох цахим бичиг баримтын лицензийн систем хөгжүүлэх зорилго тавьсан.

Зорилт:

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

1. ОНОЛЫН СУДАЛГАА

1.1 Блокчейн 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.

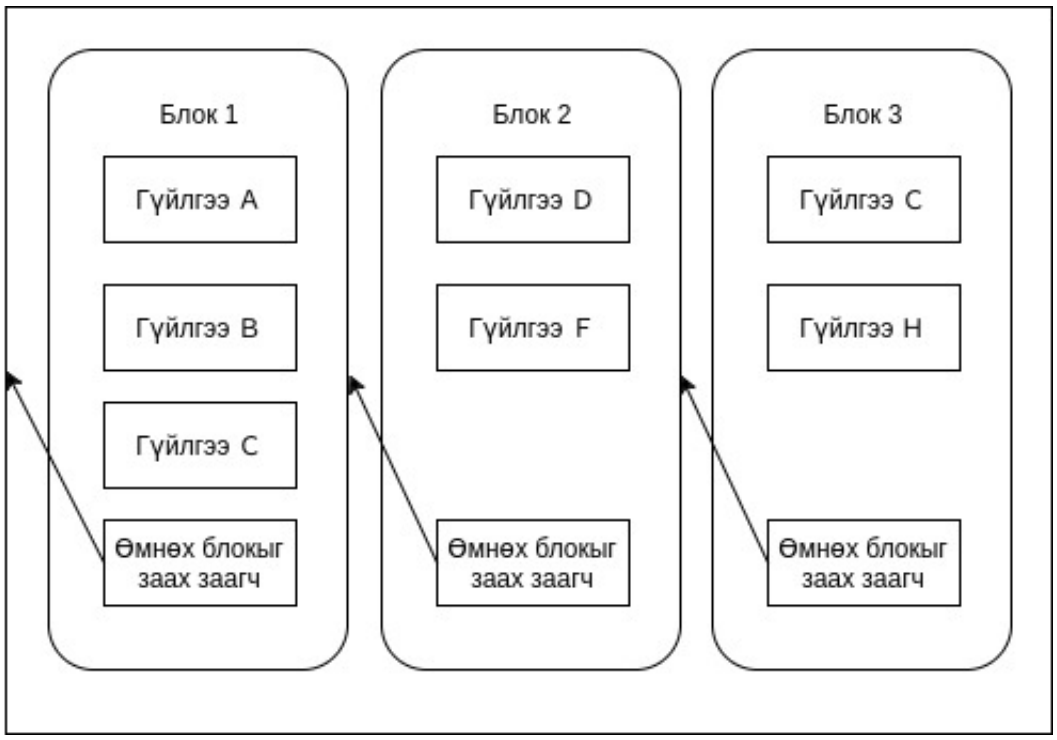
1.2 Блокчэйний онцлог

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 хэйш 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: Блокчейний өгөгдөлийн бүтэц

1.3 Блокчэйний нууцлалын технологи

1.3.1 Криптограф хэш

Криптограф хэш функц нь оруулсан өгөгдлийн уртаас үл хамааран тогтсон урттай хэйш утгуудыг буцаадаг. Оролтын зөвхөн нэг тэмдэгт өөрчлөгдөхөд гаралтын хэйш 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 Mac-ын командлайн-аар дараах командыг оруулбал, SHA-256 hash функцийн утгыг хялбархан олох болно.

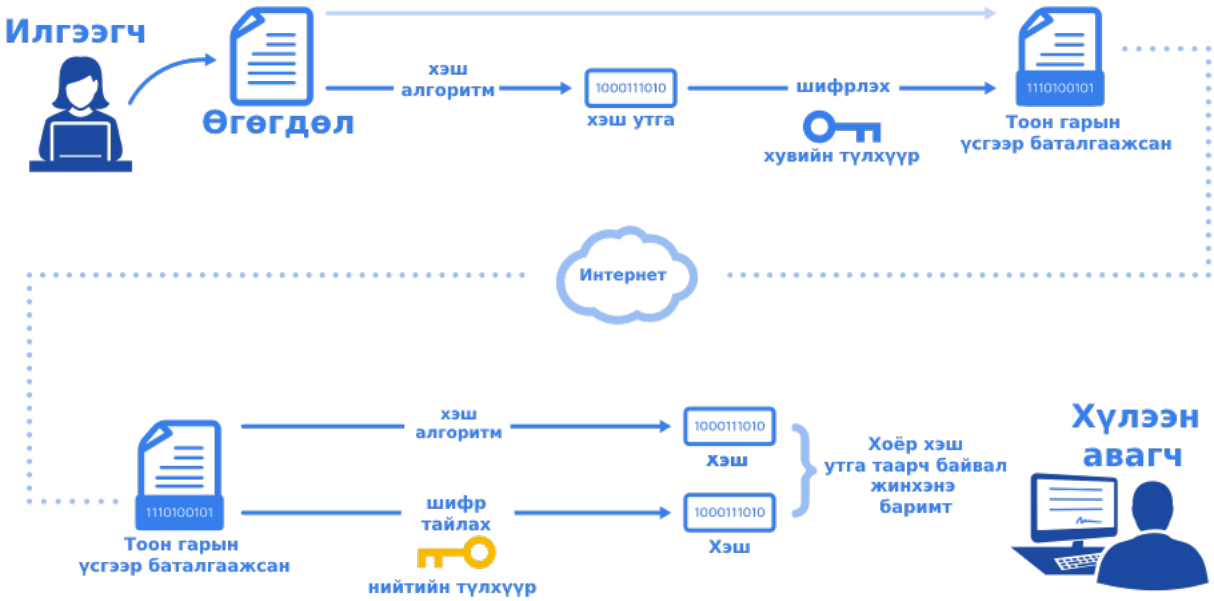
```
❯ echo "Hello World" | openssl sha256
SHA2-256(stdin)= d2a84f4b8b650937ec8f73cd8be2c74add5a911ba64df27458ed8229da804a26
❯ echo "Hallo World" | openssl sha256
SHA2-256(stdin)= e6c1396639c0b79bebc94e4448cfe2700b871d45d0d38d98df6ee9da3f09d35c
```

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 блокын 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 компьютерийн 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. Блокчейн 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 IPFS технологи

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

гэж нэрлэдэг жижиг хэсгүүдэд хувааж, сүлжээний олон зангилаанд хадгалдаг. Энэ нь файлуудыг нэг байршилд хадгалдаггүй, харин сүлжээгээр тарааж байршуулдаг.

2. СИСТЕМИЙН СУДАЛГАА, ЗОХИОМЖ

2.1 Функционал шаардлагууд

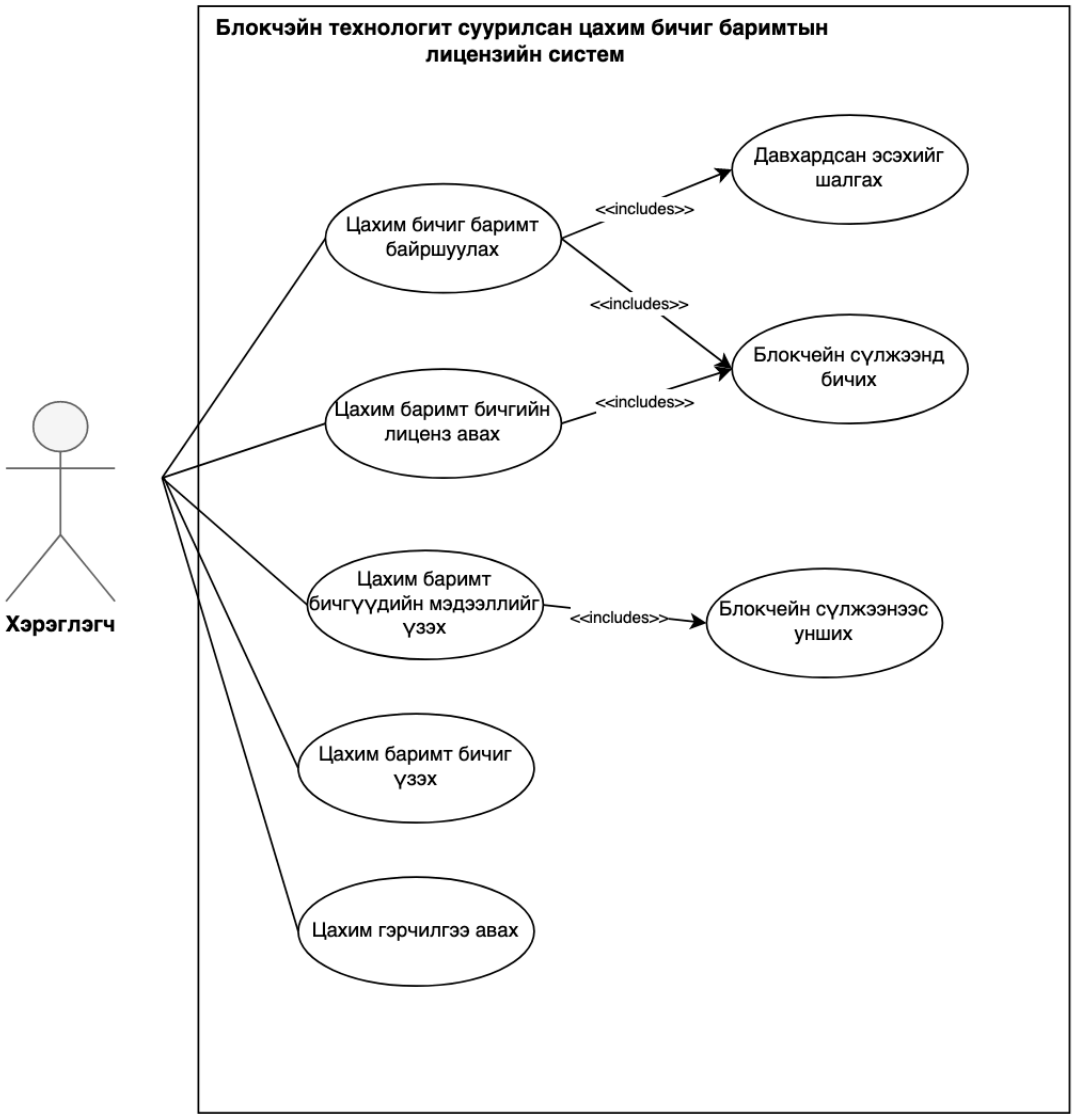
- **ФШ 100:** Систем нь цахим баримт бичгүүд болон лицензийн талаарх мэдээллийг найдвартай байдлыг хадгалахын тулд Ethereum блокчэйн-тэй харилцах ёстой.
- **ФШ 200:** Цахим баримт бичгүүд эзэмших, лиценз олгох асуудлыг зохицуулахын тулд ухаалаг гэрээг блокчейн дээр байршуулж, удирдах ёстой.
- **ФШ 300:** Систем нь цахим баримт бичгүүдийг байршуулах, лиценз авахын тулд хэрэглэгчийн крипто хэтэвчтэй холбогдсон байх ёстой.
- **ФШ 400:** Систем нь хэрэглэгчид вэб аппликейшны интерфейсээр дамжуулан PDF файлуудыг байршуулах боломжтой байх ёстой.
- **ФШ 500:** Систем нь цахим баримт бичгүүдийг байршуулахдаа баримт бичгийн мэдээллийг бүртгэх ёстой.
- **ФШ 600:** Систем нь цахим баримт бичиг байршуулах үед систем нь файлын хэшийг тооцоолж, хадгалалтыг үргэлжлүүлэхийн өмнө давхардсан эсэхийг шалгах ёстой.
- **ФШ 700:** Систем нь цахим баримт бичиг байршуулах үед систем нь файлын хэшийг тооцоолж, хадгалалтыг үргэлжлүүлэхийн өмнө давхардсан эсэхийг шалгах ёстой.
- **ФШ 800:** Хэрэглэгчид байршуулсан цахим баримт бичгүүдийн лицензийг авах боломжтой байх ёстой.
- **ФШ 900:** Цахим баримт бичгийн лиценз авахад лицензед өвөрмөц дугаар олгож, блокчейн дээр хадгалах ёстой.

- **ФШ 1000:** Систем нь хэрэглэгчид лиценз авсны дараа лицензийн дугаар, файлын мэдээлэл зэрэг лицензийнхээ дэлгэрэнгүй мэдээллийг агуулсан цахим гэрчилгээ авах ёстой.
- **ФШ 1100:** Хэрэглэгчид систем дээр байрлуулсан цахим баримт бичгүүдийн дэлгэрэнгүй мэдээллийг үзэх боломжтой байх ёстой.

2.2 Функционал бус шаардлагууд

- **ФБШ 100:** Блокчейн технологи нь өгөгдлийн бүрэн бүтэн байдлыг хангаж, лицензийн мэдээллийг зөвшөөрөлгүй өөрчлөхөөс сэргийлнэ.
- **ФБШ 200:** Систем нь гүйцэтгэлийн бууралтгүйгээр олон тооны хэрэглэгчид болон лицензүүдийг зохицуулах чадвартай байх ёстой.
- **ФБШ 300:** Ухаалаг гэрээ нь модульчлагдсан байх ёстой бөгөөд шинэчлэгдэхэд хялбар байх ёстой.
- **ФБШ 400:** Систем нь хүлээн зөвшөөрөгдсөн тодорхой хугацааны дотор баталгаажуулах хүсэлтийг хурдан боловсруулах чадвартай байх ёстой.
- **ФБШ 500:** Систем нь янз бүрийн техникийн чадвартай хэрэглэгчдэд үүнийг үр дүнтэй ашиглах боломжийг олгодог хэрэглэгчдэд ээлтэй интерфэйстэй байх ёстой.
- **ФБШ 500:** Систем нь янз бүрийн үйлдлийн систем, хөтөч, төхөөрөмжтэй нийцтэй байх ёстой.
- **ФБШ 600:** Систем нь лиценз олгох, дижитал гүйлгээ, блокчэйн технологитой холбоотой аливаа зохицуулалтын шаардлагад нийцэж байх ёстой.
- **ФБШ 700:** Энэ систем нь гамшгийн үед өгөгдөл алдагдахгүй байхын тулд найдвартай нөөцлөх, сэргээх механизмтай байх ёстой.

2.3 Use case диаграм

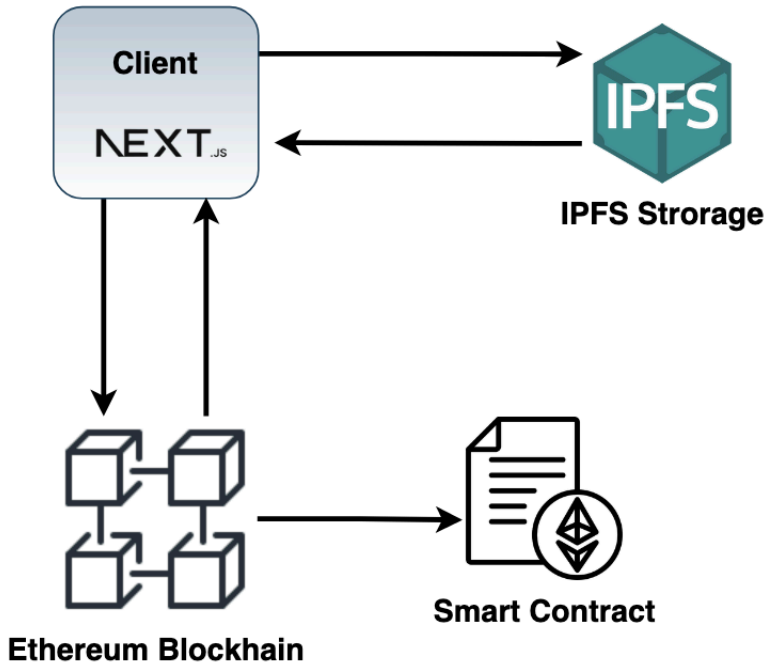


Зураг 2.1: Use-case диаграм

2.4 Архитектур

Энэхүү төслийг ажиллахад илүү хямд зардалтай хүртээмжтэй, ачаалал даах чадварыг нэмэх зорилгоор серверлесс Архитектур сонгосон юм. 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 хадгална.



Зураг 2.2: Архитектур

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 тэмплэйт-ээр бус JavaScript-ээр бичигддэг учраас өгөгдлийг апп хооронд хялбар дамжуулж, DOM-оос state-ээ тусд 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 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 блокчейнүүдийг дэмждэг.

3.1.3 Wagmi

Wagmi нь блокчейнтэй 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.

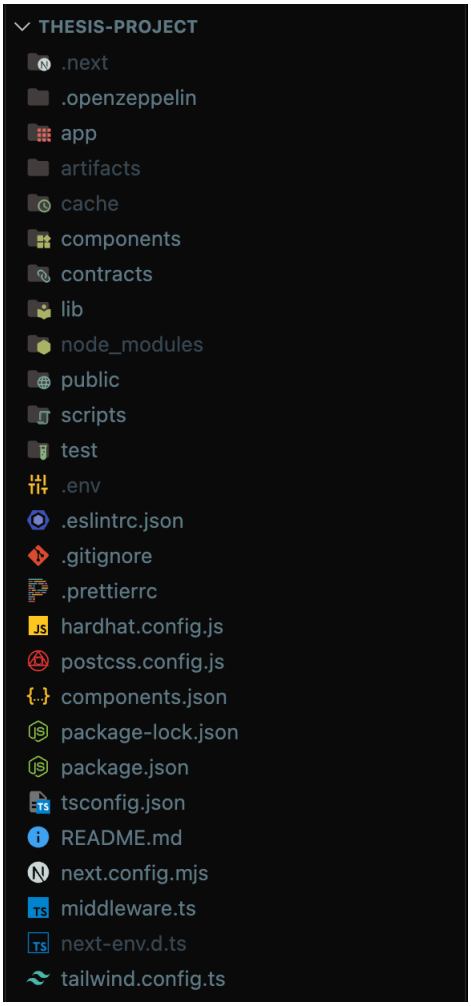
3.1.4 Pinata

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.

3.2 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 соногосон 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. Эдгээр бүтэц нь файлын ID, эзэмшигчийн хаяг, файлын нэр, тайлбар, ангилал, файлын хэш, үүсгэсэн хугацааны зэрэг атрибутуудыг агуулна. Мөн дараах функцуудтэй:

- **createFile**
- **issueLicense**
- **getAllPublicFiles**
- **getAllUserFiles**
- **getAllUserLicenses**
- **validateLicense**
- **getPublicFileById**
- **getMarketplaceFiles**
- **generateUniqueLicense**

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

contract LicenseMarketplace {
    address public owner;

    struct File {
        uint256 id;
        address owner;
        string fileName;
        string description;
        string category;
        string fileHash;
        bool isPublic;
        uint256 createdAt;
    }

    struct License {
        uint256 licenseNumber;
        uint256 fileId;
        address owner;
        string fileName;
        string description;
        string category;
        string fileHash;
        bool isPublic;
        uint256 createdAt;
    }

    mapping(address => File[]) private userFiles;
    mapping (address => License[]) private fileLicenses;
    mapping(uint256 => bool) public usedLicenses;

    File[] public publicFiles;
    uint256 public fileId;

    constructor() {
        owner = msg.sender;
    }

    function createFile(string memory _fileName, string memory
       _description, string memory _category,
        string memory _fileHash, bool  _isPublic) external{
        fileId++;
        uint256 newId = fileId;

        File memory newFile = File({
            id: newId,
            owner: msg.sender,
            fileName: _fileName,
            description: _description,
            category: _category,
            fileHash: _fileHash,
```

```
53              isPublic: _isPublic,
54              createdAt: block.timestamp
55          });
56
57          userFiles[msg.sender].push(newFile);
58
59          if(_isPublic) {
60              publicFiles.push(newFile);
61          }
62      }
63
64      function issueLicense(address _owner, uint256 _id, string
            memory _fileName, string memory _description, string
            memory _category,
65          string memory _fileHash, bool  _isPublic) external {
66
67          uint256 licNum = generateUniqueLicense();
68
69          License memory newFile = License({
70              licenseNumber: licNum,
71              fileId: _id,
72              owner: _owner,
73              fileName: _fileName,
74              description: _description,
75              category: _category,
76              fileHash: _fileHash,
77              isPublic: _isPublic,
78              createdAt: block.timestamp
79          });
80
81          fileLicenses[msg.sender].push(newFile);
82      }
83
84      function getAllPublicFiles() external view returns(File[]
            memory) {
85          return publicFiles;
86      }
87
88      function getAllUserFiles() external view returns(File[]
            memory) {
89          return userFiles[msg.sender];
90      }
91
92      function getAllUserLicenses() external view returns(License[]
            memory) {
93          return fileLicenses[msg.sender];
94      }
95
96
97      function validateLicense(uint256 licenseNumber) external view
            returns (bool) {
98          return usedLicenses[licenseNumber];
99      }
100
```

```
101     function getPublicFileById(uint256 _id) external view returns
            (File memory) {
102         for (uint256 i = 0; i < publicFiles.length; i++) {
103             if (publicFiles[i].id == _id) {
104                 return publicFiles[i];
105             }
106         }
107         revert("Public␣file␣not␣found");
108     }
109
110
111     function getMarketplaceFiles() external view returns (File[]
            memory) {
112         uint256 senderFilesCount = userFiles[msg.sender].length;
113         uint256 totalPublicFilesCount = publicFiles.length;
114
115         uint256 excludedFilesCount = senderFilesCount;
116         for (uint256 i = 0; i < fileLicenses[msg.sender].length;
                i++) {
117             if (fileLicenses[msg.sender][i].isPublic) {
118                 excludedFilesCount++;
119             }
120         }
121
122         File[] memory result = new File[](totalPublicFilesCount -
                excludedFilesCount);
123         uint256 index = 0;
124
125         for (uint256 i = 0; i < totalPublicFilesCount; i++) {
126             bool isUserFile = false;
127             bool hasUserLicense = false;
128
129             for (uint256 j = 0; j < senderFilesCount; j++) {
130                 if (publicFiles[i].id == userFiles[msg.sender][j
                        ].id) {
131                     isUserFile = true;
132                     break;
133                 }
134             }
135
136             for (uint256 k = 0; k < fileLicenses[msg.sender].
                    length; k++) {
137                 if (publicFiles[i].id == fileLicenses[msg.sender
                        ][k].fileId) {
138                     hasUserLicense = true;
139                     break;
140                 }
141             }
142
143             if (!isUserFile && !hasUserLicense) {
144                 result[index] = publicFiles[i];
145                 index++;
146             }
147         }
```

```
148
149         return result;
150     }
151
152      function generateUniqueLicense() internal returns (uint256)
            {
153         uint256 randomNumber = uint256(keccak256(abi.encodePacked
               (block.timestamp, block.difficulty, msg.sender)));
154         uint256 license = randomNumber % 10000000000;
155
156         while (usedLicenses[license]) {
157             randomNumber = uint256(keccak256(abi.encodePacked(
                   randomNumber, block.timestamp)));
158             license = randomNumber % 10000000000;
159         }
160
161         usedLicenses[license] = true;
162
163         return license;
164     }
165 }
```

Код 3.1: Ухаалаг гэрээ

3.2.3 *Ухаалаг гэрээг блокчейнд байршуулах*

```
1  const { ethers } = require('hardhat');
2
3  async function deployContract() {
4    let contract;
5
6    try {
7      contract = await ethers.deployContract('LicenseMarketplace');
8      await contract.waitForDeployment();
9
10     console.log('Contracts deployed successfully.');
11     return contract;
12   } catch (error) {
13     console.error('Error deploying contracts:', error);
14     throw error;
15   }
16 }
17
18 async function main() {
19   let contract;
20
21   try {
22     contract = await deployContract();
23     await saveContractAddress(contract);
24
25     console.log('Contract deployment completed successfully.');
26   } catch (error) {
```

```
27      console.error('Unhandled error:', error);
28    }
29  }
30
31  main().catch((error) => {
32    console.error('Unhandled error:', error);
33    process.exitCode = 1;
34  });
```

Код 3.2: Ухаалаг гэрээ

3.2.4 Хэрэглэгч талын хөгжүүлэлт (Front-end)

3.2.5 Үр дүн

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

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