

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

Нямдоржийн Энхболд

**Алгоритм Болон Програмчлалын Аргууд Сурах
Интерактив Систем
(Interactive Algorithm and Programming Learning System)**

Програм хангамж (D061302)
Бакалаврын судалгааны ажил

Улаанбаатар

2023 он

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

Алгоритм Болон Програмчлалын Аргууд Сурах
Интерактив Систем
(Interactive Algorithm and Programming Learning System)

Програм хангамж (D061302)
Бакалаврын судалгааны ажил

Удирдагч: _____ Дэд проф. Б.Сувдаа

Гүйцэтгэсэн: _____ Н.Энхболд (20B1NUM0102)

Улаанбаатар

2023 он

Зохиогчийн баталгаа

Миний бие Нямдоржийн Энхболд ”Алгоритм Болон Програмчлалын Аргууд Сурах Интерактив Систем” сэдэвтэй судалгааны ажлыг гүйцэтгэсэн болохыг зарлаж дараах зүйлсийг баталж байна:

- Ажил нь бүхэлдээ эсвэл ихэнхдээ Монгол Улсын Их Сургуулийн зэрэг горилохоор дэвшүүлсэн болно.
- Энэ ажлын аль нэг хэсгийг эсвэл бүхлээр нь ямар нэг их, дээд сургуулийн зэрэг горилохоор оруулж байгаагүй.
- Бусдын хийсэн ажлаас хуулбарлаагүй, ашигласан бол ишлэл, зүүлт хийсэн.
- Ажлыг би өөрөө (хамтарч) хийсэн ба миний хийсэн ажил, үзүүлсэн дэмжлэгийг дипломын ажилд тодорхой тусгасан.
- Ажилд тусалсан бүх эх сурвалжид талархаж байна.

Гарын үсэг: _____

Огноо: _____

ГАРЧИГ

УДИРТГАЛ	1
БҮЛГҮҮД	2
1. ОРШИЛ	2
1.1 Үндэслэл	2
1.2 Зорилго	2
2. СИСТЕМИЙН СУДАЛГАА	3
2.1 Ижил төстэй системүүд	3
2.2 Системийн онцлог	6
3. СИСТЕМИЙН ШИНЖИЛГЭЭ	8
3.1 Шаардлагууд	8
3.2 Технологийн асуудлууд	10
4. СИСТЕМИЙН ЗОХИОМЖ	12
4.1 Ажлын явцын диаграм	13
4.2 Хэрэглэгчийн интерфэйс	18
4.3 Технологиуд болон шийдлүүд	19
4.4 Өгөгдлийн сангийн зохиомж	23
5. ХЭРЭГЖҮҮЛЭЛТ	25
5.1 Frontend	25
5.2 Backend	30
ДҮГНЭЛТ	34
НОМ ЗҮЙ	34
ХАВСРАЛТ	35
А. КОДЫН ХЭРЭГЖҮҮЛЭЛТ	36

ЗУРГИЙН ЖАГСААЛТ

2.1	Accepted.mn бодлого бодох	4
2.2	Accepted.mn үр дүн	4
2.3	spoj/rgb7 бодлого бодох	5
4.1	Системийн ерөнхий зураглал	12
4.2	Ажлын явцын диаграм	13
4.3	Хэрэглэгчийн кодыг дүгнэж буй сценарийн дарааллын диаграм	17
4.4	Wireframe	18
4.5	Mockup	18
4.6	Codemirror сан	20
4.7	Өгөгдлийн нэгж хоорондын хамаарлыг дүрсэлж буй диаграм	24
5.1	Бодлого бодох хэсэг	25
5.2	NEXTjs-ийн сервер талдаа cache ашиглаж буй байдал	27
5.3	Dracula theme дээр python хэлний boilerplate бүхий код засварлагч	28
5.4	Syntax-ийн алдаа бүхий код	29
5.5	Syntax-ийн алдааг харуулж буй системийн интерфейс	29
5.6	Хэрэглэгчийн профайл хуудас	29
5.7	Сервисүүдийг байршуулсан байдал	30
5.8	Хандах боломжтой портыг тодорхойлох	31
5.9	Нөөцийг хязгаарлах	31
5.10	Зөвхөн унших боломжтой болгох	31
5.11	Firestore-ийн ProblemSolutions collection болон түүний заагч document-үүд	33
5.12	Firestore-ийн subcollection group-ийн ашиглалт, агуулж буй document-үүд	33

Кодын жагсаалт

A.1	Next.js Auth ашигласан байдал	36
A.2	Codemirror буюу @uiw/react-codemirror-ийн компонентийг ашиглаж буй байдал	37
A.3	NEXTjs бодлогууд авах серверийн endpoint	40
A.4	Бодлогуудыг cache-ээс fetch хийх component	40
A.5	CollectionGroup query бичиж хэрэглэгчийн бодолтуудыг авч буй route	41
A.6	Бодолт явуулах NEXTjs endpoint	42
A.7	Хэрэглэгчийн нэрнээс Avatar үүсгэж буй байдал	42
A.8	Python хэлний middleware	43
A.9	Stateless байдлаар firebase admin SDK ашиглалт	44
A.10	ExpressJS серверийн оролт	45

УДИРТГАЛ

Програмчлалын тэр дундаа алгоритмын түгээмэл бодлого, асуудлуудыг шийдэх нь компьютерийн ухааны чиглэлээр сурагчид болон програм хангамж хөгжүүлэгчдийн үндсэн суурь чадвар болж өгдөг. Цаашлаад тухайн алгоритмуудыг төрөл бүрийн програмчлалын хэл болон архитектур, зохиомж зэрэг бодит жишээн дээр хэрэгжүүлэх шаардлага түгээмэл гардаг.

Энэхүү судалгааны ажлаар хийсэн бүтээл нь "Coldbrains" гэх веб технологиудад суурилсан, програмчлалд суралцах сонирхолтой хэрэглэгчид рүү чиглэсэн програмчлалын бодлого бодох систем юм. Энэхүү систем нь хэрэглэгчийн програмчлал болон алгоритмын мэдлэгийг бататгаж, шийдлээ бусадтайгаа хуваалцах боломжийг бий болгоно. Цаашлаад бодлого бүрт харгалзах бодолтын санг бий болгох билээ.

1. ОРШИЛ

1.1 Үндэслэл

Бакалаврын судалгааны ажлаар инженерчлэлийн чиглэлээр хийж буй энэхүү бүтээл нь програмчлал болон архитектур, зохиомжлолын хүрээнд олон мэдлэг чадвар, судалгааг шаардаж байгаа бөгөөд миний бие Нямдоржийн Энхболд өөрийн сурсан мэдлэгээ өнөөгийн технологиудын судлагаа, практиктай хослуулан энэхүү вебэд суурилсан системийг хийвэл зохино гэж үзэв.

Монголд ашиглагдаж буй ихэнх програмчлалын бодлого боддог платформ/системүүд нь ихэвчлэн гадных байдаг бөгөөд өөрсдийн үүсгэсэн бодлогын санг оруулж хэрэглэж байгаа билээ. Програмчлалд суралцаж буй сурагч, оюутан залууст эх хэл дээр нь хэлний бэрхшээлээс ангид байдлаар суралцах боломжийг олгох нь сурах процесст эергээр нөлөөлдөг бөгөөд одоогоор Монголд хэрэгжүүлсэн системүүд маш цөөхөн, мөн өөр зорилго агуулсан байдаг.

Алгоритм болон програмчлалын салбарт өөрийн хувь нэмрээ оруулна гэх мотивацын дор энэхүү системийг хэрэгжүүлэх сэдэл төрсөн билээ.

1.2 Зорилго

Уг бүтээл нь цаашаа төсөл байдлаар урт хугацаанд хэрэгжих бөгөөд инженерчлэлийн судалгааны ажлаар MVP(Minimum viable product)-ийг бий болгохыг зорьсон.

”Coldbrains” гэх систем нь хэрэглэгчийн сурсан програмчлалын мэдлэг чадваруудыг бататгах болон үнэлэх зорилготой бөгөөд хэрэглэгч өөрийн түвшинг тодорхойлох, цаашлаад тулгарсан програмчлалын асуудлуудад ямар аргуудыг ашиглах боломжтой байдаг талаарх ойлголтуудыг хуваалцах билээ.

2. СИСТЕМИЙН СУДАЛГАА

2.1 Ижил төстэй системүүд

Уг чиглэлээр гадаадын маш олон системүүд байдаг бөгөөд өргөн сонголт бүхий хэлнүүд, түүнд зориулсан код засварлагч агуулсан байдаг. Одоогоор түгээмэл буй цогц системүүдээс дурьдвал

- Leetcode
- Hackerrank
- Codeforce
- SPOJ
- ...

гэх мэт. Эдгээр системүүдийн хэрэглэгчийн код ажиллуулж буй байдал нь нэгэн төрлийн бөгөөд UI/UX-ээрээ голчлон ялгарч байгаа билээ. Хэрэглэгчийн кодыг код засварлагч эсвэл файл уншигчаар хүлээн авч өөрсдийн сервер лүүгээ дамжуулдаг бөгөөд тухайн сервер дээр sandbox¹ -д хэрэглэгчийн кодыг stdio² -ийг ихэвчлэн ашиглан шинжилгээ хийж үр дүнг нь буцаан харуулж байна.

Хэрэглэгчийн оролтоос бусдаар аюулгүй байдал талаас мөн request/submission throttling³, debouncing⁴ зэрэг стратегиудыг ашигласан байна.

¹ хамгаалагдсан хязгаарлагдмал орчин

² Standard input output буюу оролт гаралтын стандарт урсгал

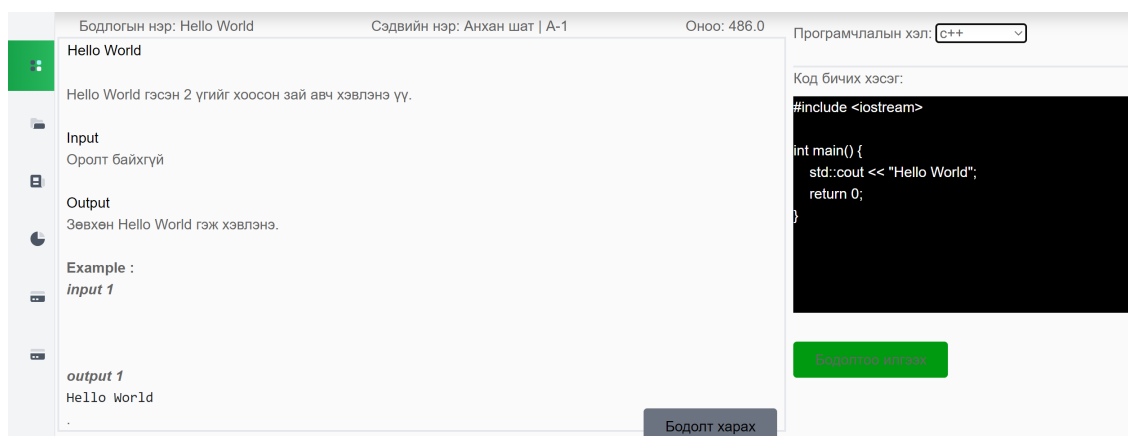
³ Хүсэлт болон дамжуулалтын давтамжийг бууруулах

⁴ Илүүдэл давхар ажиллагааг хязгаарлах

Монголд хэрэгжүүлсэн болон түгээмэл ашигладаг системүүдээс дурьдвал

- SPOJ
- Accepted.mn
- ...

зэрэг бодлогын сан, дэд систем байдлаар хэрэгжсэн байна.



Зураг 2.1: Accepted.mn бодлого бодох

accepted academy							
Хайх							
Эхлэл Мэдээлэл Математик Алгоритм Физик Хими Хандив jackerenh							
Хуудас 1 - 2350.							
Дараах Сүүлийнх »							
ХЭРЭГЛЭГЧИЙН НЭР	ОГНОО	БОДЛОГО	ҮР ДҮН	ХУГАЦАА	САНАХ ОЙ	LAN	
tuguldurtsovoo	2023-11-04 15:50:19	Нийлбэрийг ол	ACCEPTED			c++	
tuguldurtsovoo	2023-11-04 15:50:19	n ... 1	ACCEPTED			c++	
tselmuudeim	2023-11-04 15:50:19	Автобусны карт	ACCEPTED			c++	
tuguldurtsovoo	2023-11-04 15:50:19	Байрлалын тоо	WRONG_ANSWER			c++	
jackerenh	2023-11-04 15:50:19	Hello World	ACCEPTED			c++	

Зураг 2.2: Accepted.mn үр дүн

Энэхүү accepted.mn[4] нь C/C++, Java, Python, Dart зэрэг програмчлалын хэлнүүдийг дэмждэг сургалтын платформ бөгөөд код засварлагчаар хэрэглэгчийн кодыг хүлээн авч шүүж зөв/буруу

статусыг харуулдаг. Тухайн кодыг ажиллуулахдаа websocket технологийг ашиглан сервертэйгээ харьцдаг. One time буюу 1 удаа бичин үр дүнгээ хардаг байдлаар хэрэгжүүлсэн байгаа билээ.

submit a solution

Эх кодоо энд оруулна уу, эсвэл файлаа сонго: No file chosen

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     // your code here
6     return 0;
7 }
8
9 }
```

C++ (g++ 4.3.2)

Зураг 2.3: spoj/rgb7 бодлого бодох

SPOJ.com[5] нь нийт 41 програмчлалын хэл дэмждэг бөгөөд accepted.mn-ээс ялгаатай нь REST API ашиглан кодыг шүүгч сервер лүү дамжуулдаг. Мөн адил one time байдлаар бичин үр дүнгээ хардаг.

Эдгээр 2 систем нь тус бүр ижил шинж чанартай тогтмол ашигладаг олон хэрэглэгчтэй бөгөөд хэрэглэгч талдаа тодорхой алдааны мэдээлэл болон хэрэглэгчийн интерфейсийн гүйцэтгэлээрээ гадаадын системүүдээс дутмаг байгааг харж болно. Монгол хэлээр бодлогыг орчуулж оролт гаралтын жишээг тайлбарласан нь зорилтот хэрэглэгчиддээ илүү сайн хүрснийг харж болно. Эдгээр системүүдийг судалсны үндэс дээр хамгийн сайн хэрэгжүүлэлт(Best practice)-үүдийг төлөвлөсөн билээ.

2.2 Системийн онцлог

Програмчлалийн мэдлэгийг нь англи хэл дээр интернет болон нээлттэй сангуудаас хүссэн хүн нь авч болох бөгөөд олон улсын стандарт хэлийг эзэмшсэн байх нь уг салбарын мэргэжилтэнд зайлшгүй байх ёстой чадвар юм. Харин програмчлалд суралцах процесст гадаад хэл болон эх хэл нь өөр нөлөө үзүүлдэг бөгөөд учир шалтгааныг нь бие даан олох, логик сэтгэлгээгээр бодоход эх хэл нь илүү давуу талтайг олон судалгаа баталсан байдаг. Шинжилгээ хийх, логиктой сэтгэх, учир шалтгааныг нь тайлах асуудлууд дээр хийсэн судалгаагаар эх хэл дээрээ суралцсан бүлэг хүмүүс нь илүү сайн үр дүнд хүрдэг болохыг судалж баталсан байдаг[1].

2.2.1 Хэрэглэгчийн интерфейс

Систем нь Монгол хэл бүхий хэрэглэгчийн интерфейстэй байх бөгөөд одоо байгаа ижил төстэй системүүдээс зохиомж болон хэрэглэгчийн интерфейсээрээ ялгарах билээ. Үүнд:

- Intellisense⁵ бүхий код засварлагч, хэрэглэгчид түүний зарласан функц болон хувьсагчдийг санал болгон код бичих процессийг илүү хялбарчлана.
- Хэрэглэгчийн сонголт бүхий код засварлагчийн өнгө, үзэмж, хэрэглэгч өөрийн дассан өнгөний зохицол дээр бичих боломжтой.
- Өөрийн бодсон бодлогын бодолт болон бусдын бодолтуудыг харах боломжтой.
- Илүү уян хатан програмчлалын хэлний боломжуудыг ашиглах. Үүнд тухайн програмчлалын хэлний built-in өгөгдлийн бүтэц болон бусад сангуудыг ашиглах боломжууд орно.
- Хэрэглэгч өөрийн бичиж буй кодны алдаа тэр дундаа syntax⁶-ийн алдааг мөрийн дугаараар илрүүлэх боломжтой.

⁵Ухаалаг байдлаар хэрэглэгчид код санал болгогч

⁶Код бичиж буй хэрэглэгчээс үүдэлтэй кодны бүтэц, бичиглэлийн алдаа

- Тухайн бодлого нь олон тест кейсээр шалгагдах бөгөөд хэрэглэгч аль тест кейс дээр тэнцээгүйг харуулна.

2.2.2 Хэрэглэгчийн боломжууд

Системийн гол зорилго нь эх хэл дээр зөвхөн алгоритм болон кодын логик тал дээр хэрэглэгчийг төвлөрөхөд дэмжих бөгөөд алдааны мэдээлэл болон бусад стандарт програмчлалын ойлголтуудыг англи хэл дээр байлгаж, хэрэглэгчийг тэдгээр нэршлүүдийг судалж, илүү дотно болохыг дэмжинэ.

Хэрэглэгч мөн бодлогоо амжилттай бодсон төлөвт байвал бусдын хэрэгжүүлэлтүүдийг судалж шинжлэх боломжтой болох билээ.

Хэрэглэгчийн ашиглалт талаас дотоодын системүүдээс ялгарч буй зүйлсээс дурьдвал тухайн хэрэглэгч заавал бүртгэл үүсгэлгүйгээр өөрийн бусад платформууд дээр ашигладаг мэдээллээр(Github, Google credentials provider⁷) нэвтэрч орон тэгш эрхтэйгээр ашиглах боломжтой. Мөн хэрэглэгч 1 time буюу тухайн бодолтоо submission хийх процесс руу дамжуулаад үр дүнгээ хүлээхийн оронд тухайн бодолтоо илүү бодит үр дүнтэйгээр тухайн хуудсан дээр засвар оруулан дахин оролдлого хийх боломжтой. Одоогийн дотоодод ашиглаж буй системүүд нь алдааны тодорхой мэдээлэл дутмаг, мөн хэрэглэгчийн үр дүнг гаргахын тулд тухайн хэрэглэгчийн дараа дараачийн submission процессуудыг хуудас шилжин, төлөвт оруулан шинжилдэг. Энэхүү процессийг "Coldbrains" системээр хэрэглэгч талдаа илүү ойлгомжтой,тодорхой болгож шийдсэн бөгөөд хэрэглэгч удаа дараа тухайн кодыг хүлээж автал олон оролдлого 1 дор хийх боломжуудыг бий болгож байгаа билээ.

⁷Хэрэглэгчийн мэдээллийг зөвшөөрлийн дагуу олгох платформууд

3. СИСТЕМИЙН ШИНЖИЛГЭЭ

3.1 Шаардлагууд

3.1.1 Функциональ шаардлага

- *ФШ01* - Хэрэглэгчид загвар код өгөгдөнө.
- *ФШ02* - Хэрэглэгч өөрт тохирох дизайныг сонгож, customize хийх боломжтой байна.
- *ФШ03* - Хэрэглэгчид зориулсан editor нь хэрэглэгчийг бичиж байх үйл явцыг сонсон хурдан хариу өгөх intellisense-тэй байна.
- *ФШ04* - Програмчлалын Python, Javascript болон C/C++(*optional*), JAVA(*optional*) хэлнүүд дээр бодлогыг бодох боломжтой байх.
- *ФШ11* - Хэрэглэгч амжилттай бодлогоо бодсоноор бусдын кодын хэрэгжүүлэлт, шийдлүүдийг харах боломжтой байна.
- *ФШ12* - Хэрэглэгч кодоо ажиллуулах үед тэнцсэн эсвэл тэнцээгүй эсэхийг нь хэрэглэгчид мэдэгддэг байх ёстой.
- *ФШ21* - Систем нь монгол хэл бүхий хэрэглэгчийн интерфейстэй байх ёстой.

3.1.2 Функциональ бус шаардлага

- *ФБШ01* - Хэрэглэгчийн алгоритм ба програмчлалын мэдлэгийг бататгаж, цааш өөр бодлогуудаар сорьдог байна.
- *ФБШ02* - Бодох процессийг илүү хялбарчилсан байна.
- *ФБШ03* - Програмчлалын үндсэн ойлголттой ямар ч хэрэглэгчид аппликейшн нь хэрэглэхэд ойлгомжтой, хүртээмжтэй интерфейстэй байна.

- *ФБШ11* - Сервер лүү brute-force байдлаар халдалт хийхээс сэргийлсэн байх.
- *ФБШ12* - Өгөгдлийн сан руу хэрэглэгчид зөвхөн сервисүүдээр дамжин хандалт хийж болдог байх.
- *ФБШ13* - Систем нь 24/7 ажиллагаатай байх бөгөөд хэзээ ч, хаанаас ч хандах боломжтой байна.
- *ФБШ14* - Бодлогын сан нь дахин сэргээх боломжтойгоор гараар backup хийдэг байна.
- *ФБШ15* - Ачаалал даах чадвартай байх бөгөөд ямар нэгэн ачааллаас үүдэлтэй асуудал гарвал унтралгүйгээр тодорхой хугацааны дараа хэвийн ажиллагааг хангадаг байх.
- *ФБШ16* - Хэрэглэгчээс оруулсан сэжиг бүхий кодууд compiler сервер дээр ажилладаггүй байна.
- *ФБШ17* - Хэрэглэгчээс орж ирэх кодыг гуравдагч систем рүү хандах боломжгүй орчинд ажиллуулж үр дүнг гаргадаг байх хэрэгтэй.
- *ФБШ21* - Бодлого бүр тест кэйстэй байх шаардлагатай.
- *ФБШ22* - Систем нь хэрэглэгчийн мэдээлэл болон бодсон бодлогуудын мэдээллийг хадгалдаг байх ёстой.

3.2 Технологийн асуудлууд

”Coldbrains” систем нь програмчлалын бүтээгдэхүүн дотор програмчлалын үйл явцыг явуулж байгаа тул юун түрүүнд аюулгүй үйл ажиллагаа болон найдвартай байдлыг хангадаг байх хэрэгтэй билээ. Нээлттэй вебсайтаар дамжуулан гадны ямар ч хэрэглэгч хэрэглэх боломжтой бөгөөд зохисгүй хэрэглэгчдийн үйл ажиллагааг хязгаарлах маш олон програмчлалын асуудлууд бий болж байна.

3.2.1 *Middleware*

Кодыг хүлээн авч ажиллуулж буй серверийг хэрэглэгчээс хязгаарлахын тулд тухайн хэрэглэгчийн веб аппликейшн болон серверийн хооронд middleware сервер байх бөгөөд энэхүү сервер нь хэрэглэгчээс ирэх хүсэлтүүдийг боловсруулан цааш дамжуулдаг байх шаардлагатай билээ. Үүнд:

- Хэрэглэгчээс богино хугацаанд ирэх олон хүсэлтүүдийг хязгаарлах
- Жинхэнэ серверийн IP хаягийг нууцлан улмаар зөвхөн өөр дээрээсээ дамжуулах
- Хэрэглэгчийн эрхийг(user credentials/access token) шалгах
- Frontend-ээс ирэх Payload-ийг encrypt хийх

зэрэг үүргүүдийг гүйцэтгэх билээ.

3.2.2 *Sandboxing*

Middleware-ээр дамжуулагдан ирсэн кодыг аюулгүй эсвэл хортой код гэдгийг ялгах боломжгүй учраас тухайн кодны ажиллах орчныг зааж өгөн хязгаарлах шаардлагууд бий болсон. Үүнд:

- Тухайн кодны үйл ажиллагаа нь ямарваа нэгэн байдлаар гадны гуравдагч endpoint руу хандах боломжгүй байх. Өөрөөр сүлжээнд нэвтрэх боломжгүй болгох. Тухайн програмчлалын хэлний ашиглагддаг built-in сүлжээнд холбогддог аргуудыг блоклох.

- Тухайн код нь үйлдлийн систем болон файл системд хандах боломжгүй байх. Хэрэв хандсан тохиолдолд тухайн үйлдлүүдийг хийсвэр орчинд гүйцэтгэх.
- Хэрэв код нь тухайн серверийн нөөцийг бүхэлд нь ашиглах тохиолдолд тухайн серверт нөөцийн хязгаарлалт бий болгох. Тухайн хязгаарыг давсан үед алдааг буцаадаг болгох.
- Хэрэглэгч бүрийн код бие даасан байдлаар ажилладаг байхын тулд серверүүдийг stateless¹ байдлаар зохиомжлох.

гэх мэт аюулгүй байдал болон зохиомжийн шийдлүүдийг боловсруулах шаардлагатай.

3.2.3 Cloud үйлчилгээнүүд болон интеграц

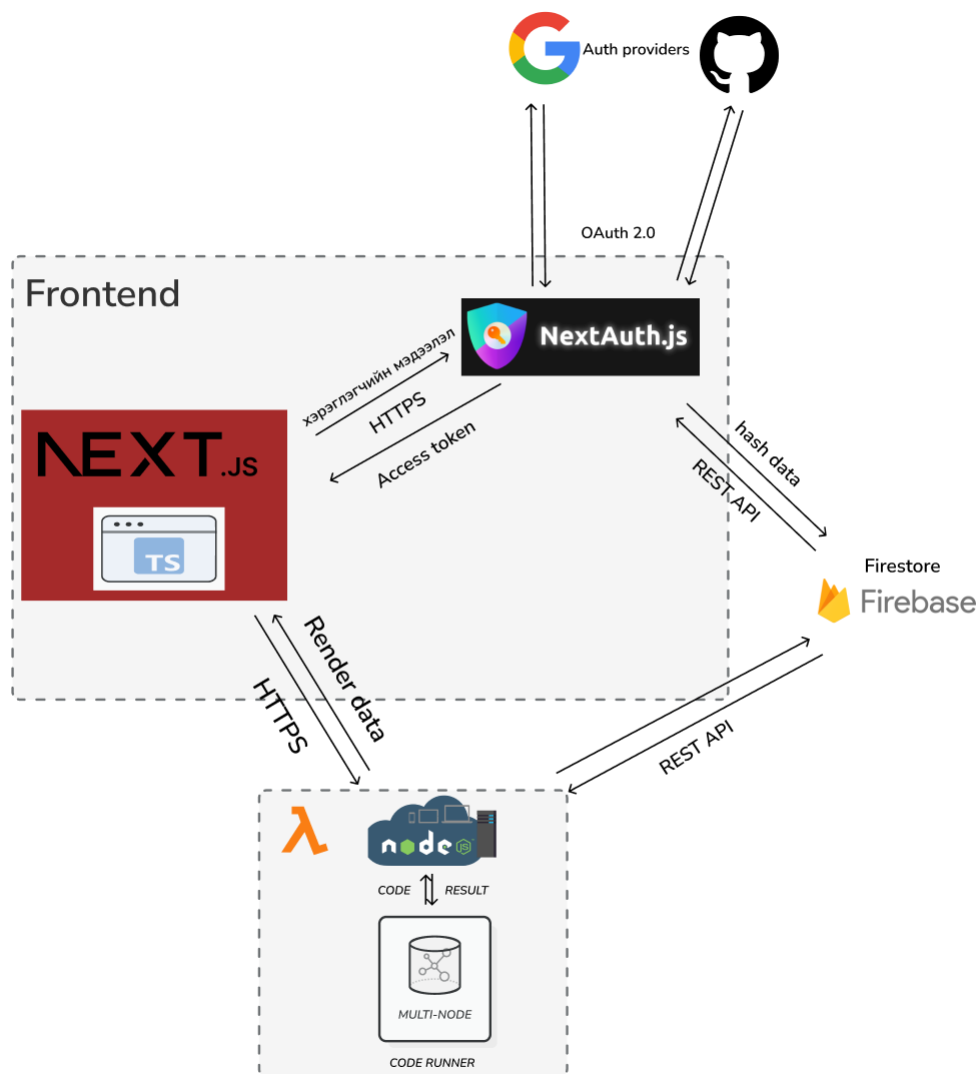
Дээр дурьдсан 2-т програм хангамж талаас асуудлуудыг тодорхойлсон бол эдгээр програм хангамжуудыг ажиллуулах техник хангамж, түүнийг ханган нийлүүлэгчдийг судалж үзэх нь дараачийн зайлшгүй асуудал мөн билээ.

Програм хангамжуудыг stateless байдлаар хэрэгжүүлснээр тэдгээрт ашиглагдах бусад сервисүүдийн инстанцуудын ажиллагаа хязгаарлагдмал болдог билээ. Socket холбоо болон цаашлаад **singleton** загвараар хэрэгждэг холболтууд нь stateless байдлаар хэрэгжүүлэхэд зохимжгүй болдог.

Үүнд ламбда илэрхийлэл буюу функциональ програмчлалын ойлголтуудыг зохиомж дээр хэрэгжүүлэх нь зүйтэй болно[6]. Нарийвчилбал, өгөгдлийн сан, гуравдагч үйлчилгээ үзүүлэгчдийг 1 удаагийн холболтоор шийдэх арга замуудыг тодорхойлох асуудлууд гарч ирч буй юм.

¹Төлөвгүй буюу ямарваа нэгэн байдлаар санах ойг шууд сав байдлаар ашигладаггүй байх

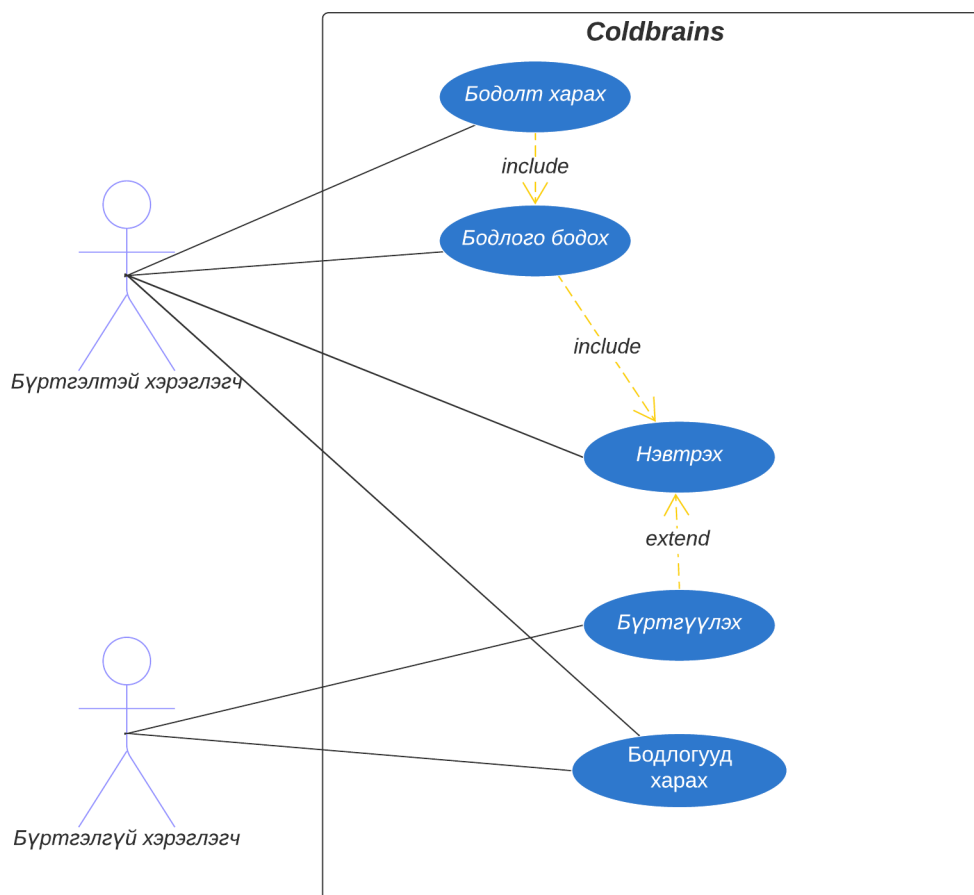
4. СИСТЕМИЙН ЗОХИОМЖ



Зураг 4.1: Системийн ерөнхий зураглал

”Coldbrains” системийн ашиглаж буй технологиуд болон тэдгээрийн холбоог харуулж байгаа бөгөөд аль болох ерөнхий байдлаар дүрслэхийг оролдов.

4.1 Ажлын явцын диаграм



Зураг 4.2: Ажлын явцын диаграм

Хэрэглэгчийг бүртгэлтэй болон бүртгэлгүй гэж ангилвал бүртгэлгүй хэрэглэгч нь системтэй танилцах зорилгоор бодлогуудын жагсаалтыг харах боломжтой бол бүртгэлтэй хэрэглэгч нь тухайн бодлогуудыг бодох, амжилттай бодсон бол бодолтуудыг харах, мөн өөрийн профайл хэсгийг харах боломжтой болно.

Ажлын явц: 1. Бүртгүүлэх

Ажлын явц	Бүртгүүлэх
Зорилго	Тоглогч системд бүртгэл үүсгэж нэвтрэхэд ашиглах
Угтвар нөхцөл	Тоглогч систем рүү хандсан байх
Тоглогч	Coldbrains системд бүртгэлгүй хэрэглэгч
Тайлбарлалт	1. Систем нэр, нууц үг, и-мейл хаягийг шаардана. 2. Шаардлагатай мэдээллийг оруулна. 3. Систем амжилттай бүртгэн нэвтрэх хуудас луу шилжүүлнэ.
Өргөтгөл	Тоглогч буцах товч даран ажлын явцыг дуусгавар болгоно.
Хувилбар	2а. Бүртгэлтэй хэрэглэгч байвал мэдэгдэж, мэдээллийг нь арилгана. Ажлын явцыг эхнээс нь эхлүүлнэ.

Ажлын явц: 2. Бодлогууд харах

Ажлын явц	Бодлогууд харах
Зорилго	Тоглогч системд буй бодлогын санг харах
Угтвар нөхцөл	Тоглогч систем рүү хандсан байх
Тоглогч	Coldbrains систем рүү хандсан хэрэглэгч
Тайлбарлалт	1. Систем бодлогын санг хэрэглэгчид харуулна.

Ажлын явц: 3. Нэвтрэх

Ажлын явц	Нэвтрэх
Зорилго	Тоглогч системийг ашиглахын тулд нэвтрэх
Угтвар нөхцөл	Тоглогч бодлогын тодорхойлолтыг харах, бодлого бодохыг оролдох,
Тоглогч	Coldbrains системд бүртгэлтэй аль эсвэл шууд нэвтрэх сонирхолтой гадны платформд бүртгэлтэй хэрэглэгч
Тайлбарлалт	1. Систем бүртгэлтэй нэр, нууц үг шаардана. 2. Шаардлагатай мэдээллийг оруулна. 3. Систем тоглогчид session үүсгэн бодлогууд хуудас руу шилжүүлнэ.
Өргөтгөл	Тоглогч буцах товч даран ажлын явцыг дуусгавар болгоно.
Хувилбар	1а. Хэрэглэгч бусад платформуудын эрхийг оруулан session үүсгэж ажлын явцыг дуусгана. 2а. Бүртгэлтэй хэрэглэгч олдоогүй бол мэдэгдэж, мэдээллийг нь арилгана. Ажлын явцыг эхнээс нь эхлүүлнэ.

Ажлын явц: 5. Бодолт харах

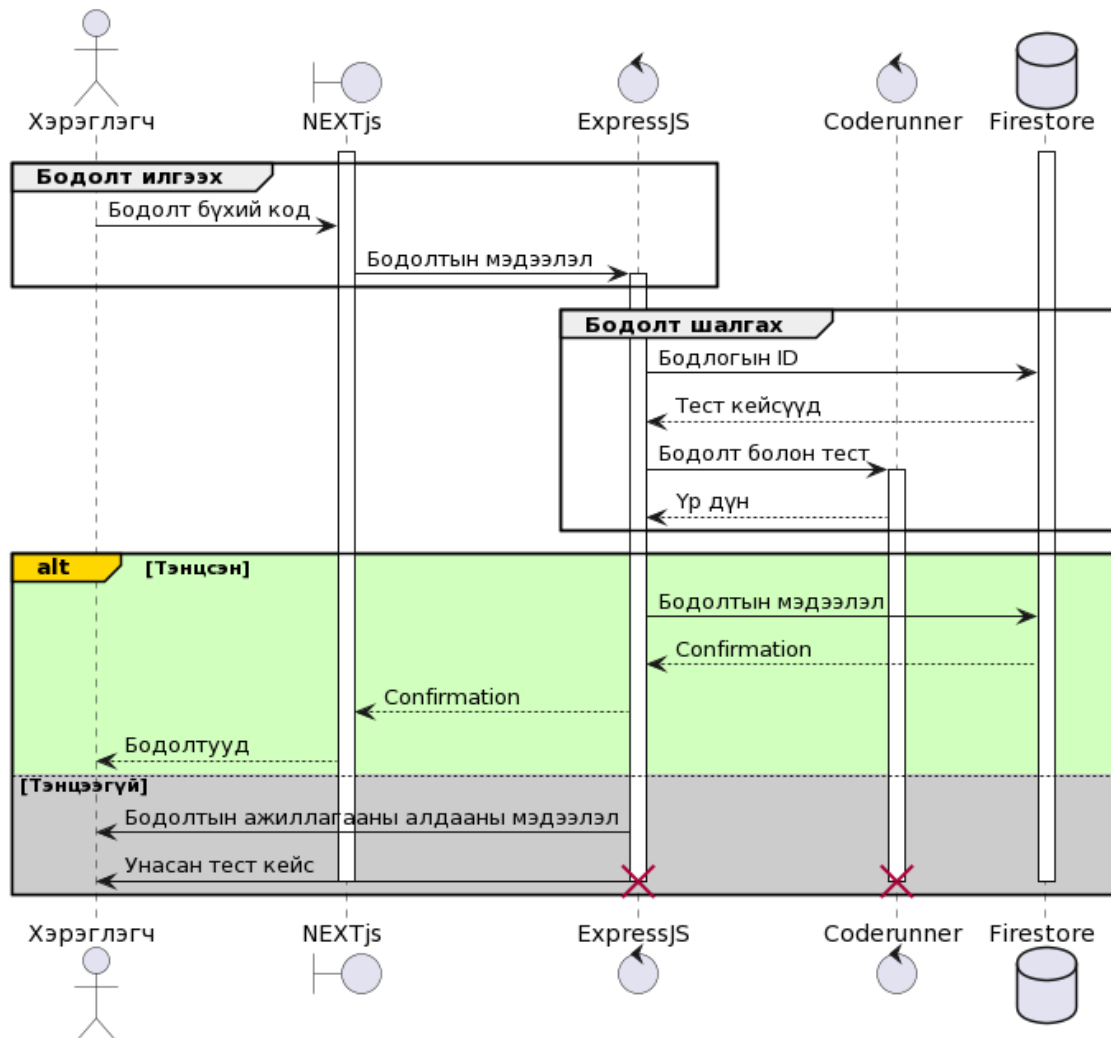
Ажлын явц	Бодолт харах
Зорилго	Тоглогч бодсон бодолгынхоо аргуудаас суралцах зорилгоор бусдын бодолтуудыг харах
Угтвар нөхцөл	Тоглогч тухайн бодлогыг бодсон байх
Тоглогч	Системд нэвтэрч орон session үүсгэсэн хэрэглэгч
Тайлбарлалт	1. Систем тухайн бодлогын бодолтуудыг өөрийн бодолтын хамт харуулна. 2. Тухайн бодолтуудын ямар хэл болон ажилласан хугацааг харуулна.

Ажлын явц: 4. Бодлого бодох

Ажлын явц	Бодлого бодох
Зорилго	Тоглогч системийн бодлогын сангаас бодлого сонгож бодон тухайн бодлогын талаар суралцах, бусдаас шийдлүүдийг сурч авах
Угтвар нөхцөл	Системд нэвтэрч орон бодлогоо сонгосон байх.
Тоглогч	Системд нэвтэрч орон session үүсгэсэн хэрэглэгч
Тайлбарлалт	<ol style="list-style-type: none"> 1. Систем хэрэглэгчид бодлогын тодорхой тайлбар болон жишээ оролт болон гаралтыг харуулна. 2. Систем оролтод ашиглах boilerplate¹ код бүхий код засварлагчийг үүсгэж өгнө. 3. Тоглогч тухайн бодлогын бодолт бүхий кодыг бичнэ. 4. Тоглогч тухайн кодыг ажиллуулах товчийг дарна. 5. Тоглогч тухайн бодлогын бодолтууд хуудас руу шилжинэ.
Өргөтгөл	<ol style="list-style-type: none"> 4b. Тоглогч кодоо дахин засварлаж дараагийн алхмаа хийнэ. 2a. Тоглогч өөрийн дуртай код засварлагчийн загваруудаас сонгоно. 1b. Тоглогч өөрийн мэддэг програмчлалын хэлээ сонгоно.
Хувилбар	4a. Тоглогчийн кодын алдааны мэдээлэл болон харгалзах тест кейсийг систем харуулан ажлын явц үргэлжлэнэ.

¹Ерөнхий хүрээнд ашиглах боломжтой, өөрчлөлт оруулж болох бүхий код

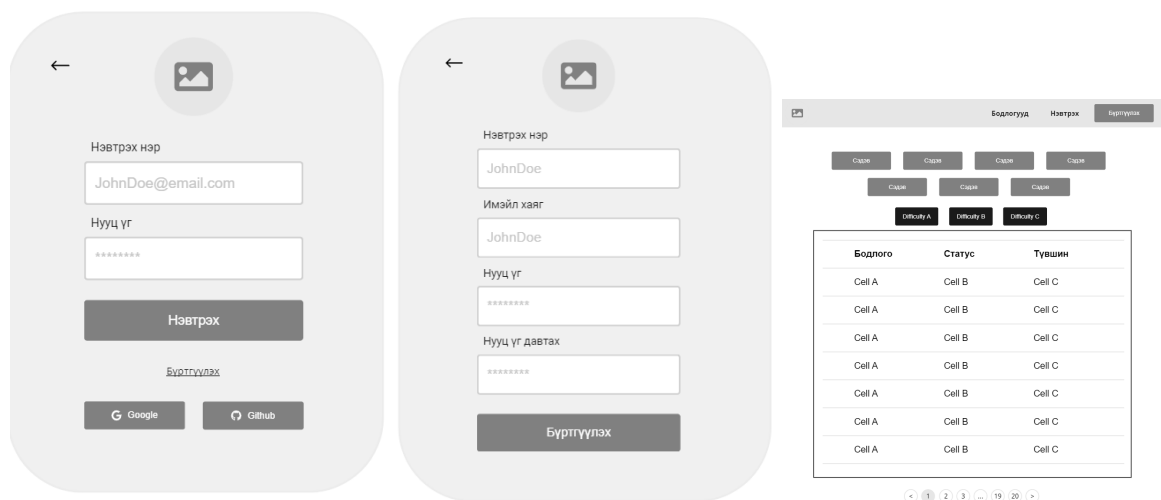
Хэрэглэгчийн кодыг тест кейс дээр ажиллуулан дүгнэх



Зураг 4.3: Хэрэглэгчийн кодыг дүгнэж буй сценарийн дарааллын диаграм

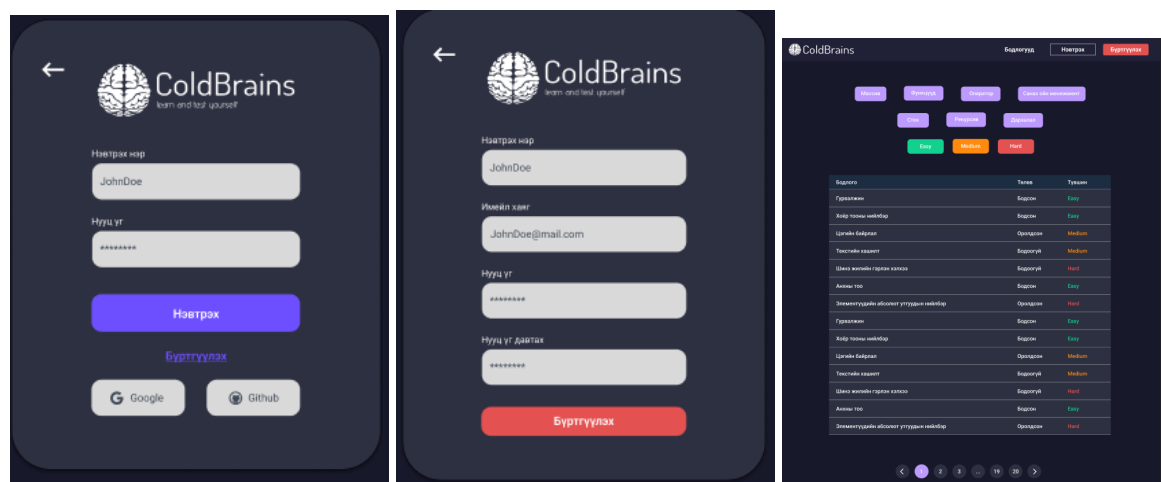
4.2 Хэрэглэгчийн интерфейс

4.2.1 Wireframe



Зураг 4.4: Wireframe

4.2.2 Mockup дизайн



Зураг 4.5: Mockup

Wireframe-ийг илүү дэлгэрэнгүй [2]-с харах боломжтой. Нэвтрэх, бүртгүүлэх, бодлогууд, профайл, бодолт, хариунууд, хариу зэрэг интерфэйсүүдийг зохиомжилсон.

Өнгөний сонголтуудыг **Visual Studio Code** код засварлагчийн хамгийн түгээмэл сонголтууд[3] болон **Programming Color Palette**-ээс сэдэвлэн оноож өгсөн билээ.

4.3 Технологиуд болон шийдлүүд

4.3.1 NEXTjs хувилбар 13

NEXTjs нь React сангийн фреймворк бөгөөд хамгийн түгээмэл хэрэглэгддэг интерактив фреймворк юм. Хэрэглэгчийн интерфэйсийг хэрэгжүүлэхэд хамгийн өргөн сонголтууд болон том экосистем²тэйгээрээ бусад технологиудаасаа илүү байж чаддаг билээ. NEXTjs-ийн тусламжтайгаар хэрэглэгчийн интерфэйсийг интерактив олон компонентуудад хувааж, хэрэглэгчийн үйл ажиллагааг хянах боломжтой.

NEXTjs-ийн хувилбар 13-ыг ашигласнаар дата дамжуулах болон сервер талдаа компонентуудыг зурах, cache хийх аргачлал зэрэг илүү хөгжүүлэгчид хялбар байх бөгөөд маш олон түгээмэл ашиглагддаг сан, package-ууд нь NEXTjs/React-д зориулан нийтлэгдсэн байдаг.

4.3.2 Next.js auth

Энэхүү сан нь NEXTjs дээр **session control** хийхэд хамгийн тохиромжтой бөгөөд хэрэглэгчид зориулсан JWT(Json Web Token) токеныг бэлдэж cookie дээр байршуулж, мөн гадны олон түгээмэл системүүдтэй хэрэглэгчийн нээлттэй датаг хуваалцах боломжийг олгодог. Үүний Google, Github зэрэг мэдээллээр хангагчтай харьцах боломж бүхий хэсгийг NEXTjs дээр амархан ашиглах боломжтой. Хэрэглэгчийн нэвтрэх токенг cookie байдлаар хадгалж тухайн токеноос хэрэглэгчийн мэдээллийг decrypt хийж ашиглах боломжоор хангаж өгдөг.

²Тухайн технологитой холбоо бүхий мэдээлэл, хэрэглээний бодит жишээ, заах арга зүйн нэгдэл

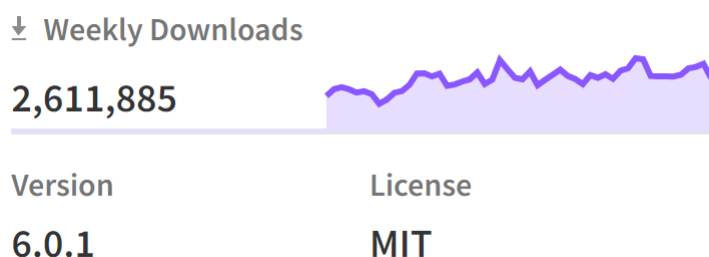
4.3.3 MUI дизайн

Material UI нь Google-ийн Material дизайныг хэрэгжүүлдэг React компонентуудын сан юм. Интерфейс хэрэгжүүлдэг frontend инженерүүдийн хувьд цагийг хэмнэсэн түгээмэл багаж бөгөөд React-ийн **Server Side Rendering**-ийг дэмждэг билээ.

Цаашлаад богино хугацаанд өөрийн өнгөний загварчилгаа(customization) бүхий Responsive интерфэйсийг хэрэгжүүлэхэд тусалдаг билээ. Одоогоор React-ийн хамгийн олон хэрэглэгчтэй сан гэж тодорхойлогддог билээ.

4.3.4 Codemirror

Codemirror гэх энэхүү javascript UI(хэрэглэгчийн интерфэйс) бүхий сан нь "Coldbrains"-д гол тоглогчийн үүргийг гүйцэтгэж байгаа бөгөөд энэхүү сангийн тухай дурьдвал 7 хоног бүр 2-3 сая гаруй таталттай байдаг[7].



Зураг 4.6: Codemirror сан

Цаашлаад Microsoft-ийн Monaco Editor(Visual Studio Code дээр ашиглагддаг), Tern.js зэрэг сангуудын тусламжтайгаар Intellisense бүхий код засварлагчийг бий болгох боломжтой, өөрөө тухайн код бичигчдээ зориулан extension³ -үүдийг бичиж өгөх боломжтой бөгөөд олон төрлийн платформуудыг дэмждэг билээ.

"Coldbrains" систем дээр энэхүү Codemirror сан нь маш хэрэгтэй хэрэглэгчийн интерфэйсийн үүргийг гүйцэтгэж байгаа бөгөөд Codemirror сан дээр бичсэн uiw/react-codemirror сан нь

³Боломжит өргөтгөл

энэхүү хэрэглээг илүү уян хатан болгож, олон хэлний синтакс болон олон өнгөний сонголт зэрэг боломжуудыг бий болгож байгаа. Код засварлагчийн болон хэрэглэгчийн интерфэйсийг уян хатан байдлаар хэрэгжүүлэхэд энэхүү @uiw/codemirror шийдэл болж байгаа.

4.3.5 AWS болон Firebase

Хэрэглэгчийн кодыг функциональ програмчлалын аргаар зохиомжлогдсон сервер дээр ажиллуулах бөгөөд энэ нь AWS-ийн Lambda буюу stateless байдлаар ажилладаг серверт тохиромжтой. AWS нь тухайн хүсэлтийг хүлээж аван тодорхой холбоо үүсгэх ба handshake хийлгүйгээр REST API-аар Firebase-ийн Firestore NoSQL өгөгдлийн сан руу дата дамжуулах билээ.

AWS lambda нь хүсэлт ирэх үед хамгийн хурдан хариу өгөхийн тулд Cold start⁴ хийхээс сэргийлж хамгийн сүүлд ажиллагаатай байсан container-ийг ашигладаг бөгөөд тухайн container-ууд ажиллагаанд ороогүй удсан тохиолдолд нөөцийг хэмнэн өөрийн санах ойг цэвэрлэдэг байна.

Singapore дээр байрлах серверийг ашигласан болно. Firestore 50000 уншилт/сар, 20000 бичилт/сард үнэгүй ашиглах боломжийг олгодог. AWS lambda нь мөн адил бөгөөд MVP(Minimum viable product)-ийг хэрэгжүүлэхэд эдгээр платформууд нь хамгийн тохиромжтой гэж үзсэн.

4.3.6 Хэрэглэгчийн кодыг ажиллуулах

Хэрэглэгчийн кодыг хэл тус бүр дээр аюулгүй ажиллагаан үүднээс AWS-ийн lambda серверийг ашиглаж байгаа. Эдгээр нь

1. Python - python-shell
2. C/C++ - emscripten
3. Javascript - javascript virtual machine 2

⁴Эхнээс нь бүр мөсөн асаах

4.3. ТЕХНОЛОГИУД БОЛОН ШИЙДЛҮҮД БҮЛЭГ 4. СИСТЕМИЙН ЗОХИОМЖ

зэрэг сан болон технологиудыг кодыг ажиллуулахаар ашигласан байгаа билээ. Эдгээр нь customizable буюу оролт/гаралт, ашиглах боломжтой сангууд, ажиллах хугацааны хязгаарлалтуудыг тус тус дэмждэг бөгөөд AWS lambda сервер дээр байршуулснаар тухайн серверийн нөөцийн хуваарилалтыг шийдэж өгч байгаа билээ.

4.3.7 Bcrypt

bcrypt нь "Blowfish" болон "crypt" гэх 2 үгнээс үүдэлтэй бөгөөд өмнөх UNIX нууц үгийн системийн crypt технологиос үндэслэн бий болсон функц билээ[8]. Одоогоор нууц үгийг hash хийхэд түгээмэл хэрэглэгдэж буй энэхүү функц нь hash хийхэд ашиглаж буй хугацаагаараа харьцангуй удаан гэдгээрээ ялгардаг бөгөөд аливаа гадны этгээдэд brute-force attack хийхээс сэргийлж байгаа билээ. Энэхүү функц нь одоогийн технологийн чадалд нийцэх бөгөөд супер компьютеруудад болон ирээдүйд гарч ирэх технологийн хөгжлөөс үүдэн аюулгүй байдлыг бүрэн хангаж чадахгүй байх эрсдэлтэй. Bcrypt нь salt буюу нэмэлт encrypt хийх тэмдэгт мөрүүдийг ашигладаг. "Coldbrains" системийн хувьд хэрэглэгчийг бүртгэх болон нэвтэрч ороход энэхүү функцийг ашиглаж байгаа билээ.

4.3.8 ExpressJS

Хүсэлтийг Nodejs орчинд боловсруулахыг дэмждэг энэхүү фреймворк нь экосистем баялаг байдгаараа олон асуудлуудын шийдэл болж өгдөг. Үүн дээр үндэслэн үүссэн NestJS, Sails.js, Restify зэрэг олон технологиуд байдаг бөгөөд "Coldbrains" системд middleware-уудыг нь түлхүү ашигласан билээ.

1. throttling - Хүсэлтийн тоог тодорхой хугацааны завсарт хязгаарлаж өгдөг бөгөөд **express-rate-limit** санг орж ирж буй хүсэлтүүдийн IPv4 хаягаар танин хязгаарладаг. Ихэвчлэн серверийн ачааллыг бууруулах, нөөцийг хэмнэх, хэрэгцээгүй тооцооллыг багасгах зорилгоор ашиглагддаг.
2. route - ExpressJS-ийн хүсэлтүүдийг боловсруулах модулуудад хуваах боломжтой. Энэ

нь программистуудын хувьд илүү цэгцтэй хамтын ажиллагааг дэмжиж өгдөг билээ.

4.3.9 *Caching*

Уншилт бичилтээр хязгаарлагдах шийдэлд хамгийн хэрэгцээт зүйл нь хэрэглэгчийн статик хэрэгцээт өгөгдлүүдийг *Cache* хийх байдаг. NEXTjs-ийн 13 хувилбар нь javascript хэлний **fetch** функцийг дахин тодорхойлж, хэрэгцээт датаг өгөгдлийн сангаас биш сервер дээр үүсгэн хадгалах боломжийг олгодог. Эдгээр давуу талуудыг Vercel платформ нь дэмжиж, хэрэглэгчид тодорхой цагийн завсарт ижил өгөгдлийг дамжуулж өгдөг. Ингэснээр тухайн Firebase-ийн шийдлийг илүү үр дүнтэйгээр ашиглах боломжтой бөгөөд хэрэглэгчийнхээ тоог үржүүлэх нэмэгдүүлэх боломжийг олгоно.

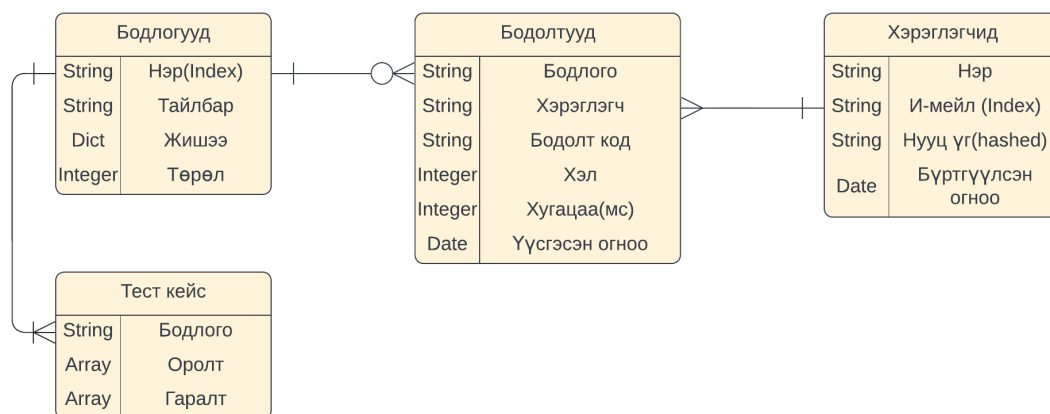
NEXTjs-ийн caching техникийг ашигласнаар "Coldbrains" нь Firebase-ээс өдөртөө зөвхөн ганцхан уншилт хийгээд бодлогуудын мэдээллийг авах боломжтой болох юм. Хэрэглэгч бодолтоо харахын тулд мөн хугацааны интервал буюу 5 минут тутамд шинээр татагдаж cache болж буй өгөгдлөөс харах боломжтой. Ингэснээр $1440 / 5 = 288$ уншилт/өдөр хамгийн ихдээ хийх билээ.

4.4 Өгөгдлийн сангийн зохиомж

Өгөгдлийн сан нь NoSQL бөгөөд Firebase-ийн Firestore өгөгдлийн санг ашигласан болно. Тухайн өгөгдлийн сан нь **collection**, **document** зэргээр өгөгдлийг загварчилдаг бөгөөд дотроо collection нь олон document-үүдийн цуглуулгыг илэрхийлнэ. Мөн collection нь document-ийг агуулах төдийгүй document нь өөртөө collection харьяалуулах боломжтой бөгөөд түүнийг sub-collection гэж нэрлэдэг.

Firestore-ийн document нь өөрийн дотроо collection-г агуулахгүй хэдий ч тухайн collection-ийн заагч байдлаар оршиж болдог билээ. Document-д заагдсан collection-г **subcollection** гэж Firebase платформ дээр тодорхойлсон байдаг[9].

4.4.1 ER диаграм



Зураг 4.7: Өгөгдлийн нэгж хоорондын хамаарлыг дүрсэлж буй диаграм

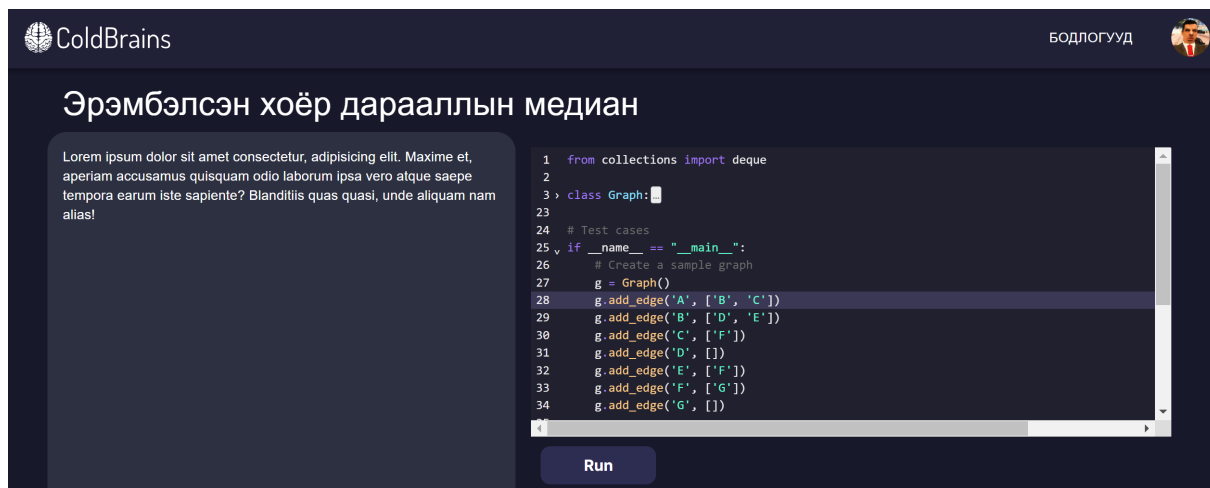
Дээрх өгөгдлийн хамаарлын үндсэн дээр NoSQL firestore өгөгдлийн санд **Problems, test-Cases, Users** гэх Collection-уудыг үүсгэж зохиомжилсон бөгөөд тус бүрт агуулагдах Document-үүд нь index-лэгдсэн. Бие даасан Document-ийг хайлт хийхэд $O(1)$ хугацаанд ажиллана.

Хэрэглэгчээс орж ирэх бодолтыг **solutions** Subcollection байдлаар зохиомжилсон бөгөөд хэрэглэгчээс уншилт хийхдээ агуулагдаж буй олон document рүү зэрэг хандах нөхцөлтэй учир **collectionGroup[10]** байдлаар индекслэн уншилт хийх хурдыг дэмжиж байгаа билээ.

5. ХЭРЭГЖҮҮЛЭЛТ

5.1 Frontend

Хэрэглэгчийн интерфeйсийг NEXTjs болон MUI сангийн тусламжтайгаар хэрэгжүүлсэн бөгөөд нэвтрэх логик болон ажиллах явцын логикуудыг хэрэгжүүлсэн билээ. Google, Github-ыг ашиглан нэвтрэх, өөрөө бүртгэл үүсгэн нэвтрэх, route-үүдийг хамгаалах, middleware-ийг хэрэгжүүлэх, сервер талдаа хүсэлт явуулах зэрэг ажлуудыг хэрэгжүүлээд байгаа билээ.



Зураг 5.1: Бодлого бодох хэсэг

5.1.1 Session

Хэрэглэгчийн бүртгүүлэх мэдээлэл болон нэвтрэх мэдээллийг өмнө дурьдсанчлан NEXTjs-ийн next-auth санг ашиглан хэрэгжүүлсэн бөгөөд хэрэглэгчийн мэдээлэл болох *email*, *Нэр*, *Зураг* зэргийг үүсгэж session дээрээ JWT ашиглан encrypt хийх хэрэглэгч тал дээр cookie байдлаар хадгалж байгаа билээ. Энэхүү cookie нь next auth-ийн default behavior-оор 1 сар буюу 30 хоногийн хугацаатайгаар хэрэглэгчийн төхөөрөмж дээр оршино.

Хэрэглэгчийг бүртгэж авахдаа тухайн хэрэглэгчээс нэр, и-мейл, нууц үгийг авч байгаа

бөгөөд хэрэглэгч талын middleware буюу NEXTjs middleware[11] дээр validation¹ -ийг хэрэгжүүлсэн байгаа билээ.

next auth сан болон firebase платформууд нь өөрсдийн гуравдагч платформуудын credentials provider-луу хандах боломжуудыг бий болгодог бөгөөд хэрэглэгч өөрийн и-мейл хаягийг хийн нэвтэрж орж буй тохиолдолд тухайн хэрэглэгчийн и-мейлийг баталгаажуулах процесс хийх нь шаардлагагүй гэж үзэв. Хэрэв тухайн хэрэглэгч энэхүү и-мейл хаягаар өөр платформ дээр ашигладаг бол "Coldbrains" системд бүртгэл үүсгэх шаардлагагүйгээр шууд нэвтрэх боломжтой билээ.

Хэрэглэгчийг тухайн хандах боломжтой эсэхийг системийн Frontend нь өөрийн сервер тал дээр тухайн хэрэглэгчийн session болох cookie-г decrypt хийн тогтмол шалгах бөгөөд тухайн хэрэглэгчийн session хугацаа нь дууссан, эсвэл мэдээлэл нь буруу байгаа тохиолдолд хэрэглэгчийг бүртгэлгүй хэрэглэгч гэж тооцон одоо хийж буй ажлын явцыг дуусгавар болгох боломжтой.

5.1.2 Data fetching

Хэрэглэгч нь Frontend рүү хүсэлт явуулах тоолонд сервер талаас өгөгдлийн сан руу хандах шаардлага бий болж байгаа билээ. Өгөгдлийн сангаас уншилт хийх аргыг хэдий оптимал байдлаар шийдсэн ч firestore нь өөрөө уншилт, бичилтүүдийн хязгаарлалтуудыг тавьж өгсөн тул системийн найдвартай ажиллагааны үүднээс тухайн уншилт бичилтүүдэд хязгаарлалт тавьж өгөх шаардлагатай.

NEXTjs-ийн сервер талын route-үүд буюу тухайн route-үүд дээр firebase платформын админ SDK² -г ашиглан индекслэн document-үүд рүүгээ хандаж буй байдлаар хэрэгжүүлсэн. NEXTjs хувилбар 13 дээр шийдсэн javascript хэлний built-in **fetch(...)**[12] аргыг ашиглан сервер талдаа cache үүсгэж байгаа бөгөөд тухайн cache үүссэн тохиолдолд тэрхүү өгөгдлийг авах хүсэлт илгээж буй хэрэглэгчийн интерфэйсүүд нь өгөгдлийн сангаас уншилт хийх шаардлагагүйгээр

¹Хэрэглэгчийн оролтыг баталгаажуулах процесс

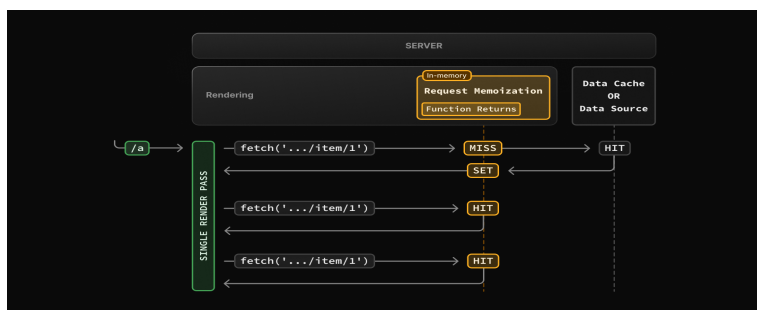
²Software Development Kit

тухайн cache-ийг уншиж авна.

Caching хийх нь Realtime өгөгдөлтэй харьцдаг системүүдийн хувьд тохиоромжгүй бөгөөд "Coldbrains" системийн хувьд бодлогын сан нь хэрэглэгч талдаа урт хугацаанд үл шинэчлэгдэх өгөгдөл гэж үзэн NEXTjs-ийн default хугацаа буюу 1 сарын хугацаатайгаар cache үүсгэж байгаа билээ. Харин идэвхитэй өөрчлөгдөх өгөгдөл буюу хэрэглэгчийн бодолтыг тухайн хэрэглэгчид realtime байдлаар харуулах шаардлага үүсч байна. Энэхүү тохиолдолд ерөнхий байдлаар caching хийх нь тохиромжгүй хэдий ч зардал багатайгаар шийдэх боломжууд мөн бий. Хэрэглэгчээс тухайн нөөцийг авах хүсэлт орж ирэх үед хамгийн боломжит богино интервалтайгаар cache үүсгэх боломжтой бөгөөд тухайн cache-ийг 3 секундын хугацаатайгаар хэрэгжүүлснээр богино хугацаанд орж ирэх зэрэгцээ хүсэлтүүдэд computing³ хийх нөөцийг хэмнэх юм.

$86400\text{секунд}/\text{өдөр} : 3\text{секунд}/\text{уншилт} = 28800\text{уншилт}/\text{өдөр}$ болох бөгөөд хэрэгцээт уншилтуудын хязгаарт хэдэн ч хэрэглэгч байсан багтаах боломжтой болж байгаа юм.

3 секундын интервалиар тохируулж өгөх нь бодолгоо дөнгөж бодож буй хүнд эргээгээд бодолтуудыг харуулах нь илүү урт хугацаа мэт байгаа боловч зэрэгцээ тухайн хэрэглэгчээс өмнө 3 секунд дотор өөр хэрэглэгч ижил бодлогыг бодож cache үүсгэх нь ховор тохиолдол бөгөөд дараачийн хэрэглэгч нь тухайн хуудсыг дахин ачааллуулах нь UX-д ашиглагддаг түгээмэл практикуудын нэг юм.



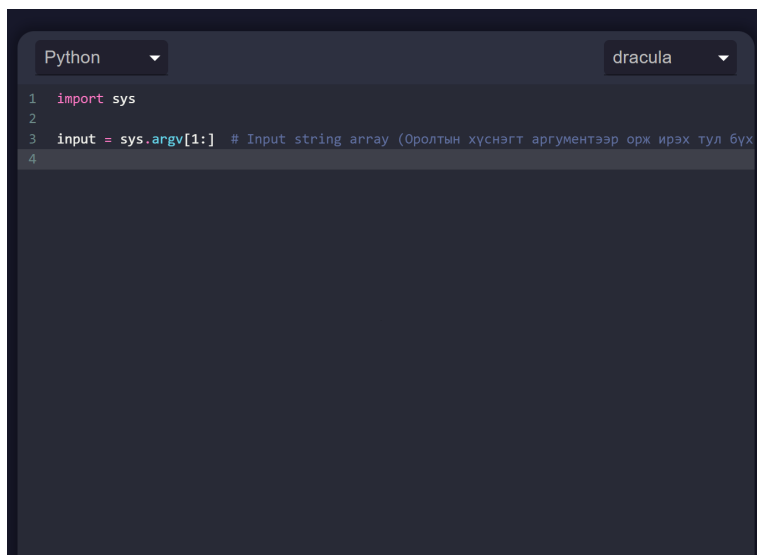
Зураг 5.2: NEXTjs-ийн сервер талдаа cache ашиглаж буй байдал

³Тооцоолж, боловсруулах процесс

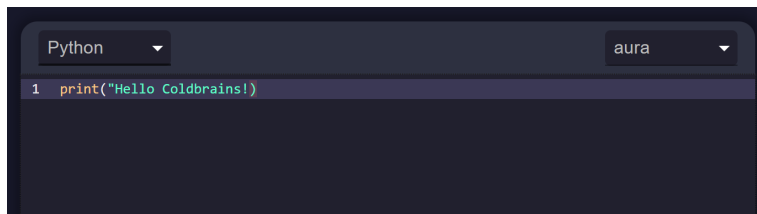
5.1.3 Code editor

Технологиуд дээр дурьдсанчлан хэрэглэгчийн кодыг засварчлахдаа **Codemirror** javascript хэлний санг ашигласан бөгөөд тухайн санг React дээр интеграц хийж илүү олон боломжуудтайгаар ашиглахын тулд **@uiw/react-codemirror** санг ашигласан билээ. Өнгөний сонголт, түгээмэл код засварлагчийн theme-үүдийг түүний өргөтгөл сан болох **@uiw/codemirror-themes-all** санг ашигласан. Энэхүү сан нь React дээр бэлтгэж өгсөн код засварлагч компонентээр хангаж өгдөг бөгөөд тухайн код засварлагч дээр customize буюу өөрөө гараар бичсэн өргөтгөл функцуудыг нэмэх боломжтой. Системд ашиглахдаа өөр бусад Tern.js зэрэг ухаалаг санал болгогч төлөвлөсөн байгаа бөгөөд одоогоор бодлого бодох үед хэрэглэгчид ухаалгаар санал болгогч ашиглуулах нь хэрэглэгчид хэтэрхий хялбарчилж өгч байсан тул хэрэглэгчийн суурь мэдлэгийг бататгах үүднээс системийн бодлого бодох хэсэгт нэмээгүй байгаа билээ.

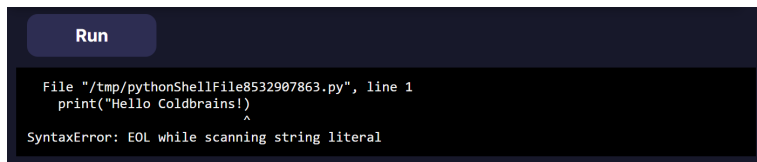
Одоогоор python хэлний код ажлуулагч серверийг validation, sandboxing зэргээр хэрэгжүүлсэн байгаа билээ. Үүнд зориулан python хэлэн дээр хэрэглэгчид boilerplate кодыг анх тайлбартайгаар үүсгэж өгч байгаа бөгөөд хэрэглэгчийн тухайн кодыг хэрхэн ашиглаж програмын оролтыг ашиглахыг зөвлөсөн байгаа юм.



Зураг 5.3: Dracula theme дээр python хэлний boilerplate бүхий код засварлагч

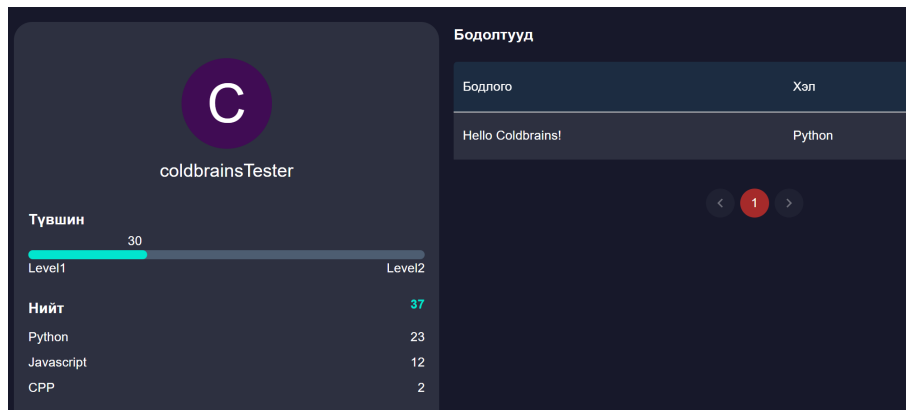


Зураг 5.4: Syntax-ийн алдаа бүхий код



Зураг 5.5: Syntax-ийн алдааг харуулж буй системийн интерфэйс





Эдгээр хэрэглэгчийн коднууд нь -> NEXTjs сервер -> ExpressJS сервер -> Coderunner сервер зэрэг байдлаар дамжин ажиллуулж эргээд Coderunner -> ExpressJS сервер-> NEXTjs сервер -> интерфэйс гэх дарааллаар хариугаа хэрэглэгчид харуулж байгаа юм.



Зураг 5.6: Хэрэглэгчийн профайл хуудас

5.2 Backend

Хэрэглэгчийн нууц үгийг hash-лах bcrypt ашиглан, auth серверийг MySQL дээр тест байдлаар хэрэгжүүлсэн байгаа бөгөөд ExpressJS дээр modular-programming техникийг ашиглаж байгаа билээ. Одоогоор хэрэглэгчийн кодыг Python-shell дээр ажиллуулах, хэрэглэгчийг бүртгэх, нэвтрэх, request throttling хийх зэрэг ажлуудыг гүйцэтгээд байгаа билээ.

NAME	STATUS	TYPE	RUNTIME	REGION
 cold-brains-server	 Deployed	Web Service	Node	Singapore
 coldbrains-auth	 Deployed	Web Service	Node	Singapore

Зураг 5.7: Сервисүүдийг байршуулсан байдал

Зохиомжийн дагуу хэрэглэгчийн кодыг дүгнэх процессийг хэрэгжүүлсэн бөгөөд ExpressJS-ийн middleware-уудыг ашиглан хэрэглэгчээс орж ирэх хүсэлтүүдийг шүүж хязгаарлаж өгсөн билээ.

5.2.1 Middleware

express-rate-limit санг ашиглан хэрэглэгчийн хандалтад хязгаар тавьж өгсөн бөгөөд одоогоор 5 секундэд 5 удаа хандах боломжтой байлгаж хязгаарласан байгаа билээ. Хязгаарлалтыг тавьж өгснөөр Coderunner серверийн нөөцийг хэрэггүй зүйлд ашиглахаас сэргийлж байгаа бөгөөд brute-force болон бусад өөр зорилго агуулсан хандалтаас сэргийлэх боломжтой.

expressJS дээр цаашлаад ирж болон хүсэлтийн payload буюу body датаг хязгаарлалт хийн шийдэж өгч байгаа. ExpressJS сервер нь хэрэглэгчээс бодолтыг авч тухайн бодолтын тест кейсүүдийг firestore-оос уншиж цааш Coderunner луу дамжуулах үүрэг хүлээж байгаа бөгөөд хэрэглэгчээс орж ирэх кодын хэмжээг хязгаарлах, мөн хаанаас ирэх хаягийг CORS ашиглан блоклож байгаа юм. Хэрэглэгчийн кодын хэмжээ нь бодлого бодох хэмжээнээс хэтэрч ямар нэгэн гуравдагч санг хэрэгжүүлсэн тохиолдолд кодын хэмжээгээр хязгаарлах байдлаар шийдсэн.

5.2.2 Coderunner

python-shell санг ашиглан тухайн ирж буй кодыг ажиллуулахыг AWS Lambda сервертэй цогцоор нь шийдсэн. Тухайн код их хэмжээний нөөцийг ашиглах, тухайн төхөөрөмжийг улмаар хэвийн ажиллагааг зөрчилдүүлсэн тохиолдолд AWS Lambda сервер нь cold boot хийн stateless учир ямарваа нэгэн өмнөх ажилласан мэдээллээ хадгалахгүйгээр дахин шинээр ажиллаж эхлэх билээ.

Өөр нэгэн шийдэл бол container технологийг өөрөө хэрэгжүүлэн тухайн container-т CPU, RAM зэрэг computing нөөцүүдийг хязгаарлаж, сүлжээнд хандах болон бусад сүлжээ рүү хариулахаас сэргийлэх боломжтой. Энэ нь энгийн бөгөөд үр дүнтэй ажиллах боломжтой арга билээ. Хөгжүүлэлт болон тохиргооны хувьд ихээхэн зардал гарах боломжтой бөгөөд тухайн Coderunner серверийг багцлан container-ийг үүсгэсэн тохиолдолд эдгээр командуудыг ажиллуулан тухайн серверийн Sandboxing, Network restriction зэрэг асуудлуудыг шийдэх боломжтой.

```
bash
docker run -p 8080:3000 my-express-app
```

Зураг 5.8: Хандах боломжтой портыг тодорхойлох

```
bash
docker run --cpu-shares=512 --memory=512m my-express-app
```

Зураг 5.9: Нөөцийг хязгаарлах

```
bash
docker run --read-only my-express-app
```

Зураг 5.10: Зөвхөн унших боломжтой болгох

Энэхүү шийдэл нь зэрэгцээ хандалт болон тухайн скриптийг хэрхэн хувааж хэрэглэх болон бусад олон асуудлуудыг бий болгох бөгөөд илүү enterprise түвшний хөгжүүлэлтүүдийг шаардах билээ. "Coldbrains" системийн хувьд MVP-г зорьсон бөгөөд AWS Lambda ашигладаг Render платформ дээр container болгон Coderunner серверийг байршуулсан байгаа юм. Уг шийдэл нь дээр дурьдсан шийдлээс илүү болхи хэдий ч зардал багатай, чанарын үзүүлэлтүүдээр дүйцэхүйц байгаа билээ.

Уг шийдэл дээр Python програмчлалын хэлний built-in сүлжээнд ажиллах, файл системд нэвтрэх, үйлдлийн системд бичих сангуудыг блоклож байгаа бөгөөд үүнийг *regex* ашиглан хэрэглэгчийн кодноос тухайн сангуудын ашиглалтад хайлт хийдэг middleware байдлаар хэрэгжүүлсэн.

5.2.3 Байршуулалт, ашиглалт

Render платформын AWS Lambda серверүүдийг ашиглаж байгаа бөгөөд идэвхигүй болсон үедээ **spin down** хийдэг энэхүү серверүүд нь эргээд асах буюу **cold boot** хийхдээ анх build хийхэд байсантайгаа ижил төлөвт ордог билээ[13]. Spin down хийсэн container-үүд нь энэхүү Render платформын төлөвлөгөөнд эргээд 10-15 секунд асахад зарцуулдаг гэж заасан. Сүлжээнд алдах цагийг багасгах зорилгоор Coderunner сервер болон ExpressJS серверүүдийг тус бүрийг нь Render платформ дээр байршуулсан бөгөөд хоорондоо local орчинд шууд харилцан датагаа богино хугацаанд дамжуулах боломжтой билээ. Сүлжээнд тест кейс болон хэрэглэгчийн кодыг дамжуулах нь ихээхэн хугацаа шаардах бөгөөд хамтад нь Singapore дээрх инстанци дээр байршуулав.

Хэрэглэгчээс ирэх дата нь эхэлж ExpressJS-ийг cold boot хийлгээд тухайн cold boot хийсэн container нь дахин CodeRunner-ийг асаана. Энэ нь хүлээлгийн араас хүлээлэг үүсгэн *GATEWAY TIMEOUT 504* алдааг хэрэглэгчид харуулахад хүргэх бөгөөд нийт 40-45 секунд хугацааг код ажиллуулах хүсэлт боловсруулахад зарцуулж байгаа билээ. Үүнийг шийдсэн арга нь тухайн 2 container-ийг зэрэгцээ cold boot хийлгэсэн бөгөөд код ажиллуулах хүсэлт явуулахдаа NEXTjs route middleware-ээс тухайн Coderunner container лүү ping хийж нийт 10-15 секунд

cold boot хийхэд зарцуулж байгаа билээ.

Цаашлаад stateless байдлаар кодыг зохиомжилсон бөгөөд санах ой дээр үүсдэг ямарваа нэгэн тогтмол ашиглагдах инстанциудыг cold boot хийх үед дахин үүсгэх байдлаар програмчилж бичив.

5.2.4 Firestore

Frontend болон Backend тус бүр Firebase admin SDK ашиглан query бичин байгаа бөгөөд тухайн query-г хурдацтай ажиллуулахын тулд collectionGroup үүсгэн тухайн бодлогын бодолтууд болон хэрэглэгчийн бодолтуудыг богино хугацаанд авах боломжтой болгосон билээ.

(default)	problemSolutions	calcFactorial
+ Start collection	+ Add document	+ Start collection
problemSolutions >	calcFactorial >	solutions
problems	helloColdbrains	

Зураг 5.11: Firestore-ийн ProblemSolutions collection болон түүний заагч document-үүд

calcFactorial	solutions	jackerenh@gmail.com
+ Start collection	+ Add document	+ Start collection
solutions >	jackerenh@gmail.com >	+ Add field
+ Add field		code: "import sys\ninput = sys.argv[1:] # Input string array\n(Оролтын хүснэгт аргументээр орж ирэх тул бүх утга string төрөлтэй.)\ndef fact(num): if(num == 0): return 1; return num * fact(num - 1);\nprint(fact(int(input[0])))"

Зураг 5.12: Firestore-ийн subcollection group-ийн ашиглалт, агуулж буй document-үүд

Дүгнэлт

Энэхүү инженерчлэлийн судалгааны ажлаар алгоритмын бодлого бодох болон хэрэглэгчийн кодыг ажиллуулах дотоод ба гадаадын системүүдийн практик, хэрэгжүүлэлтүүдийг сурч, өөрийнхөө нөхцөл байдалд хэрхэн хэрэгжүүлэх боломжууд байгаа вэ гэдгийг сурч мэдэж авлаа. Нийт ажлын хүрээнд судалгаа шинжилгээ, архитектур зохиомж, хэрэгжүүлэлт зэрэг багтана. Зорилтот хэрэглэгчид нь түгээмэл байсан бөгөөд бүтээлээ туршиж, сайжруулахад найз нөхөд, мэргэжилтэнгүүд, зөвлөгөө өгсөн багш нар гээд олон хүмүүсийн санал хүсэлт нөлөөлсөн билээ. Судалгаа хийх явцад өөрийн сурч авсан мэдлэгээ гадны олон технологиуд дээр ашиглах, тулгарсан асуудлуудыг шийдэхэд ашигласан бөгөөд "Coldbrains" системийг хийж, ажиллагаанд оруулахад 0 төгрөг зарцуулсан болно. Өнөөгийн хүн бүрт нээлттэй платформ, технологиудыг ашиглан уг системийг хэрэгжүүлэх нь цаашлаад дараа дараачийн судалгаа хийж буй оюутан залууст ямар их нээлттэй боломжууд байгааг харуулж, илүү шинэлэг судалгаа болон бүтээл хийх урам өгнө гэж найдаж байна.

Ном зүй

- [1] Benefits of Learning in Mother Tongue language <https://www.krsf.in/blog/benefits-of-learning-in-mother-tongue-language/>
- [2] Coldbrains, miro workspace https://miro.com/app/board/uXjVNf5zADk=?share_link_id=375550425121
- [3] Most popular VS Code themes (2021) <https://visualstudiomagazine.com/articles/2021/07/07/vs-code-themes.aspx>
- [4] Accepted academy 2023 оны 12 сар <https://www.accepted.mn/it>
- [5] Spoj - "Алгоритмын цагаан толгой" бодлогын сан <https://www.spoj.com/RGB7>
- [6] AWS Lambda - Programming Model <https://docs.aws.amazon.com/lambda/latest/dg/foundation-progmodel.html>
- [7] Codemirror - Extensible code editor <https://codemirror.net>
- [8] Understanding bcrypt. <https://auth0.com/blog/hashing-in-action-understanding-bcrypt>
- [9] Firestore - Data Modeling <https://firebase.google.com/docs/firestore/data-model>
- [10] Firestore - Index types <https://firebase.google.com/docs/firestore/query-data/index-overview>
- [11] NEXTjs - Middleware <https://nextjs.org/docs/app/building-your-application/routing/middleware>
- [12] NEXTjs - Data fetching, Caching and Revalidating <https://nextjs.org/docs/app/building-your-application/data-fetching/fetching-caching-and-revalidating>
- [13] Render - Spinning down on idle <https://render.com/docs/free>

A. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

```
1 import NextAuth, { AuthOptions } from "next-auth";
2 import GoogleProvider from "next-auth/providers/google";
3 import GithubProvider from "next-auth/providers/github";
4 import CredentialsProvider from "next-auth/providers/credentials";
5
6 export const authOptions: AuthOptions = {
7   providers: [
8     CredentialsProvider({
9       id: "coldbrains-credentials",
10      name: "Coldbrains",
11      credentials: {
12        username: { label: "Username", type: "text" },
13        password: { label: "Password", type: "password" },
14      },
15      async authorize(credentials, req) {
16        const reqBody = {
17          username: credentials?.username,
18          password: credentials?.password,
19        };
20
21        const res = await fetch(`${process.env.APP_URL}/api/login`, {
22          method: "POST",
23          body: JSON.stringify(reqBody),
24        });
25        if (res.status === 200) {
26          const userCred = await res.json();
27          return userCred;
28        }
29        return null;
30      },
31    ),
32    GoogleProvider({
33      clientId: process.env.GOOGLE_CLIENT_ID!,
34      clientSecret: process.env.GOOGLE_CLIENT_SECRET!,
35    }),
36    GithubProvider({
37      clientId: process.env.GITHUB_CLIENT_ID!,
38      clientSecret: process.env.GITHUB_CLIENT_SECRET!,
39    }),
40  ],
41  session: {
42    strategy: "jwt",
43  },
44  pages: {
45    signIn: "/login",
46    signOut: "/problems",
47  },
48 };
49
50 const handler = NextAuth(authOptions);
51
52 export { handler as GET, handler as POST };
```

Код А.1: Next.js Auth ашигласан байдал

```

1  "use client";
2  import CodeMirror from "@uiw/react-codemirror";
3  import {
4    aura,
5    abyss,
6    androidstudio,
7    dracula,
8    material,
9    copilot,
10   githubDark,
11   githubLight,
12   sublime,
13   xcodeDark,
14   xcodeLight,
15 } from "@uiw/codemirror-themes-all";
16 import { python } from "@codemirror/lang-python";
17 import { useState, useEffect } from "react";
18 import styles from "../page.module.css";
19 import { javascript } from "@codemirror/lang-javascript";
20 import { oneDark } from "@uiw/react-codemirror";
21 import { Language, Theme } from "../states/enum";
22 import {
23   Box,
24   Select,
25   MenuItem,
26   FormControl,
27   InputLabel,
28   FormHelperText,
29 } from "@mui/material";
30 import { SelectChangeEvent } from "@mui/material/Select";
31
32 export default function CodeEditor({
33   boilerplate,
34   problemId,
35   problemTitle,
36 }: {
37   boilerplate: string | undefined;
38   problemId: string;
39   problemTitle: string;
40 }) {
41   const [pythonCode, setPythonCode] = useState(boilerplate || "");
42   const [output, setOutput] = useState("");
43
44   const [language, setLanguage] = useState("Python");
45   const [editorTheme, setEditorTheme] = useState(Theme.aura);
46   const [themeExt, setThemeExt] = useState(aura);
47
48   const handleLanguageChange = (event: SelectChangeEvent) => {
49     setLanguage(event.target.value);
50   };
51
52   const handleThemeChange = (event: SelectChangeEvent) => {
53     setEditorTheme(Number(event.target.value));
54     switch (Number(event.target.value)) {
55       case Theme.abyss:
56         setThemeExt(abyss);
57         break;
58       case Theme.androidstudio:

```

```

59         setThemeExt(androidstudio);
60         break;
61     case Theme.aura:
62         setThemeExt(aura);
63         break;
64     case Theme.dracula:
65         setThemeExt(dracula);
66         break;
67     case Theme.githubDark:
68         setThemeExt(githubDark);
69         break;
70     case Theme.githubLight:
71         setThemeExt(githubLight);
72         break;
73     case Theme.material:
74         setThemeExt(material);
75         break;
76     case Theme.oneDark:
77         setThemeExt(oneDark);
78         break;
79     case Theme.copilot:
80         setThemeExt(copilot);
81         break;
82     case Theme.sublime:
83         setThemeExt(sublime);
84         break;
85     case Theme.xcodeDark:
86         setThemeExt(xcodeDark);
87         break;
88     case Theme.xcodeLight:
89         setThemeExt(xcodeLight);
90         break;
91     default:
92         setThemeExt(aura);
93         break;
94     }
95 };
96
97 const handlePythonOnChange = (newPythonCode: string) =>
98     setPythonCode(newPythonCode);
99 const handlePythonCodeRun = async () => {
100     const body = {
101         id: problemId,
102         code: pythonCode,
103         language: language,
104         title: problemTitle,
105     };
106     const response = await fetch("/api/run", {
107         method: "POST",
108         body: JSON.stringify(body),
109     });
110
111     console.log(response.status);
112     if (response.status !== 202) {
113         const data = await response.json();
114         console.log(data);
115         setOutput(data.traceback ?? "");
116     } else {

```

```

117     setOutput("Accepted");
118   }
119 };
120
121 return (
122   <Box>
123     <Box
124       sx={{
125         bgcolor: "var(--primaryBlack)",
126         borderTopLeftRadius: "15px",
127         borderTopRightRadius: "15px",
128         boxShadow: "0 0 10px 5px rgb(0, 0, 0, 0.3)",
129       }}
130     >
131       <Box display={"flex"} justifyContent={"space-between"}>
132         <Select
133           variant="standard"
134           value={language}
135           onChange={handleLanguageChange}
136           sx={{
137             my: 1,
138             mx: 2,
139             px: 1,
140             width: "120px",
141             color: "#BBBBBB",
142             bgcolor: "#21202e",
143             borderRadius: "5px",
144             ".MuiSvgIcon-root ": {
145               fill: "white",
146             },
147           }}
148         >
149           <MenuItem value={"Python"}>Python</MenuItem>
150           <MenuItem value={"JS"}>Javascript</MenuItem>
151           <MenuItem value={"CPP"}>C/C++</MenuItem>
152         </Select>
153         <Select
154           variant="standard"
155           value={editorTheme.toString()}
156           onChange={handleThemeChange}
157           sx={{
158             my: 1,
159             mx: 2,
160             px: 1,
161             width: "120px",
162             color: "#BBBBBB",
163             bgcolor: "#21202e",
164             borderRadius: "5px",
165             ".MuiSvgIcon-root ": {
166               fill: "white",
167             },
168           }}
169         >
170           <MenuItem value={Theme.abysse}>abysse</MenuItem>
171           <MenuItem value={Theme.androidstudio}>androidstudio</MenuItem>
172           <MenuItem value={Theme.aura}>aura</MenuItem>
173           <MenuItem value={Theme.copilot}>copilot</MenuItem>
174           <MenuItem value={Theme.dracula}>dracula</MenuItem>

```

```

175         <MenuItem value={Theme.githubDark}>githubDark</MenuItem>
176         <MenuItem value={Theme.githubLight}>githubLight</MenuItem>
177         <MenuItem value={Theme.material}>material</MenuItem>
178         <MenuItem value={Theme.oneDark}>oneDark</MenuItem>
179         <MenuItem value={Theme.sublime}>sublime</MenuItem>
180         <MenuItem value={Theme.xcodeDark}>xcodeDark</MenuItem>
181         <MenuItem value={Theme.xcodeLight}>xcodeLight</MenuItem>
182     </Select>
183 </Box>
184 <CodeMirror
185   value={pythonCode}
186   onChange={handlePythonOnChange}
187   height="450px"
188   extensions={[python()]}
189   theme={themeExt}
190 />
191 </Box>
192
193 <div className={styles.btn_submit} onClick={handlePythonCodeRun}>
194   Run
195 </div>
196 <div className={styles.target_output}>
197   <pre>{output}</pre>
198 </div>
199 </Box>
200 );
201 }

```

Код А.2: Codemirror буюу @uiw/react-codemirror-ийн компонентийг ашиглаж буй байдал

```

1  import { NextRequest, NextResponse } from "next/server";
2  import firestore from "@/configs/firebase";
3  import { getDocs, collection } from "firebase/firestore";
4
5
6  export async function GET(request: NextRequest) {
7    const querySnapshot = await getDocs(collection(firestore, "problems"));
8
9    const problemsArray = querySnapshot.docs.map((doc) => {
10      return {
11        id: doc.id,
12        ...doc.data()
13      }
14    });
15
16    return NextResponse.json(problemsArray);
17  }

```

Код А.3: NEXTjs бодлогууд авах серверийн endpoint

```

1  "use client";
2  import MainLayout from "@/components/layouts/mainLayout/mainLayout";
3  import styles from "@/page.module.css";
4  import CodeMirrorTest from "@/components/editor/codeEditor";
5  import ProblemList from "@/components/problemList/problemList";
6  import { useSession } from "next-auth/react";
7  import { useRouter } from "next/navigation";
8  import { useEffect, useState } from "react";

```

```

9
10 export default function Problems() {
11   const [data, setData] = useState([]);
12   const { status } = useSession();
13   const router = useRouter();
14
15   const columns = [
16     { field: "title", headerName: " ", flex: 1, minWidth: 300 },
17     { field: "status", headerName: " ", width: 150 },
18     { field: "difficulty", headerName: " ", width: 150 },
19   ];
20
21   useEffect(() => {
22     fetch("/api/getProblems")
23       .then((res) => res.json())
24       .then((resJson) => {
25         setData(resJson);
26       });
27   }, []);
28
29   const handleRowClick = (params: any) => {
30     if (status === "authenticated") {
31       router.push(`/problem/${params.row.id}`);
32     } else {
33       router.push("/login");
34     }
35   };
36
37   return (
38     <MainLayout>
39       <ProblemList
40         data={data}
41         columns={columns}
42         onClick={handleRowClick}
43         rowsPerPage={12}
44       />
45     </MainLayout>
46   );
47 }

```

Код А.4: Бодлогуудыг cache-ээс fetch хийх component

```

1 import { getSession } from "@providers/client-provider";
2 import { NextRequest, NextResponse } from "next/server";
3 import firestore from "@configs/firebase";
4 import { getDocs, collectionGroup } from "firebase/firestore";
5
6 export async function GET(request: NextRequest) {
7   const sesdata = await getSession();
8   const email = sesdata?.user?.email;
9
10   const querySnapshot = await getDocs(collectionGroup(firestore, "solutions"));
11   ;
12   const userSolutionDocs = querySnapshot.docs.filter((doc) => doc.id === email);
13   ;
14   const userSolutions = userSolutionDocs.map((userDoc) => userDoc.data());
15   return NextResponse.json(userSolutions);
16 }

```

Код А.5: CollectionGroup query бичиж хэрэглэгчийн бодолтуудыг авч буй route

```

1 import { NextRequest, NextResponse } from "next/server";
2 import { getSession } from "@providers/client-provider";
3
4 export async function POST(request: NextRequest) {
5   const { id, code, language, title } = await request.json();
6
7   const sesdata = await getSession();
8   if (!sesdata) {
9     return new NextResponse("Forbidden", { status: 403 });
10  }
11  const userCred = sesdata?.user;
12
13  switch (language) {
14    case "Python":
15      fetch(process.env.PROD_PYTHON_SERVER!);
16      break;
17    default:
18      break;
19  }
20
21  // const res = await fetch(`${process.env.DEV_SERVER}/${language}`, {
22  const res = await fetch(`${process.env.PROD_SERVER}/${language}`, {
23    method: "POST",
24    cache: "no-cache",
25    headers: new Headers({ "content-type": "application/json" }),
26    body: JSON.stringify({ id, code, userCred, title }),
27  });
28
29  return new NextResponse(res.body, { status: res.status });
30 }

```

Код А.6: Бодолт явуулах NEXTjs endpoint

```

1 import { Avatar } from "@mui/material";
2
3 const stringToColor = (string: string) => {
4   let hash = 0;
5   let i;
6
7   /* eslint-disable no-bitwise */
8   for (i = 0; i < string.length; i += 1) {
9     hash = string.charCodeAt(i) + ((hash << 5) - hash);
10  }
11
12  let color = "#";
13
14  for (i = 0; i < 3; i += 1) {
15    const value = (hash >> (i * 8)) & 0xff;
16    color += `00${value.toString(16)}`.slice(-2);
17  }
18  /* eslint-enable no-bitwise */
19
20  return color;
21 };
22

```



```

23 type Size = "sm" | "md" | "lg";
24
25 const stringAvatarProps = (name: string, size?: Size) => {
26   const fontSize = size === "sm" ? 18 : size === "lg" ? 60 : 20;
27
28   return {
29     sx: {
30       backgroundColor: stringToColor(name),
31       width: size === "sm" ? 30 : size === "lg" ? 100 : 40,
32       height: size === "sm" ? 30 : size === "lg" ? 100 : 40,
33       fontSize: fontSize,
34     },
35     children: `${name.charAt(0).toUpperCase()}`,
36   };
37 };
38
39 const StringAvatar = ({ name, size }: { name: string; size?: Size }) => {
40   return <Avatar {...stringAvatarProps(name, size)} />;
41 }
42
43 export default StringAvatar;

```

Код А.7: Хэрэглэгчийн нэрнээс Avatar үүсгэж буй байдал

```

1  import { Router } from "express";
2  import getFirebaseDB from "../db/firebase.js";
3  import { doc, getDoc, serverTimestamp, setDoc } from "firebase/firestore";
4  import axios from "axios";
5  import "dotenv/config";
6
7  const router = Router();
8
9  router.post("/", async (req, res) => {
10    const code = req.body.code;
11    const problemId = req.body.id;
12    const problemTitle = req.body.title;
13    const { name: username, email, image } = req.body.userCred;
14
15    if (!code || !problemId) {
16      return res.sendStatus(409);
17    }
18
19    const docRef = doc(getFirebaseDB(), "testCases", problemId);
20    const docSnapshot = await getDoc(docRef);
21
22    if (!docSnapshot.exists()) {
23      return res.sendStatus(404);
24    }
25
26    const testCases = docSnapshot.get("testCase");
27    const reqBody = {
28      code,
29      testCases,
30    };
31
32    try {
33      // const axiosRes = await axios.post("http://localhost:8081", reqBody);
34      const axiosRes = await axios.post(process.env.PROD_PYTHON_SERVER, reqBody)
35    }

```

```

35
36   if (axiosRes.status === 202) {
37     const docFields = {
38       code,
39       createdAt: serverTimestamp(),
40       language: "Python",
41       time: axiosRes.data,
42       username,
43       problemId,
44       problemTitle,
45       ...(image ? {image} : {})
46     };
47
48     const solutionDocRef = doc(
49       getFirestoreDB(),
50       "problemSolutions",
51       problemId,
52       "solutions",
53       email
54     );
55     await setDoc(solutionDocRef, docFields);
56   }
57   return res.status(axiosRes.status).json(axiosRes.data);
58 } catch (err) {
59   return res.sendStatus(503);
60 }
61 });
62
63 export default router;

```

Код А.8: Python хэлний middleware

```

1  import { initializeApp } from "firebase/app";
2  import { getFirestore } from "firebase/firestore";
3  import "dotenv/config";
4
5  let db;
6
7  const firebaseConfigs = {
8    apiKey: process.env.FIREBASE_API_KEY,
9    authDomain: process.env.FIREBASE_AUTH_DOMAIN,
10   databaseURL: process.env.FIREBASE_DATABASE_URL,
11   projectId: process.env.FIREBASE_PROJECT_ID,
12   storageBucket: process.env.FIREBASE_STORAGE_BUCKET,
13   messagingSenderId: process.env.FIREBASE_MESSAGING_SENDER_ID,
14   appId: process.env.FIREBASE_APP_ID,
15   measurementId: process.env.FIREBASE_MEASUREMENT_ID,
16 };
17
18 export default function getFirestoreDB() {
19   if (!db) {
20     console.log("Initializing Firebase...");
21     const app = initializeApp(firebaseConfigs);
22     db = getFirestore(app);
23     console.log("Firebase initialized.");
24   }
25   return db;
26 }

```

Код А.9: Stateless байдлаар firebase admin SDK ашиглалт

```
1 import express from "express";
2 import cors from "cors";
3 import { rateLimit } from "express-rate-limit";
4
5 import getFirestore from "../db/firebase.js";
6 import pythonRouter from "../routes/python.js";
7
8 const app = express();
9 const port = 8080;
10 const limiter = rateLimit({
11   windowMs: 1000 * 15,
12   limit: 5,
13   message: "Too many requests. Please wait for 5 seconds and try again.",
14 })
15
16 app.use(cors());
17 app.use(express.json());
18 app.use(limiter)
19
20 app.use("/python", pythonRouter);
21 app.use("/js", null)
22 app.use("/c", null)
23
24 app.get("/", (req, res) => {
25   res.sendStatus(200);
26 })
27
28 app.listen(port, () => {
29   console.log("Python server listening.");
30 });
```

Код А.10: ExpressJS серверийн оролт