

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

Дэлгэрцогтын Цэцэнбилэг

Дипломын ажлын бүртгэлийн систем
(Web application for Diploma thesis registration)

Програм хангамж (D061302)
Бакалаврын судалгааны ажил

Улаанбаатар

2022 он

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

Дипломын ажлын бүртгэлийн систем
(Web application for Diploma thesis registration)

Програм хангамж (D061302)
Бакалаврын судалгааны ажил

Удирдагч: _____ Б.Батням

Гүйцэтгэсэн: _____ Д.Цэцэнбилэг (18B1NUM1505)

Улаанбаатар

2022 он

Зохиогчийн баталгаа

Миний бие Дэлгэрцогтын Цэцэнбилэг ”Дипломын ажлын бүртгэлийн 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
1. СУДАЛГАА	3
1.1 Ижил төстэй системийн судалгаа	3
1.2 Технологийн судалгаа	4
1.3 Бүлгийн дүгнэлт	11
2. СИСТЕМИЙН ШИНЖИЛГЭЭ	12
2.1 Хэрэглэгчийн шаардлага	12
2.2 Системийн шаардлага	13
2.3 Бүлгийн дүгнэлт	15
3. СИСТЕМИЙН ЗОХИОМЖ	16
3.1 Статик загвар	16
3.2 Динамик загвар	28
3.3 Өгөгдлийн сангийн зохиомж	33
3.4 Хэрэглэгчийн харьцах хэсгийн хар зураг	44
3.5 Бүлгийн дүгнэлт	49
4. ХЭРЭГЖҮҮЛЭЛТ, ҮР ДҮН	50
4.1 Хөгжүүлэгдсэн байдал	50
4.2 Бүлгийн дүгнэлт	54
ДҮГНЭЛТ	55
НОМ ЗҮЙ	55
ХАВСРАЛТ	56
А. КОДЫН ХЭРЭГЖҮҮЛЭЛТ	57

ЗУРГИЙН ЖАГСААЛТ

1.1	Муис ахисан түвшний хайлт хийх хэсэг	4
1.2	Дипломын лавлагааны хэсэг	4
1.3	ШУТИС электрон каталог	5
1.4	ШУТИС судалгааны ажлын хэсэг	6
1.5	Application graph[2]	8
1.6	Provider & Controller[2]	9
3.1	Ажлын явцын диаграм	17
3.2	Системийн классын диаграм	27
3.3	Админ системрүү нэвтрэх үйл ажиллагааны диаграм	28
3.4	Диплом бүртгэлийн үйл ажиллагааны диаграм	29
3.5	Админ системрүү нэвтрэх дарааллын диаграм	30
3.6	Дипломын ажил бүртгэх дарааллын диаграм	31
3.7	Админ үүсгэх дарааллын диаграм	32
3.8	Админ тэнхим бүртгэх дарааллын диаграм	33
3.9	Админ болон их сургууль хүснэгтийн холбоос	34
3.10	Их сургууль болон тэнхим хүснэгтийн холбоос	35
3.11	Тэнхим болон багш хүснэгтийн холбоос	35
3.12	Багш болон Диплом хүснэгтийн холбоос	36
3.13	Өгөгдлийн сангийн диаграм	37
3.14	Үндсэн цонх	45
3.15	Дипломын дэлгэрэнгүй цонх	45
3.16	Нэвтрэх цонх	46
3.17	Админ цонх	46
3.18	Багш бүртгэх цонх	47
3.19	Диплом бүртгэх цонх	47

3.20	Өгөгдөл харагдах цонх	48
3.21	Админ нэмэх цонх	48
4.1	Диплом хайх хэсэг	50
4.2	Нэвтрэх цонх	51
4.3	Админ цонх	52
4.4	Багш нэмэх цонх	52
4.5	Диплом нэмэх хуудас	53
4.6	Админ нэмэх хуудас	54
4.7	Тэнхим үүсгэх болон санал хүсэлт илгээх хуудас	54
4.8	Өгөгдөл харах хуудас	55

ХҮСНЭГТИЙН ЖАГСААЛТ

3.1	Ажлын явц 2: Системийн админ үүсгэх	18
3.2	Ажлын явц 3: Админ эрх идэвхгүй болгох	19
3.3	Ажлын явц 4: Санал хүсэлт илгээх	20
3.4	Ажлын явц 5: Нэвтрэх	21
3.5	Ажлын явц 6: Дипломын ажил бүртгэх	22
3.6	Ажлын явц 12: Дипломын ажлын бүртгэлийг засварлах	23
3.7	Ажлын явц 7: Тэнхим үүсгэх	24
3.8	Ажлын явц 9: Багш нэмэх	25
3.9	Admin хүснэгт	38
3.10	University хүснэгт	39
3.11	Department хүснэгт	40
3.12	Teacher хүснэгт	41
3.13	Diploma хүснэгт	42
3.14	Teacher-Diploma хүснэгт	43

Кодын жагсаалт

A.1	Prisma модел (schema.prisma файл)	57
A.2	Authentication Service (Хэрэглэгч шалгах)	58
A.3	Users Service	60
A.4	Department Service	61
A.5	Diploma Service	62
A.6	Teacher Service	64
A.7	Нэвтрэх хуудсын хэрэгжүүлэлт	65
A.8	Өгөгдөл харах хуудас хэрэгжүүлэлт	66
A.9	Диплом бүртгэх хуудас	68

УДИРТГАЛ

Монгол улсын хэмжээнд сүүлийн 5 жилийн дунджаар их, дээд сургууль, коллежийг төгсөгчдийн тоо 29000 гаруй гэсэн тоо хэмжээ улсын хэмжээнд бүртгэгдсэн байна. Програм хангамж хөтөлбөрийн хувьд сүүлийн 3 жилийн дунджаар жилд 59 хүүхэд төгсдөг. Тэгвэл тэдгээр дипломын судалгааны ажлуудыг сургуулийн архивт биет байдлаар хадгаладаг. Ингэж бичиг баримтыг хадгалах нь ашиглалтын элэгдэлд өртөх, ямар нэгэн осол аваар, хадгалалтын горим алдагдах гэх мэт гэнэтийн нөхцөл байдлаас болж үрэгдэж алга болох эрсдэлтэй юм. Тиймээс эдгээр судалгааны ажлуудыг баталгаажуулан програм хангамжийн систем ашиглан хадгалах нь хамгийн үр дүнтэй аюулгүй бөгөөд найдвартай, хүртээмжтэй байдлаар олон хүн ашиглах боломжтой шалгарсан арга гэж үзсэн тул миний бие бакалаврын судалгааны ажлын хүрээнд энэхүү дипломын ажлын бүртгэлийн системийг хийж байна билээ. Энэхүү систем нь дан ганц Монгол улсын их сургуулийн төгсөгчдийн дипломын ажлыг бүртгэх бус Монгол улсын хэмжээн дэх их, дээр сургуулиудын дипломын ажлын бүртгэлийг давхар хийх, дипломын ажлын төвлөрсөн лавлагааны санг бий болгох цаашдын зорилготой билээ. Уг ажлын хүрээнд эдгээр зорилтуудыг тавьсан болно.

1. Ижил төстэй системийн судалгаа хийх
2. Дипломын ажлын бүртгэлийн системийн хэрэгцээ шаардлагыг судлах.
3. Технологийн судалгаа хийх
4. Уг системийн шинжилгээ ба зохиомжийг боловсруулах
5. Дээрх алхмуудын дагуу гарсан зохиомж ба архитектурын дагуу хэрэгжүүлэлтийг хийж гүйцэтгэх.

1. СУДАЛГАА

Энэхүү бүлэг хэсэгт дипломын бүртгэлийн ажлын системтэй ижил төстэй системүүд болон, энэхүү сэдвээр судалгааны ажил хийгдсэн эсэхийг судалж хооронд нь харьцуулалт хийх бөгөөд үүний үр дүнд системийн хэрэгцээ шаардлага, давуу тал, өөр ямар нэмэлт боломжууд байж болох талаар мэдэх юм. Мөн системийг хөгжүүлэхэд шаардагдах програмчлалын хэлүүд болон фрэймворкуудын талаарх судалгаа хийх болно. Дипломын ажлын бүртгэлийн систем нь веб програм хангамж бөгөөд үүнд хэрэглэгчийн харьцах талын хөгжүүлэлтэд NextJS, сервер талын хөгжүүлэлтэд NestJS , Prisma. Өгөгдлийн сангийн хувьд Postgresql ашиглана.

1.1 Ижил төстэй системийн судалгаа

Монгол улсын их сургууль

Монгол улсын их сургуулийн хувьд ямар нэгэн дипломын судалгааны ажлын нэгдсэн харах, хайх хэсэг гэж байхгүй. Харин электрон каталогтой бөгөөд түүнээс ахисан түвшний хайлт хийх, цахим магистрын ажил, цахим диссертаци хайж онлайнаар унших боломжтой.[4] Зураг1.1 Дипломын судалгааны ажил онлайнаар үзэх боломжгүй бөгөөд харин дипломын лавлагааны хэсгээс дипломын мэдээлэл хайж болдог. Зураг1.2

Шинжлэх ухаан технологийн их сургууль

Шутис нь өөрсдийн гэсэн цахим каталогтой бөгөөд тухайн каталогоос электрон ном, сурах бичиг хайх зориулалттай. Зураг1.3 Тухайн систем нь судалгааны сан гэсэн хэсэгтэй Зураг1.4 бөгөөд тэр нь бүрэн ажиллагаанд ороогүй бөгөөд ямар ч судалгааны ажил бүртгэгдээгүй байсан. Тухайн системийн хувьд ямар нэгэн дипломын судалгааны ажлыг харуулах хэсэг байгаагүй.

Нүүр · Ахисан түвшний хайлт

Хайх талбар сонгох:

Түлхүүр үг

ба Түлхүүр үг [+][-]

ба Түлхүүр үг [+][-]

[Хайх](#) [\[Энгийн тохиргоо\]](#) [\[Шинэ хайлт\]](#)

Зүйлийн төрөл

Зүйлийн төрлийн хязгаарлалт:

<input type="checkbox"/> Цахим ном /EBSCO/	<input type="checkbox"/> CD, DVD	<input type="checkbox"/> Газрын зураг	<input type="checkbox"/> Диссертаци
<input type="checkbox"/> Дуу бичлэгийн хуурцаг	<input type="checkbox"/> Дууны нот	<input type="checkbox"/> Зурагт ном	<input type="checkbox"/> Зөөврийн компьютер
<input type="checkbox"/> Магистрын ажил	<input type="checkbox"/> Ном	<input type="checkbox"/> Сонин	<input type="checkbox"/> Сэтгүүл
<input type="checkbox"/> Тайлан, зөвлөмж	<input type="checkbox"/> Цахим диссертаци	<input type="checkbox"/> Цахим магистрын ажил	<input type="checkbox"/> Цахим материал /АХБ/
<input type="checkbox"/> Цахим ном	<input type="checkbox"/> Цахим сэтгүүл	<input type="checkbox"/> Цахим эрдэм шинжилгээний бичиг	

Зураг 1.1: Муис ахисан түвшний хайлт хийх хэсэг

Дипломын мэдээлэл шалгах

Дипломын дугаар: [Шалгах](#)

Жишээ нь: 0201212345

Тайлбар

Дипломын дугаар оруулан шалгахад тухайн төгсөгчийн ерөнхий мэдээлэл гарч ирнэ.

Хэрэв дэлгэрэнгүй дүнгийн мэдээлэл харах бол Дипломын баруун доод буланд байрлах доорх зураг №1 бүхий зургийг баркод уншигчаар уншуулж гарч ирэх линкээр орж шалгана.

Жич: 2011 оноос өмнөх төгсөгчдийн мэдээлэл системд бүрэн ороогүй тул Төгсөгчдийн мэдээлэл хэсгээс харах уу.

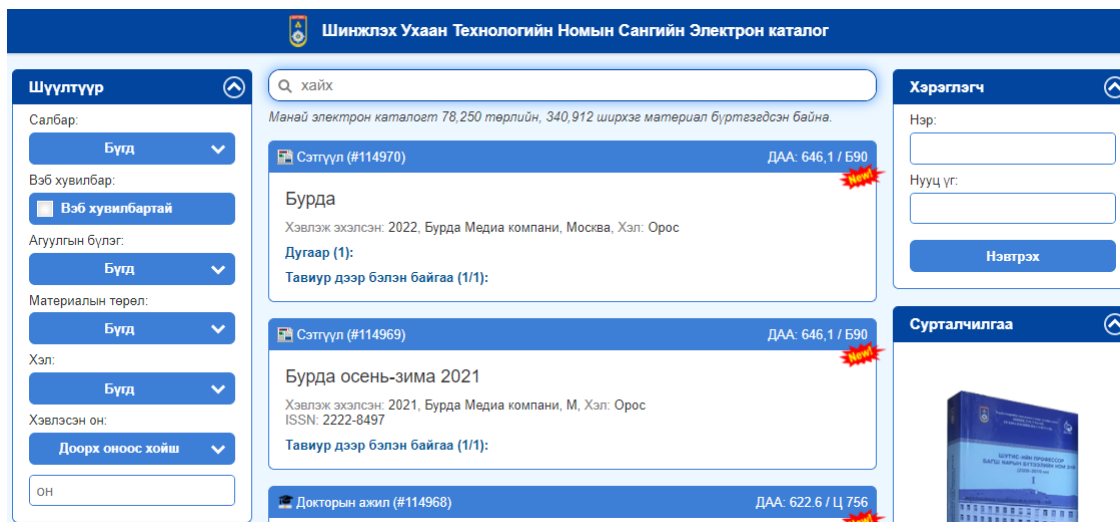
Зураг №1 Дипломын нууцлал бүхий баркод

Зураг 1.2: Дипломын лавлагааны хэсэг

1.2 Технологийн судалгаа

NextJS-ийн судалгаа

NextJS нь нээлттэй эхийн веб хөгжүүлэлтэд ашиглагддаг ReactJS -ийн фрэймворк юм. NextJS нь react-ийн бүхий л боломжуудыг өөртөө агуулсан бөгөөд харин ч илүү их боломжуудыг нэмэлтээр бий болгосон технологи юм. ReactJS нь virtual DOM, syntex exten-



Зураг 1.3: ШУТИС электрон каталог

tion to Javascript, components, state, hooks гэх мэт маш өргөн боломжуудтай хэл юм. Харин эдгээр зүйлс дээр нэмэгдээд NextJS өөрийн гэсэн олон давуу талуудтай.[1] Үүнд:

Page based routing system

Хэрвээ ReactJS дээр хуудас үүсгэх шаардлагатай болвол заавал component үүсгэх react-router-dom гэх нэмэлт санг ашиглан router-рүү нэмж өгдөг бол NextJS дээр pages гэдэг folder-т хуудас үүсгэх өгөхөд л хангалттай юм.

Nested routes

Хэрвээ pages folder дотор nested folder-ууд үүсгэвэл бид доор харагдаж байгаа шиг nested route үүсгэж чадна.

```
pages/blog/first-post.js → /blog/first-post
```

Dynamic routes

Динамик route үүсгэхийн тулд хаалт ашиглан хуудас үүсгэх хэрэгтэй.

```
pages/[username]/settings.js → /:username/settings (/foo/settings)
```



Зураг 1.4: ШУТИС судалгааны ажлын хэсэг

Pre-rendering (Static generation SSG & Server-side rendering SSR)

SSG

Static generation нь тухайн нэг хуудас build хийгдэх үедээ нэг үүсээд дараа дараагийн хүсэлтүүд нь тухайн үүссэн хуудас дахин ашиглагддаг байна. Хэрвээ ямар нэгэн өөрчлөгдөж болох датанаас хамааралтайгаар зурагддаг хуудас бол NextJS нь `getStaticProps`, `getStaticPaths` гэсэн функцуудыг build time дээр нь ашиглах боломжоор хангаж өгсөн.

SSR

Хэрвээ server-side rendering ашиглаж байгаа тохиолдолд `getServerSideProps` гэх функцийг ашиглагдаг бөгөөд build time дээр биш хүсэлт болгон дээр HTML хуудсаа дахин үүсгэдэг байна. Server-side rendering нь Static generation-с хамаагүй удаан ажилладаг бөгөөд client side rendering хийхээс хамаагүй хурдан ажилладаг. ReactJS нь client side rendering хийдэг.

SEO friendly

Nextjs нь хэрэглэгчид болон хайлтын бот-д HTML хуудас үүсгэж өгдөг.

NestJS

NestJS нь үр дүнтэй, хурдан, өргөтгөж болохуйц nodeJS дээр суурилсан сервер талын хөгжүүлэлтийг хийх боломжтой фреймворк юм. Typescript-г бүрэн дэмждэг нээлттэй эхийн OOP¹, FP², FRP³ гэх хөгжүүлэлтийн элементуудийг агуулдаг бөгөөд эдгээрийг ашиглах бүрэн боломжтой юм.[2]

NestJS ашиглах шалгаан

NodeJS гарч ирсэнээр Javascript-г хэрэглэгчийн харагдах хэсэгт, мөн сервер талын хөгжүүлэлтэнд аль алинд нь ашиглах боломж гарч ирсэн. Хэдийгээр node js -д зориулсан маш олон багажууд, сангууд байсан хэдий ч эдгээрийн хамгийн гол тулгараад байсан асуудал нь архитектурын шийдэл байсан. Харин тулгараад байгаа хамгийн гол асуудал болох архитектурыг нь шийдсэн нь NestJS-ийн хамгийн гол философи юм.

Nest нь тестлэж болохуйц, цааш нь хөгжүүлэхэд хялбар, өргөтгөх боломжтой архитектурыг хэрэгжүүлдэг бөгөөд Angular-с санаа хамгийн их санаа авсан. Архитектур нь төслийг microservice-үүдэд маш хялбараар хуваах боломжийг олгодог. Nest нь PostgreSQL, MySQL, MongoDB гэх мэт өгөгдлийн сангуудийг дэмжин ажилладаг.

Nest-ийн үндсэн бүрэлдэхүүн хэсгүүд**1. Modules**

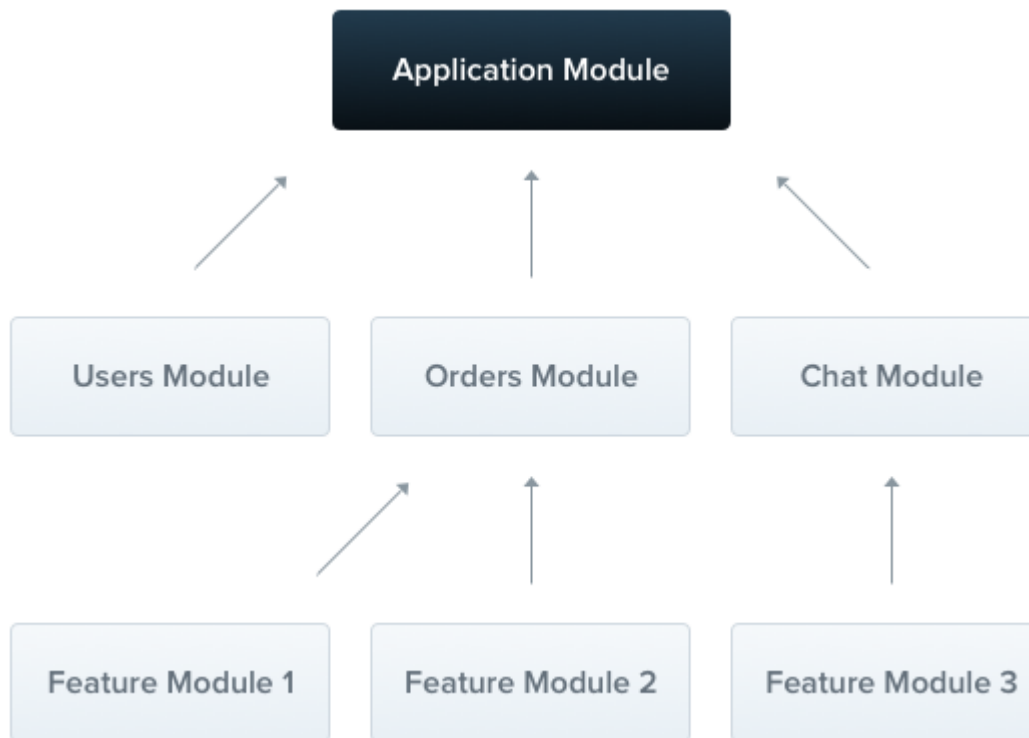
Модуль класс нь @Module() гэх тодорхойлогч | чимэглэгчээр тодорхойлог зарлагддаг.

¹Object Oriented Programming

²Functional Programming

³Functional Reactive Programming

Тухайн `@Module()` гэх тодорхойлогчийг ашиглан Nest нь өөрийн аппликэйшн бүтцээ зохион байгуулдаг. Аппликэйшн бүр ядаж нэг модультай байдаг түүнийг root module гэнэ. Root module нь аппликэйшн граф-г үүсгэх эхлэх цэг нь болж өгдөг. Зураг 1.5

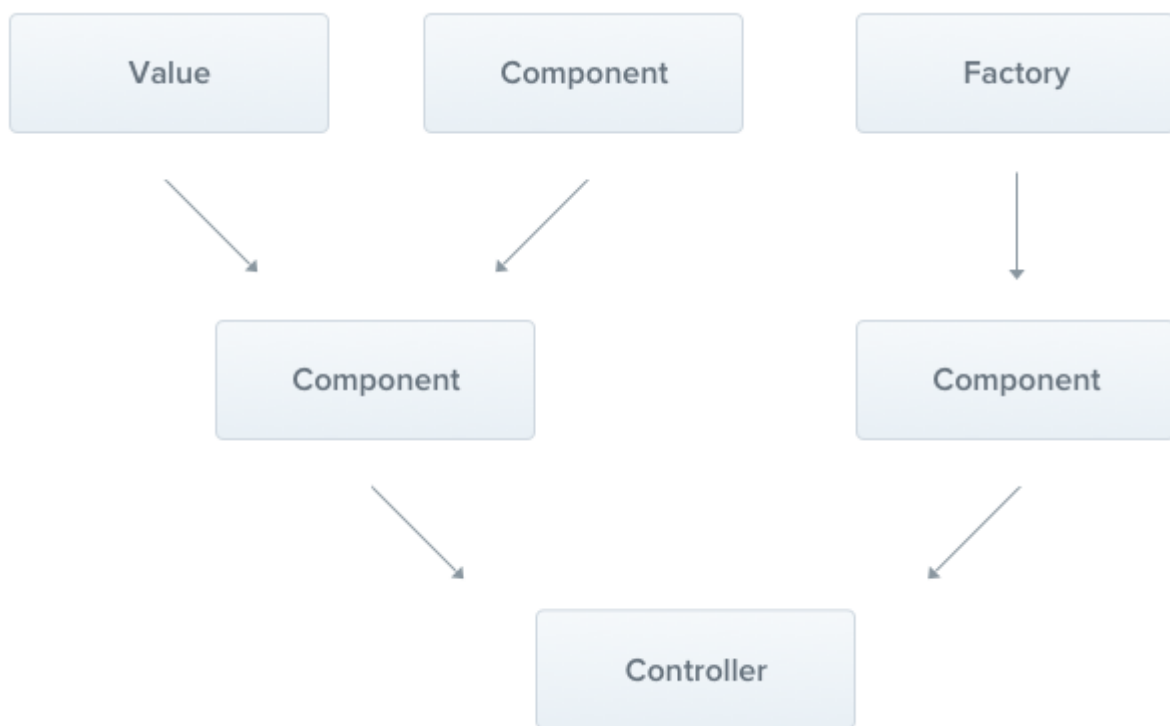


Зураг 1.5: Application graph[2]

2. Providers

Provider буюу Service нь Nest-ийн үндсэн суурь ойлголтуудыг нэг юм. Provider-ийн гол санаа нь аппликэйшны класс хоорондын холбоо хамаарлыг үүсгэх, ашиглах, нэг provider-г нөгөө provider-т ашиглах мөн Controller класс-т provider inject хийх боломжийг олгодог. Provider классыг `@Injectable` түлхүүр үгийг ашиглан тодорхойлдог. Энэ түлхүүр үгээр Nest нь Nest Ioc container ашиглан тухайн классыг зохион байгуулдаг.

3. Controllers



Зураг 1.6: Provider & Controller[2]

Controller-ийн гол зорилго нь ирж буй хүсэлтийг хүлээн авч хэрэглэгч рүү хариу өгөх үүрэгтэй. Нэг controller нь ихэвчлэн олон замтай(route) байдаг бөгөөд routing механизм нь аль controller-ийн аль зам ямар хүсэлтийг хүлээн авах вэ гэдгийг удирддаг. Controller класс үүсгэхийн тулд `@Controller("")` түлхүүр үг ашиглаж үүсгэх бөгөөд Nest routing map үүсгэх боломжийг олгодог.

Prisma

Prisma нь орчин үеийн ORM юм.⁴ .ORM гэдэг нь өгөгдлийн сангуудийн хүснэгтүүдийг аппликэйшн талд объект болгон дүрслэн харуулахыг хэлдэг. Prisma-н үндсэн зорилго нь хөгжүүлэгчдэд өгөгдлийн сантай үр дүнтэй, илүү бүтээлчээр, хялбар байдлаар ажиллах боломжийг олгох юм. Бүх өгөгдлийн сангийн моделиудаа prisma schema file дотор байрлуулдаг бөгөөд энэ нь үнэний нэг эх сурвалжаар өгөгдлийн сан болон аппликэйшн

⁴Object-relational mapper

моделийг хангадаг. Prisma schema file нь ямар нэгэн decorator болон класс ашигладаггүй PSL⁵ ашиглан бичигддэг.[3]

Prisma-г ашиглах шалтгаан⁶

1. Thinking in object instead of mapping relational data
2. Queries not classes to avoid complex model objects
3. Single source of truth for database and application model
4. Type-safe database queries

Postgre SQL

Postgre нь их хэмжээний өгөгдлийг хадгалахад зориулагдсан нээлттэй эхийн уламжлалт холбоост өгөгдлийн сангийн менежмент систем. Хүснэгт болон мөрүүдийг үүсгэх тэдгээрийн хоорондын холбоо хамаарлыг тодорхойлон өгөгдлийн зохион байгуулж хадгаладаг.

PostgreSQL давуу талууд

Performance and scalability

Маш томоохон өгөгдлийн сангууд дээр уншил болон бичилтийн хурд маш чухал байдаг. Үүн дээр Postgre нь үнэхээр үр дүнтэйгээр ажиллаж чаддаг. Postgre нь олон төрлийн үйл ажиллагааны тохируулгыг дэмждэг бөгөөд газарзүй, хязгааргүй зэрэгцээ хандалтуудыг хийх боломжтой учир маш гүн гүнзгий өргөн хүрээтэй дата анализ хийх үед маш үр дүнтэйгээр хурдан ажиллаж чаддаг.

⁵Prisma Schema Language

⁶Дэлгэрэнгүйг эндээс уншина уу. <https://www.prisma.io/docs/concepts/overview/why-prisma>

Concurrency Support

Хэрвээ хэд хэдэн хэрэглэгч нэг өгөгдөл рүү нэг цагт хандан үед бусад уламжлалт өгөгдлийн сангууд уншилт бичилтын зөрчил үүсгэхгүйн тулд бичилт хийх хандалтыг нь түгждэг. Харин postgres нь MVCC⁷ ашиглан зэрэгцээ хандалтыг маш үр дүнтэйгээр зохион байгуулдаг бөгөөд энэ нь уншилт нь бичилтээ түгжихгүй, бичилт нь уншилтаа түгжихгүй гэсэн үг юм.

Deep language support

Postgre нь маш уян хатан, өрсөлдөхүйц, олон програмчлалын хэлийг дэмжин ажилладаг өгөгдлийн сангуудын нэг юм. Энэ нь хөгжүүлэгчдэд өөрсдийн хүссэн, боломжит хэл дээрээ хөгжүүлэлт хийн ажиллах боломжтой болгодог.

1.3 Бүлгийн дүгнэлт

Програмын хөгжүүлэлтэд ашиглагдах технологиудын судалгааг дээрх бүлэгт хийхэд NestJS нь маш уян хатан, өргөтгөх, тестлэх боломжтой кодын архитектурыг бидний хүчин чармайлтгүйгээр гаргаж өгөх бөгөөд зөвхөн бизнес логик, хөгжүүлэлтэд гол анхаарлаа төвлөрүүлэх, маш хурдтайгаар програм хангамжийг хөгжүүлэх боломж олгодог nodejs фреймворк юм. Харин Prisma ашигласнаар маш хурдтайгаар өгөгдлийн санг бий болгох, query бичихэд бага цагийг харин логикдоо их цагийг зарцуулах боломжийг олгож байгаа нь таалагдлаа. Харин хэрэглэгчийн харьцах хэсэгт ашиглагдах nextjs нь component based react фреймворк бөгөөд нэг бүрэлдэхүүн хэсгийг дахин дахин ашиглах боломжийг олгодог учир маш хурдан хэрэглэгчийн харилцах хэсгийн веб аппликэйшн-д зориулсан технологи юм. Postgre нь маш олон боломжтой өгөгдөл хадгалах, удардах систем юм. Дээрх технологиуд нь Дипломын бүртгэлийн ажлын системийг хөгжүүлэхэд тохирсон технологиуд гэж үзлээ.

⁷Multiversion Concurrency Control

2. СИСТЕМИЙН ШИНЖИЛГЭЭ

Энэ бүлэгт өмнөх бүлгээс олж авсан мэдээллийн дагуу хэрэглэгчийн шаардлага буюу problem domain-с solution domain буюу системийн шаардлагыг боловсруулан гаргана.

2.1 Хэрэглэгчийн шаардлага

Энэ хэсэгт системийн хэрэглэгчдийн илрүүлэх тэдгээрийн хэрэгцээ шаардлагыг тодорхойлно.

Системийн хэрэглэгчид

- Оюутан - дипломын судалгааны ажил хайж буй судлаач эсвэл оюутан
- Супер админ - Системийн админуудыг удирдан зохион байгуулагч
- Админ - Дипломын ажил оруулах хариуцсан хүн буюу сургууль

Оюутны шаардлага

1. Дипломын ажил хайж болдог байх.

Супер админы шаардлага

- Супер адман эрхээрээ нэвтрэх
- Сургууль нэмж админы эрх олгох
- Үүсгэсэн сургуулийн админ эрхийг хасах

Админы шаардлага

- Бүртгүүлсэн эрхээрээ нэвтрэх
- Нууц үгээ сольж болдог байх

- Тэнхим үүсгэж болдог байх
- Өөрийн сургуулийг хариуцсан админ нэмж чаддаг байх
- Үүсгэсэн тэнхим дээр багш нэмэх
- Үүсгэсэн тэнхим болон багшийн нэр дээр дипломын ажил бүртгэх, өөрчлөх, устгах
- Бүртгэсэн дипломын ажлуудыг шүүж хардаг байх

2.2 Системийн шаардлага

Доорх хэсэгт систем юу хийж чаддаг байх ёстой вэ, юу хийж чаддаг байх хэрэгтэй вэ гэдэг асуултын дагуу шаардлагыг боловсруулна.

Функциональ шаардлага

/ФШ10/ Систем рүү дипломын судалгааны ажил хайж үзэхээс өөр хэрэглэгчид системрүү нэвтэрч орох боломжтой байх.

/ФШ20/ Систем нь дипломын ажлыг хайж онлайнаар үзэх боломжийг олгодог байх.

/ФШ30/ Систем нь системийн супер админ нь сургууль бүртгэж тухайн сургуулийн админ эрхийг олгодог байх боломжийг олгох ёстой.

/ФШ40/ Систем нь сургуулийг хариуцсан админ тэнхим үүсгэх, устгах засварлах боломжийг олгодог байх.

/ФШ50/ Систем нь сургуулийн админ дипломын ажил бүртгэх, устгах засварлах боломжийг олгодог байх.

/ФШ60/ Систем нь сургуулийн админ сургуулийн багш бүртгэх, устгах, засварлах боломжийг олгодог байх.

/ФШ70/ Систем нь админыг өөрийн нууц үгээ өөрчлө боломжоор хангадаг байх.

/ФШ80/ Хэрэглэгчид дипломын судалгааны ажлыг сэдэв, удирдагч, гүйцэтгэгчээр хайж болдог байх.

/ФШ90/ Хэрэглэгчид дипломын судалгааны ажлыг татаж авч болдог байх.

/ФШ100/ Систем нь хэрэглэгчид өөрийн санал хүсэлтээ системрүү илгээх боломжоор хангадаг байх.

Функциональ бус шаардлага

Availability - Хүртээмжтэй байдал

/ФБШ10/ Систем нь өдрийн аль ч цагт орсон хэрэглэгчид шаардлагатай үйлчилгээг үзүүлдэг байна.

/ФБШ20/ Сард 5 цагаас дээшгүй хугацаагаар системийн доголдол үүсч болно.

/ФБШ30/ Хэрэглэгчдийн хандалт серверийн хүчин чадлыг давбал сербериg scale-дэх боломжтой байх

Performance - Гүйцэтгэл

/ФБШ40/ Хэрэглэгчдийн хүсэлтэд хариу өгөх хугацаа 6 секундээс доошгүй байна.

/ФБШ50/ Файл системруу байршуулах хугацаа 10 секундээс доошгүй байна.

/ФБШ60/ Системд болж буй үйл явцыг байнга хэрэглэгчид мэдээллэх. Үүнд амжилттай болсон, амжилтгүй гэх мэт төлвүүдийг хэрэглэгчдэд мэдэгдэхийг хэлнэ.

Flexibility - Уян хатан байдал

/ФБШ70/ Системийг веб броузер ашиглан гар утас, компьютер, планшетаар хандаж үйлчилгээ авах боломжтой байна.

/ФБШ80/ Системийн модулиудыг өргөтгөж болохуйц байна.

Security - Аюулгүй байдал

/ФБШ90/ Админ хуудасруу зөвхөн системийн админ хандаж мэдээлэл бүртгэх, өөрчлөх боломжтой байна.

/ФБШ100/ Аюулгүй байдлын үүднээс админ системрүү нэвтрээд 1 цаг болсны дараа системээс автоматаар гардаг байна.

/ФБШ110/ Админы бүртгэлгүй хэрэглэгчдийн системрүү нэвтрэхийг хориглох

Maintainability - Засварлах боломжит байдал

/ФБШ120/ Системийн архитектур, кодонд шинэ модуль нэмт өргөтгөх боломжтой байх.

/ФБШ130/ Системийн бүрэлдэхүүн хэсгүүдийг дахин ашиглаж болохуйц байх.

/ФБШ140/ Ямар ч хөгжүүлэгч системийн хөгжүүлэлтийг дааж авсан ч ойлгогдохуйц байх.

Portability - Зөөвөрлөгдөж болохуйц байдал

/ФБШ150/ Систем нь ямар ч броузер дээр ажилладаг байх

/ФБШ170/ Систем нь аль ч орчинд бусад системүүдтэй зэрэгцэн ажиллаж чаддаг байх

Testability - Тестлеж болохуйц байдал

/ФБШ160/ Системд нэгжийн тест, бүрэлдэхүүн хэсгийн тест, системийн тестийг хийж болохуйц байх.

2.3 Бүлгийн дүгнэлт

Энэхүү бүлэгт хэрэглэгчийн талаас хүсч буй шаардлагуудыг гарган системийн талаас хэрхэн хэрэгжүүлж, шийдэж болох вэ гэдгийг судалж шаардлагуудыг боловсруулсан бөгөөд хэрэглэгчийн талаас системийг харж мөн систем талаас нь чадварыг нь судалсанаар систем юу хийж чадах вэ, хэрэглэгч юу хүсч байна вэ гэдэг 2 өнцгийг хэрхэн нийлүүлж нэгэн бүхэл болгох дадлага туршлагыг хуримтлууллаа.

3. СИСТЕМИЙН ЗОХИОМЖ

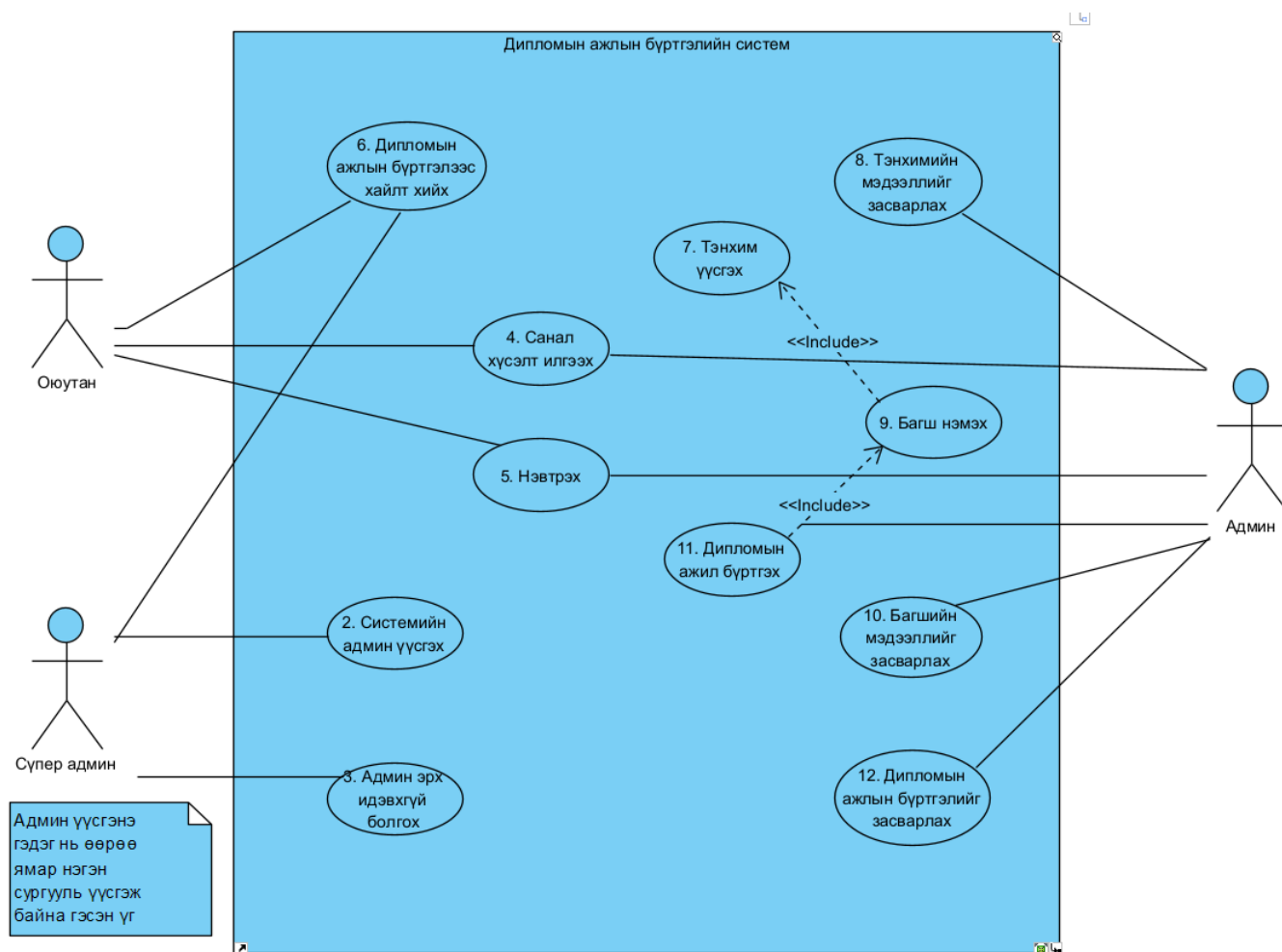
Тухайн бүлэгт системийн динамик зан төлвийг судалж хэрэглэгч болон системд тулгарч болох асуудлыг олж илрүүлэх юм. Энэ бүлэгт байгаа бүх диаграмыг Visual Paradigm¹ хэрэглүүрийг ашиглан зурсан.

3.1 Статик загвар

Системийн ажлын явц

Дипломын бүртгэлийн ажлын явцыг дараах байдлаар тодорхойлов. Системийн оролцогч талуудыг диплом ажлын дэлгэрэнгүйг харах гэж буй хэрэглэгч буюу оюутан. Нөгөө талаас дипломын ажил оруулах сургууль буюу админ, нийт админуудыг хянах, нэмэх супер адман гэж тодорхойлсон. Зураг 3.1. Хүснэгт 3.6, 3.1, 3.2, 3.3, 3.4, 3.5, 3.7, 3.8-т ажлын явцын өргөтгөлийг оруулж өгөв.

¹Дэлгэрэнгүй үзэх <https://www.visual-paradigm.com/>



Зураг 3.1: Ажлын явцын диаграм

Table 3.1: Ажлын явц 2: Системийн админ үүсгэх

Ажлын явц	Системийн админ үүсгэх
Зорилго	Системийн админ үүсгэснээр тухайн админаас хийгдэх үйлдлүүдийг хийж чаддаг болно
Ангилал	Анхдагч
Тоглогч	Админ
Төгсгөл нөхцөл амжилттай	Системийн админруу хандаж чаддаг болсон байна.
Төгсгөл нөхцөл амжилтгүй	Бүртгэл амжилтгүй болж системрүү хандаж чадахгүй
Өдөөгч	Дипломын ажил бүртгэх.
Тайлбар	Системийн админ үүсгэнэ гэдэг нь ямар нэгэн сургууль давхар үүсч байна гэсэн үг юм.
Өргөтгөл	

Table 3.2: Ажлын явц 3: Админ эрх идэвхгүй болгох

Ажлын явц	Админ эрх идэвхгүй болгох
Зорилго	Админий эрхийг хүчингүй болгох
Ангилал	Анхдагч
Тоглогч	Сүпер админ
Төгсгөл нөхцөл амжилттай	Админ админ хуудасруу орж чадахгүй.
Төгсгөл нөхцөл амжилтгүй	Админы эрхийг хүчингүй болгож чадахгүй
Өдөөгч	Админы эрхийг хасах шаардлага гарах.
Тайлбар	Админуудын эрхийг зөвхөн сүпер админ өөрчилдөг.
Өргөтгөл	

Table 3.3: Ажлын явц 4: Санал хүсэлт илгээх

Ажлын явц	Санал хүсэлт илгээх
Зорилго	Системийн хэрэглэгчид системд нэмэлтээр байж болох зүйлүүд эсвэл доголдол алдаа, нэмэлт мэдээлэл өгөх зорилготойгоор санар хүсэлт илгээнэ.
Ангилал	Анхдагч
Тоглогч	Админ, Хэрэглэгчид
Төгсгөл нөхцөл амжилттай	Системд хэрэглэгчдийн өгсөн санал хүсэлтүүд амжилттай очих.
Төгсгөл нөхцөл амжилтгүй	Санал хүсэлтээ илгээж чадахгүй байх.
Өдөөгч	Хэрэглэгчдэд асуудал тулгарсан эсвэл мэдээллэх шаардлагатай зүйл гарах үед.
Тайлбар	<ol style="list-style-type: none"> 1. Санал хүсэлт илгээхийн тулд заавал системрүү нэвтрэх шаадлагагүй 2. Санал хүсэлт илгээх форм бөглөж илгээх товч дарна.
Өргөтгөл	

Table 3.4: Ажлын явц 5: Нэвтрэх

Ажлын явц	Нэвтрэх
Зорилго	Админууд систем рүү нэвтрэн өгөгдөл нэмэх, засварлах үйлдлүүдийг хийнэ.
Ангилал	Анхдагч
Тоглогч	Админ,Сүпер админ
Төгсгөл нөхцөл амжилттай	Админ хуудасруу амжилттай нэвтрэнэ
Төгсгөл нөхцөл амжилтгүй	Системрүү нэвтрэрч чадахгүй байх
Өдөөгч	Админ системрүү нэвтрэхийг хүссэн.
Тайлбар	<ol style="list-style-type: none"> 1. Нэвтрэх хуудсыг дуудаж нэвтрэх нэр нууц үгээ оруулна. 2. Хэрвээ нүүц үг буруу бол нэвтрэхгүй, эсвэл мартсан бол нууц үгээ солино.
Өргөтгөл	Нүүц үгээ мартсан бол нууц үгээ сэргээнэ.

Table 3.5: Ажлын явц 6: Дипломын ажил бүртгэх

Ажлын явц	Дипломын ажил бүртгэнэ.
Зорилго	Их сургууль төгсөгчдийн дипломын ажлыг системд бүртгэнэ.
Ангилал	Анхдагч
Тоглогч	Админ
Төгсгөл нөхцөл амжилттай	Дипломын ажлыг амжилттай бүртгэнэ.
Төгсгөл нөхцөл амжилтгүй	Дипломын ажлыг бүртгэж чадахгүй.
Өдөөгч	Дипломын ажил бүртгэх шаардлага гарна.
Тайлбар	<ol style="list-style-type: none"> 1. Админ нь өөрийн бүртгэлээрээ системийн админ хуудасруу нэвтрэн орно. 2. Админ нь дипломын ажил бүртгэхийн тулд эхлээд тэнхим үүсгэнэ. Дараа нь тухайн тэнхимийн багшийг үүсгэнэ. Үүний дараа дипломын ажлыг бүртгэж болно.
Өргөтгөл	

Table 3.6: Ажлын явц 12: Дипломын ажлын бүртгэлийг засварлах

Ажлын явц	Дипломын ажлын бүртгэлийн засварлах
Зорилго	Бүртгэсэн дипломын ажлын өгөгдлийг засварлана.
Ангилал	Анхдагч
Тоглогч	Админ
Төгсгөл нөхцөл амжилттай	Бүртгэлтэй дипломын ажил засварлана.
Төгсгөл нөхцөл амжилтгүй	Дипломын ажил засварлаж чадахгүй байх.
Өдөөгч	Бүртгэлтэй дипломын ажлын өгөгдлийг өөрчлөх хэрэгцээ гарах.
Тайлбар	<ol style="list-style-type: none"> 1. Нэвтрэх хуудсыг дуудаж админ нэвтрэх нэр нууц үгээ оруулна. 2. Админ хуудасруу нэвтэрсний дараа бүртгэлтэй дипломын ажлуудаас хайлт хийн засварлана.
Өргөтгөл	

Table 3.7: Ажлын явц 7: Тэнхим үүсгэх

Ажлын явц	Тэнхим үүсгэх
Зорилго	Сургуулийн тэнхимүүдийг үүсгэнэ.
Ангилал	Анхдагч
Тоглогч	Админ
Төгсгөл нөхцөл амжилттай	Тэнхим амжилттай үүсгэнэ.
Төгсгөл нөхцөл амжилтгүй	Тэнхим үүсгэж чадахгүй байх
Өдөөгч	Шинээр тэнхим бүртгэхийг хүсвэл.
Тайлбар	<ol style="list-style-type: none"> 1. Нэвтрэх хуудсыг дуудаж нэвтрэх нэр нууц үгээ оруулна. 2. Хэрвээ нүүц үг буруу бол нэвтрэхгүй, эсвэл мартсан бол нууц үгээ солино.
Өргөтгөл	Нүүц үгээ мартсан бол нууц үгээ сэргээнэ.

Table 3.8: Ажлын явц 9: Багш нэмэх

Ажлын явц	Багш нэмэх
Зорилго	Тэнхимп харьяалалтай багш нарыг бүртгэнэ
Ангилал	Анхдагч
Тоглогч	Админ
Төгсгөл нөхцөл амжилттай	Багш амжилттай системрүү нэмнэ.
Төгсгөл нөхцөл амжилтгүй	Багш нэмэх үйлдэл амжилтгүй болох.
Өдөөгч	Багш нэмэхийг хүссэн.
Тайлбар	<ol style="list-style-type: none"> 1. Админ системрүү нэвтрэнэ 2. Багш нэмэх формыг бөглөж илгээх товч дарагдахад багшийг нэмнэ.
Өргөтгөл	

3.1.1 Системийн классын бүтэц зохион байгуулалт

Энэ хэсэгт системийн классуудыг илрүүлж хоорондын харьцааг тодорхойлно. Зураг 3.2-с классын диаграмыг харна уу. Системийн классууд:

Админ

Админ класс нь системийн админы овог, нэр, и-мэйл хаяг, утасны дугаар гэх мэт мэдээллүүд болон аргуудыг тодорхойлж буй класс юм. Админ классаас SimpleAdmin, SuperAdmin гэх 2 класс удамших бөгөөд SimpleAdmin класс нь тухайн нэгэн их сургуулийн дипломын ажлыг удирдан зохион байгуулах үүрэгтэй бол SuperAdmin нь тухайн системийн нийт админуудыг мэдээллийг удирдан зохион байгуулах үүрэгтэй юм.

Их сургууль

Их сургууль класс их сургуулийн мэдээлэл болох нэр, хаяг, дугаар нь, дэлгэрэнгүй мэдээллийг агуулдаг. Их сургууль класс нь SimpleAdmin класстай 1 1 харьцаатай aggregation холбоосоор холбогдоно. Их сургууль нэг админтай, админ нэг нэг их сургуультай байна.

Тэнхим

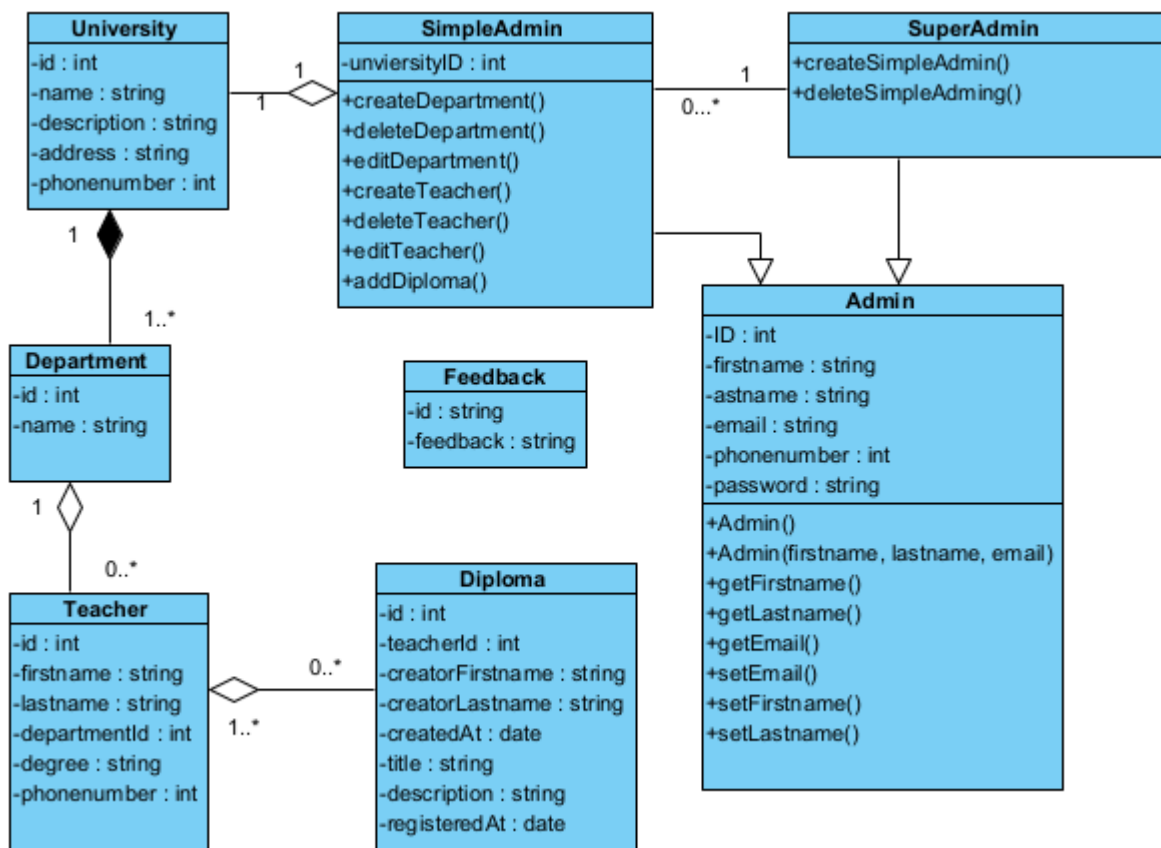
Тэнхим класс нь тэнхимийн нэр, харьяа сургуулийн id-г агуулах бөгөөд их сургууль класстай 1 1..* харьцаатай composition холбоосоор холбогдоно. Тэнхим нь нэг сургуульд харьяалагдах бөгөөд сургууль нь олон тэнхимтэй байж болно.

Багш

Багш класс нь багшийн овог, нэр, утасны дугаар, харьяа тэнхимийн id-г хадгалах класс бөгөөд тэнхим класстай 1 0..* харьцаатай aggregation холбоосоор холбогдоно. Багш нь нэг л тэнхимд харьяалагдах бөгөөд тэнхим нь олон багштай байж болно.

Диплом

Диплом класс нь судалгааны ажлын нэр, тайлбар, гүйцэтгэгийн овог нэр, удирдагч багшийн id дугаар диплом хамгаалсан хугацаа, диплом бүртгэсэн хугацаа зэрэг мэдээллийг агуулах класс байна. Диплом нь багш 1 0..* харьцаатай aggregation холбоосоор холбогдох бөгөөд нэг багш 0 юмуу түүнээс дээш диплом удирдсан байж болно. Харин диплом нь нэг удирдагчтай байна.

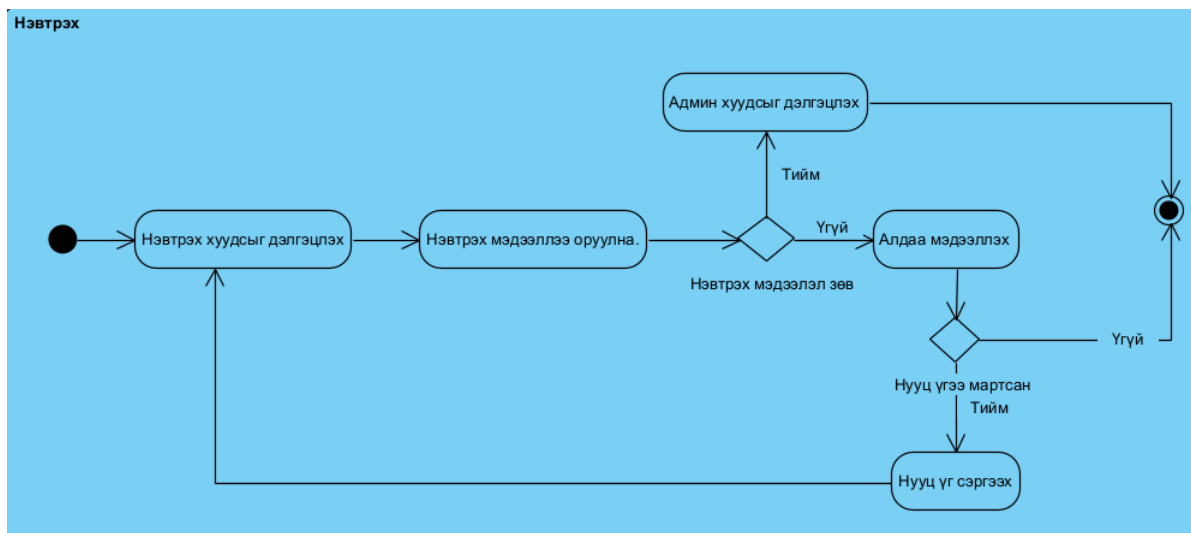


Зураг 3.2: Системийн классын диаграм

3.2 Динамик загвар

Системийн админ системрүү нэвтрэх үеийн үйл ажиллагааны дараалал. Зураг 3.3-с үйл ажиллагааны дарааллыг диаграмыг үзнэ үү.

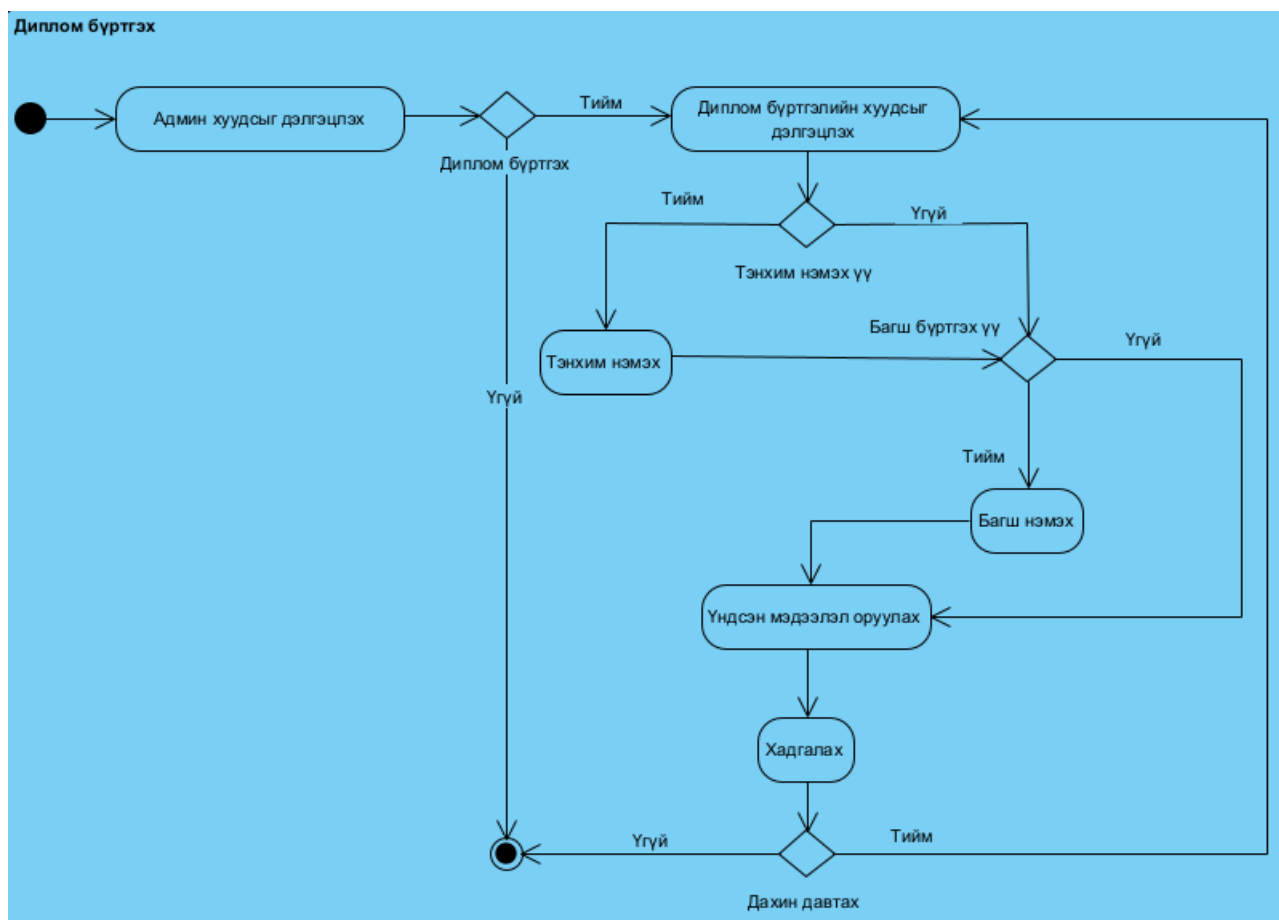
- Системийн админ /admin url-руу хандсанаар нэвтрэх хуудас гарч ирнэ. Тухайн хуудас нь ямар нэгэн бүртгүүлэх хэсэггүй бөгөөд зөвхөн нууц үг сэргээх товч мэдээлэл бөглөх форм агуулна.
- Админ өөрийн нэвтрэх мэдээллийг бөглөснөөр систем мэдээлэл үнэн эсэхийг шалгаж хариу үр дүнг мэдээллэнэ.
- Хэрвээ тухайн хэрэглэгч нууц үгээ мартсан бол нууц үг сэргээх товчийг даран нууц үгээ сэргээнэ.
- Бүх үйлдэл амжилттай болсон бол админ админ хуудасруу нэвтэрнэ.



Зураг 3.3: Админ системрүү нэвтрэх үйл ажиллагааны диаграм

Диплом бүртгэлийн үйл ажиллагааны дараалал. Зураг 3.4-с үйл ажиллагааны дарааллын диаграммыг үзнэ үү.

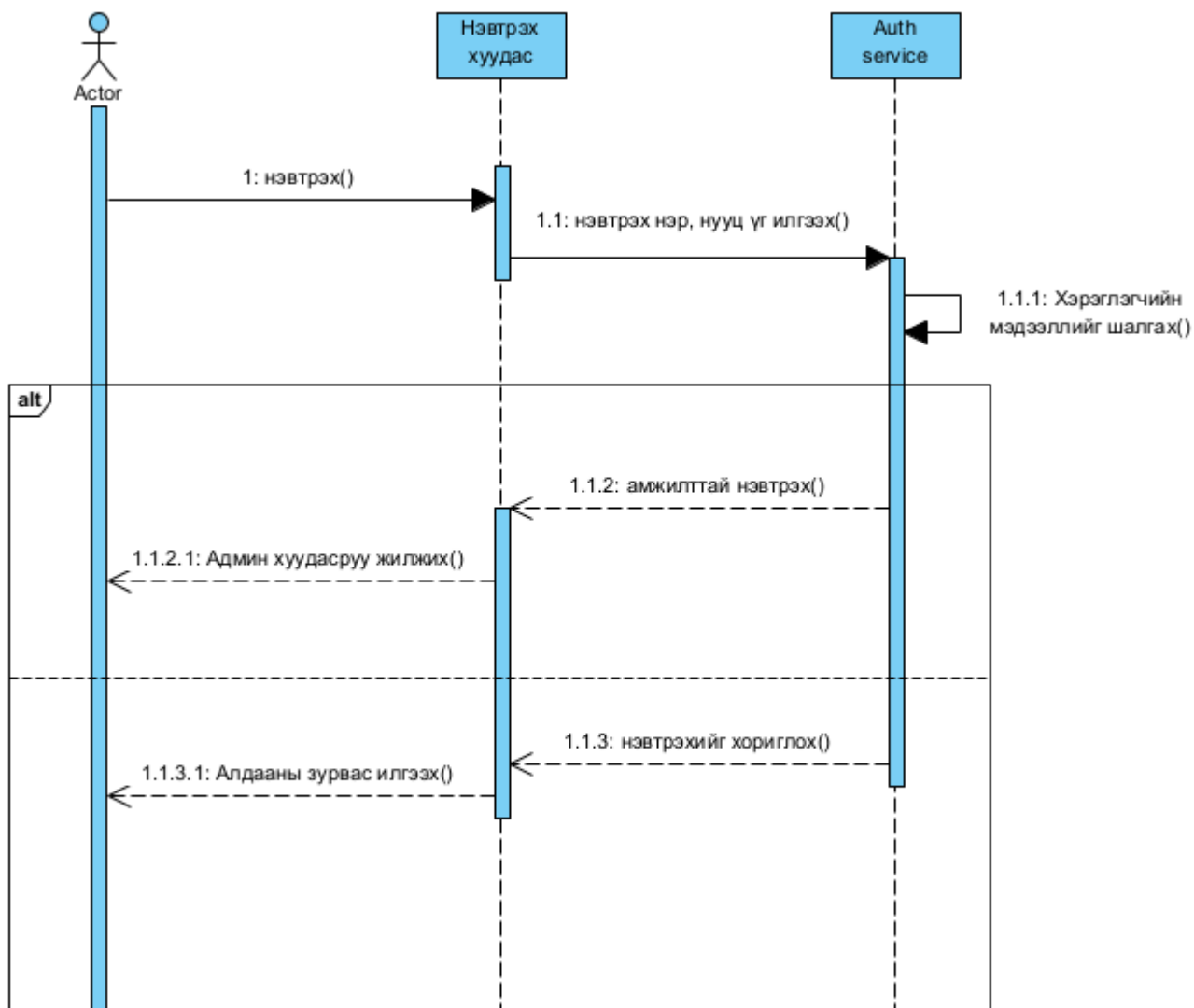
- Админ нэвтрэн орсоны дараа админ цэснээс диплом бүртгэх цэсийг сонгон тухайн хуудасруу шилжинэ.
- Диплом бүртгэхэд шаардлагатай үндсэн мэдээллүүдийг оруулах форм бөглөх ба харьяалагдах тэнхим, болон багшийг сонгоно.
- Хэрвээ тухайн тэнхим болон багш системд бүртгэлгүй бол нэмж өгнө.
- Бүх мэдээллийг оруулан хадгалах товч дарах ба систем хариу үр дүнг харуулна.



Зураг 3.4: Диплом бүртгэлийн үйл ажиллагааны диаграм

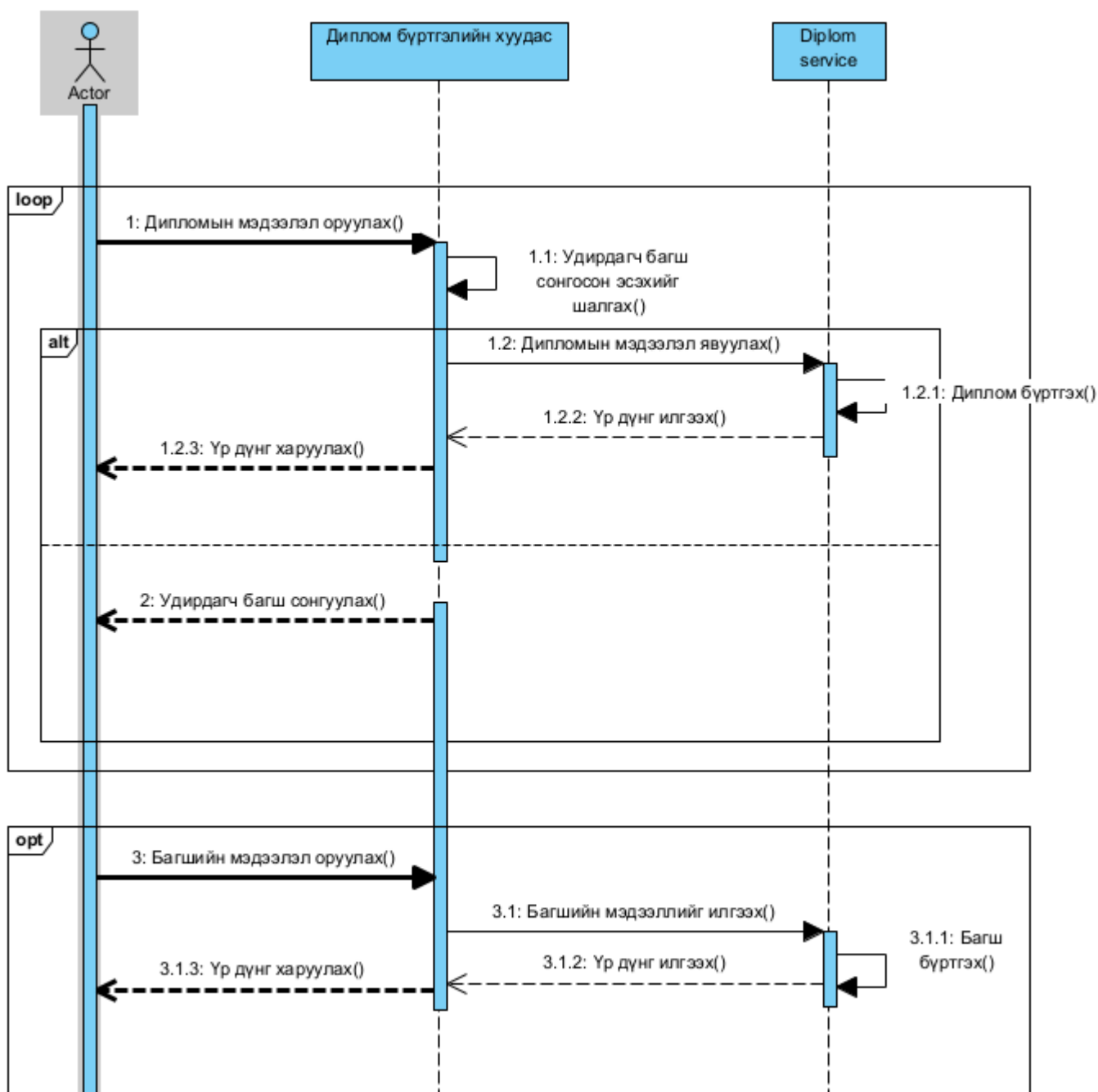
Системийн бүрэлдэхүүн хэсгүүд хоорондоо хэрхэн холбогдож ажиллаж байгааг үйл ажиллагааны диаграмаар дүрслэн харуулав.

- Админ системрүү нэвтрэх үйл ажиллагааны дарааллыг зураг 3.5-т харуулав.



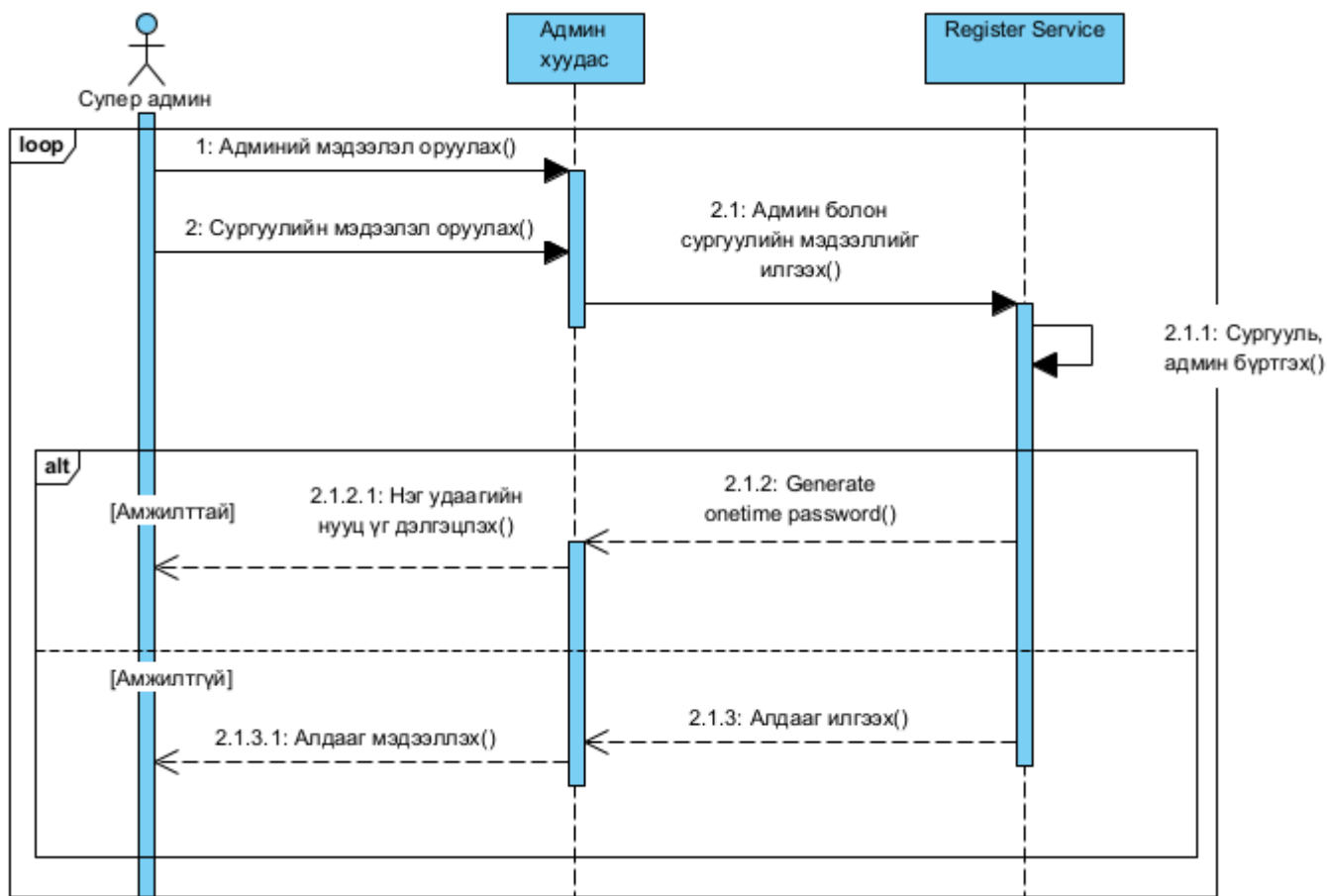
Зураг 3.5: Админ системрүү нэвтрэх дарааллын диаграм

- Админ дипломын ажил бүртгэх үйл ажиллагааны дарааллыг зураг 3.6-т харуулав.



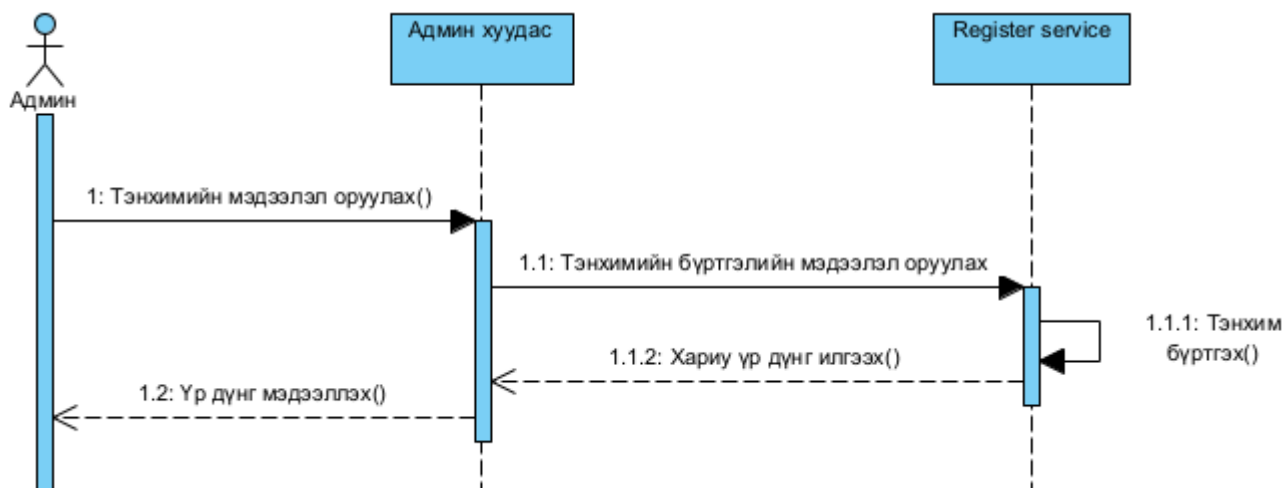
Зураг 3.6: Дипломын ажил бүртгэх дарааллын диаграм

- Супер админ админ нэмэх үйл ажиллагааны дарааллыг зураг 3.7-т харуулав.



Зураг 3.7: Админ үүсгэх дарааллын диаграм

- Админ тэнхим бүртгэх үйл ажиллагааны дарааллыг зураг 3.8-т үзүүлэв.



Зураг 3.8: Админ тэнхим бүртгэх дарааллын диаграм

3.3 Өгөгдлийн сангийн зохиомж

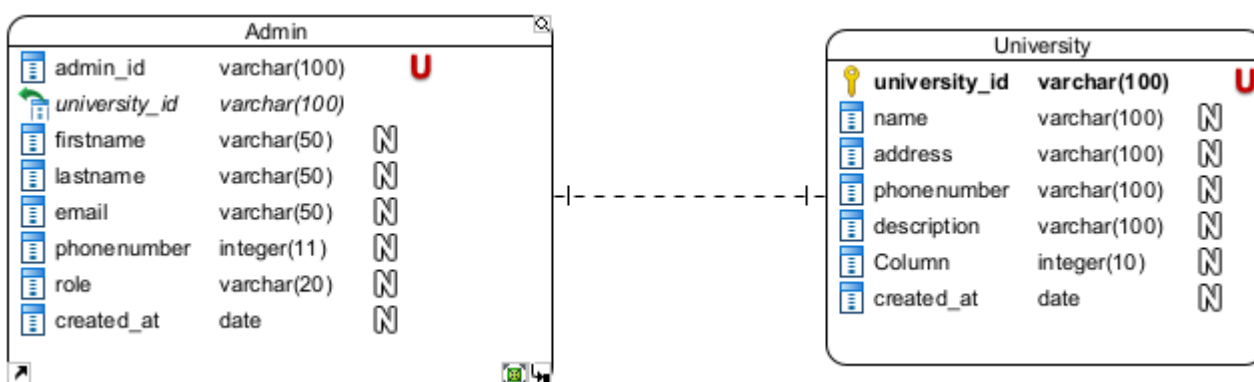
Системийн өгөгдлийн сангийн зохиомжийг **Entity Relationship Diagram**-р дүрслэн үзүүлэв. Зураг 3.13-с ERD диаграмыг үзнэ үү.

Системийн Strong Entity

- Админ - Admin
- Их сургууль - University
- Тэнхим - Department
- Багш - Teacher
- Диплом - Diploma

Холбоосын төрлийг тогтоох**Админ - Их сургууль**

Admin хүснэгт нь **University** хүснэгттэй 1:1² холбоосоор холбогдоно. Энэ нь нэг админ нь нэг л сургуулийн хариуцан ажиллана. Харин сургууль нь нэг л админтай байж болно гэсэн үг юм. Үүнд их сургуулийг устахаас сэргийлж University id нь primary key болж буй учир админ утсан ч University хүснэгтэд нөлөөлж чадахгүй юм. Хүснэгтийн холбоосыг зураг 3.9-с үзнэ үү.



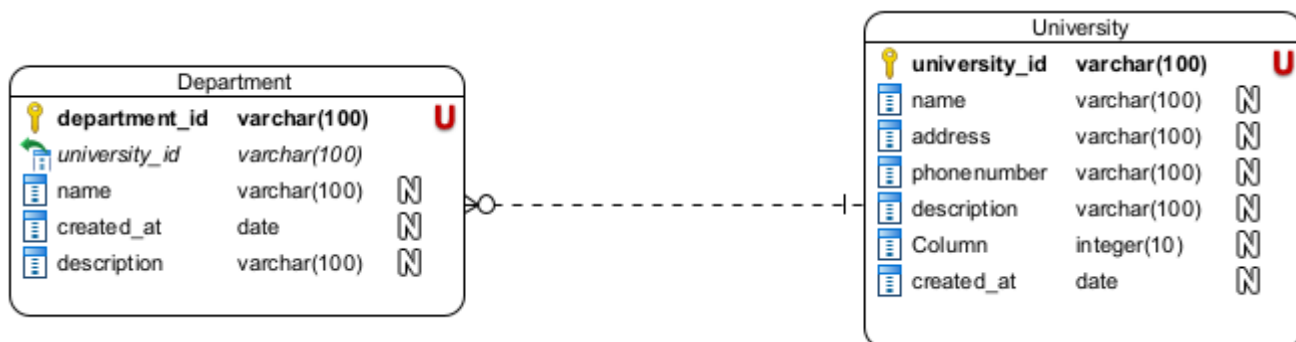
Зураг 3.9: Админ болон их сургууль хүснэгтийн холбоос

Их сургууль - Тэнхим

University хүснэгт нь **Department** хүснэгттэй 1:0³ холбоосоор холбогдох юм. Их сургууль нь олон тэнхимтэй байж болно. Харин тэнхим нь олон их сургуульд харьяалагдаж болохгүй. Хүснэгтийг зураг 3.10-с үзнэ үү.

²one-to-one

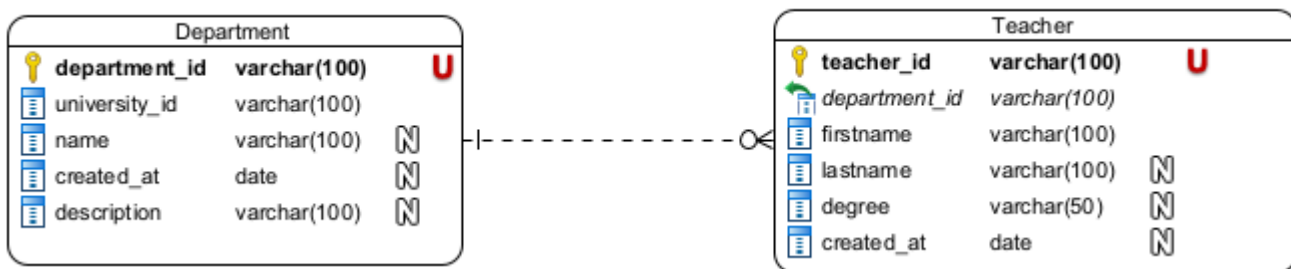
³one-to-many



Зураг 3.10: Их сургууль болон тэнхим хүснэгтийн холбоос

Тэнхим - Багш

Department хүснэгт нь **Teacher** хүснэгттэй 1:0 холбоосоор холбогдоно. Тэнхим нь олон багштай байж болно. Багш нэг л тэнхимд харьяалагдана. Дэлгэрэнгүй зургийг 3.11-с үзнэ үү.

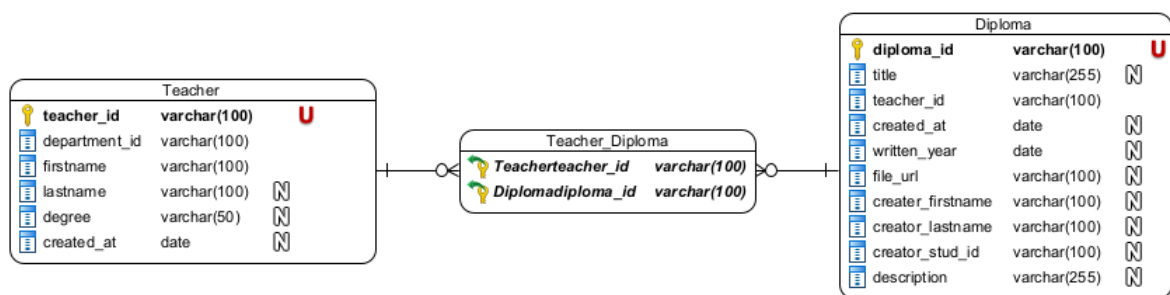


Зураг 3.11: Тэнхим болон багш хүснэгтийн холбоос

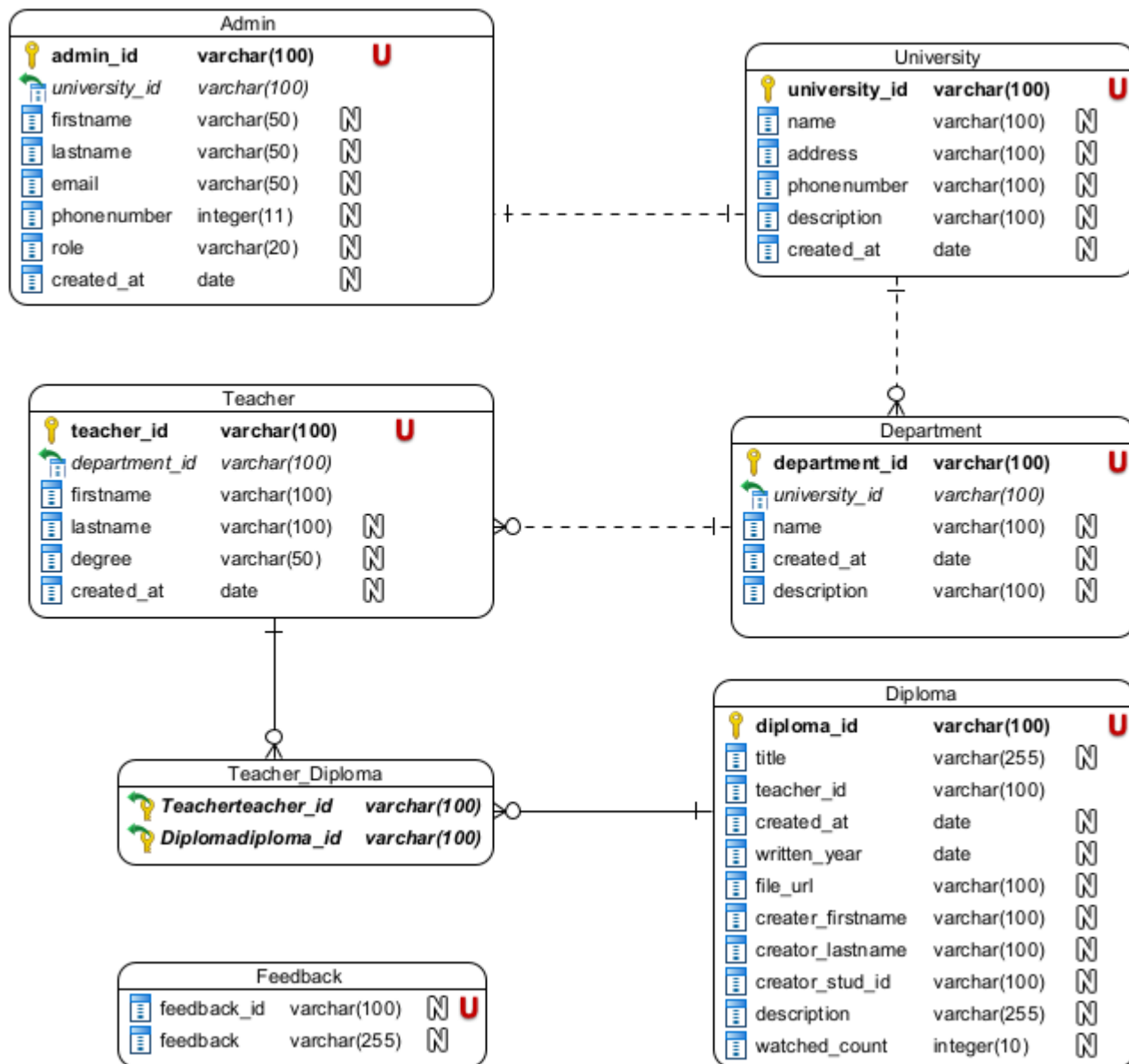
Багш - Диплом

Teacher хүснэгт нь **Diploma** хүснэгттэй 0:0⁴ холбоосоор холбогдсон учир дундын **Teacher _Diploma** хүснэгтийг үүсгэсэн. Өөрөөр хэлбэл нэг диплом олон удирдагч багштай байж болно. Багш нь олон диплом удирдсан байж болно. Хүснэгтийн холбоосыг зураг 3.12-с үзнэ үү.

⁴many-to-many



Зураг 3.12: Багш болон Диплом хүснэгтийн холбоос



Зураг 3.13: Өгөгдлийн сангийн диаграм

Table 3.9: Admin хүснэгт

№	Альтернатив нэр	Түлхүүр өгөгдөл	Өгөгдлийн төрөл	Хоосон утга	Тайлбар
1	adminId	PK	varchar	not null	Админд дахин давтагдашгүй утгыг автоматаар олгоно.
2	universityId	FK	varchar	not null	Их сургуулийн дугаар
3	firstname		varchar	notnull	Админы нэр
4	lastname		varchar	not null	Админы овог
5	email		varchar	not null	e-mail хаяг
6	phonenumner	PK	int	not null	Утасны дугаар дахин давтагдашгүй байна.
7	role		varchar	not null	систем дэх хандалтын эрхийг илэрхийлнэ.
8	createdAt		date	not null	Автоматаар системд үүссэн цагийг бүртгэнэ.

Table 3.10: University хүснэгт

№	Альтернатив нэр	Түлхүүр өгөгдөл	Өгөгдлийн төрөл	Хоосон утга	Тайлбар
1	universityId	PK	varchar	not null	Их сургуульд дахин давтагдашгүй утгыг автоматаар олгоно.
2	name		varchar	not null	Их сургуулийн нэр
3	address		varchar	nonnull	Их сургуулийн хаяг
4	phonenumner		int	null	Их сургуулийн утасны дугаар
5	description		varchar	null	Тайлбар
6	createdAt		date	not null	Үүсгэсэн хугацааг автоматаар хадгална.

Table 3.11: Department хүснэгт

№	Альтернатив нэр	Түлхүүр өгөгдөл	Өгөгдлийн төрөл	Хоосон утга	Тайлбар
1	departmentId	PK	varchar	not null	Тэнхимд дахин давтагдашгүй утгыг автоматаар олгоно.
2	universityId	FK	varchar	not null	Их сургуулийн дугаар
3	name		varchar	notnull	Тэнхимийн нэр
4	createdAt		date	not null	Тэнхим үүссэн хугацааг автоматаар хадгална.
5	description		varchar	null	Тайлбар

Table 3.12: Teacher хүснэгт

№	Альтернатив нэр	Түлхүүр өгөгдөл	Өгөгдлийн төрөл	Хоосон утга	Тайлбар
1	teacherId	PK	varchar	not null	Багшид дахин давтагдашгүй утгыг автоматаар олгоно.
2	departmentId	FK	varchar	not null	Тэнхимийн дугаар
3	firstname		varchar	not null	Багшийн нэр
4	lastname		varchar	not null	Багшийн овог
5	degree		varchar	null	Зэрэг
6	createdAt		date	not null	Үүсгэсэн хугацааг автоматаар хадгална.
7	phonenumner		int	null	Утасны дугаар

Table 3.13: Diploma хүснэгт

№	Альтернатив нэр	Түлхүүр өгөгдөл	Өгөгдлийн төрөл	Хоосон утга	Тайлбар
1	diplomaId	PK	varchar	not null	Дипломын дахин давтагдашгүй утгыг автоматаар олгоно.
2	title		varchar	not null	Дипломын сэдэв
3	creatorFname		varchar	null	Гүйцэтгэгчийн нэр
4	creatorLname		varchar	null	Гүйцэтгэгчийн овог
5	creatorStudId		varchar	null	Гүйцэтгэгчийн дугаар
6	description		varchar	null	Сэдвийн тайлбар
7	createdAt		date	not null	Үүссэн хугацааг автоматаар хадгална.
8	writtenYear		date	not null	Диплом хамгаалсан жил
9	fileUrl		varchar	not null	Файлын зам
10	watchedCount		int	not null	Нийт үзэгдсэн тоо.

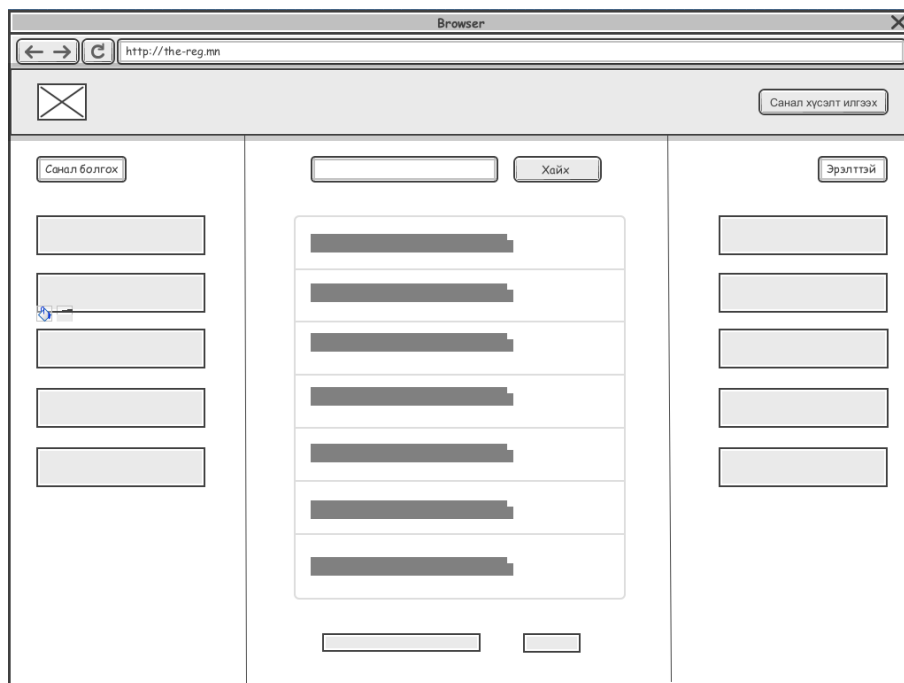
Table 3.14: Teacher-Diploma хүснэгт

№	Альтернатив нэр	Түлхүүр өгөгдөл	Өгөгдлийн төрөл	Хоосон утга	Тайлбар
1	diplomaId	FK	varchar	not null	Дипломын ажлын дугаар
2	teacherId	FK	varchar	not null	Багшийн дугаар

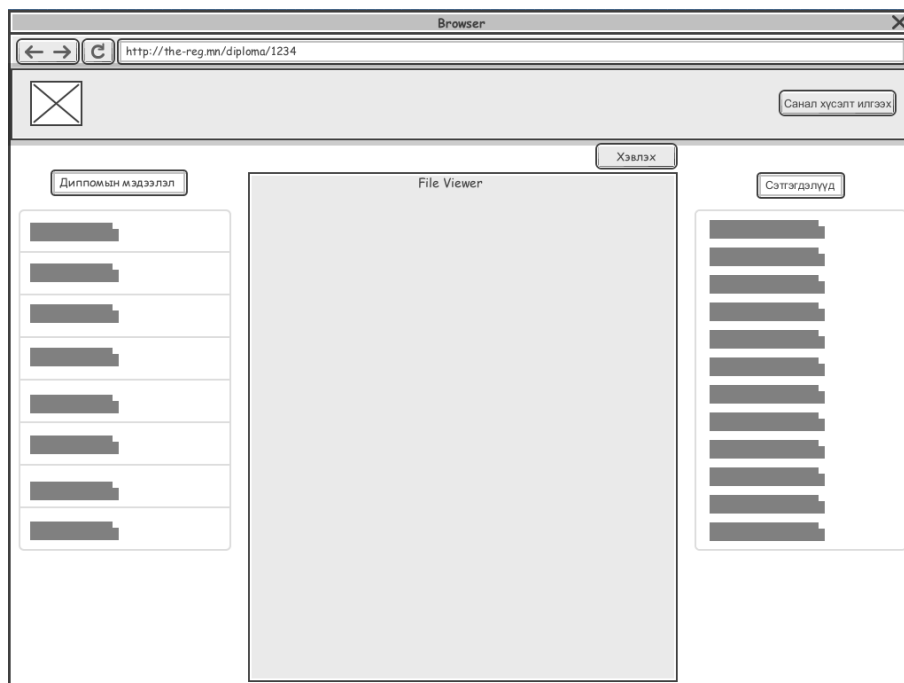
3.4 Хэрэглэгчийн харьцах хэсгийн хар зураг

Хэрэглэгчийн харагдах хэсгийн хар зургийг хэрэглэгчийн шаардлага, системийн зохиомж дээр үндэслэн гаргасан болно.

- Хэрэглэгч системээс диплом хайх, хамгийн их үзэгдсэн дипломуудыг харах, санал болгох диплом харах үндсэн цонх. Зураг 3.14
- Дипломын дэлгэрэнгүй мэдээллийн харах, татаж авах цонх. Зураг 3.15
- Системийн админууд нэвтрэх цонх. Зураг 3.16
- Админ цонх. Зураг 3.17
- Админ цонхноос багш бүртгэх. Зураг 3.18
- Диплом бүртгэх цонх. Зураг 3.19
- Бүртгэлтэй багш, тэнхим, дипломын мэдээллийг харагдах ерөнхий цонх. Зураг 3.20
- Супер админ энгийн админ нэмэх цонх. Зураг 3.21



Зураг 3.14: Үндсэн цонх



Зураг 3.15: Дипломын дэлгэрэнгүй цонх

Browser

← → ↻ http://the-reg.mn/admin

✕

✕ Санал хүсэлт илгээх

✕

Нэвтрэх нэр

Нууц үг

Нэвтрэх

Нууц үгээ мартсан

Зураг 3.16: Нэвтрэх цонх

Browser

← → ↻ http://the-reg.mn

✕

✕ Санал хүсэлт илгээх

✕

Гарах

Зураг 3.17: Админ цонх

Зураг 3.18: Багш бүртгэх цонх

Зураг 3.19: Диплом бүртгэх цонх

Browser

← → ↻ http://the-reg.mn/admin/data

✕

✕ Санал хүсэлт илгээх

Гарах

Зураг 3.20: Өгөгдөл харагдах цонх

Browser

← → ↻ http://the-reg.mn

✕

✕ Санал хүсэлт илгээх

Админ нэмэх

Гарах

Админ мэдээлэл

Овог Нэр

И-мэйл хаяг Утасны дугаар

Сургуулийн мэдээлэл

Сургуулийн нэр Утасны дугаар

Хаяг

Хадгалах

Зураг 3.21: Админ нэмэх цонх

3.5 Бүлгийн дүгнэлт

Дээрх бүлэгт өмнөх бүлэгт тодорхойлосн шаардлагуудыг улам боловсронгуй болгон түүний дагуу системийн статик болон динамик диаграмуудыг зохиомжилсон. Мөн өгөгдлийн сангийн загварчлал болон хэрэглэгчийн харьцах хэсгийн загварыг шаардлага болон системийн статик, динамик загварууд дээр суурьлан гаргалаа.

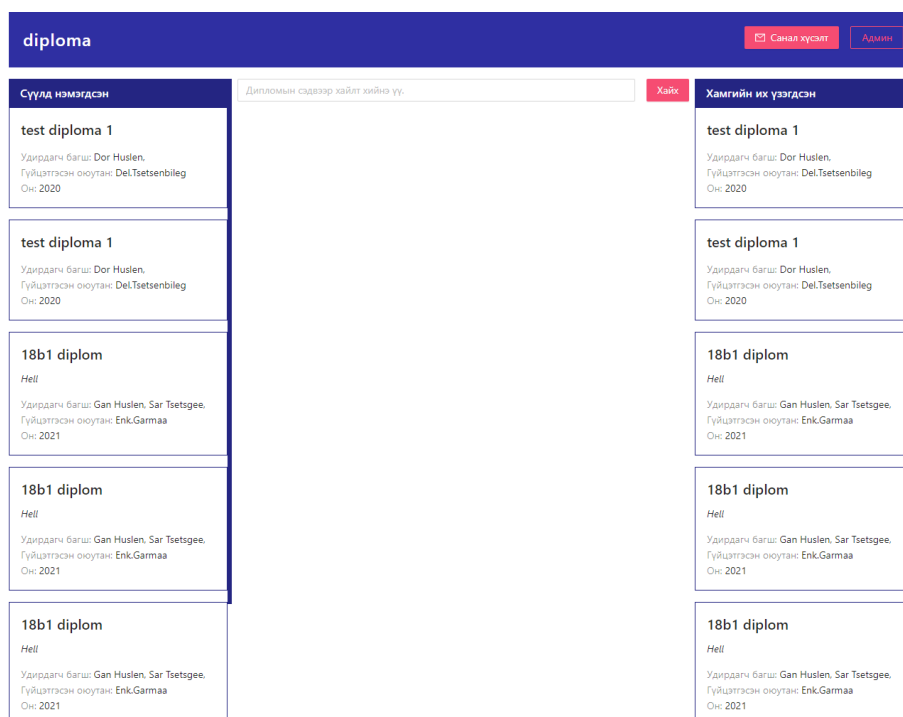
4. ХЭРЭГЖҮҮЛЭЛТ, ҮР ДҮН

Хөгжүүлэлтийг дээрх бүлэг сэдвүүд дээр тодорхойлосон шинжилгээ ба зохиомжийн дагуу хийсэн болно. Хэрэгжүүлэлтийг хавсралт хэсгээс харна уу.

4.1 Хөгжүүлэгдсэн байдал

Үндсэн дэлгэц

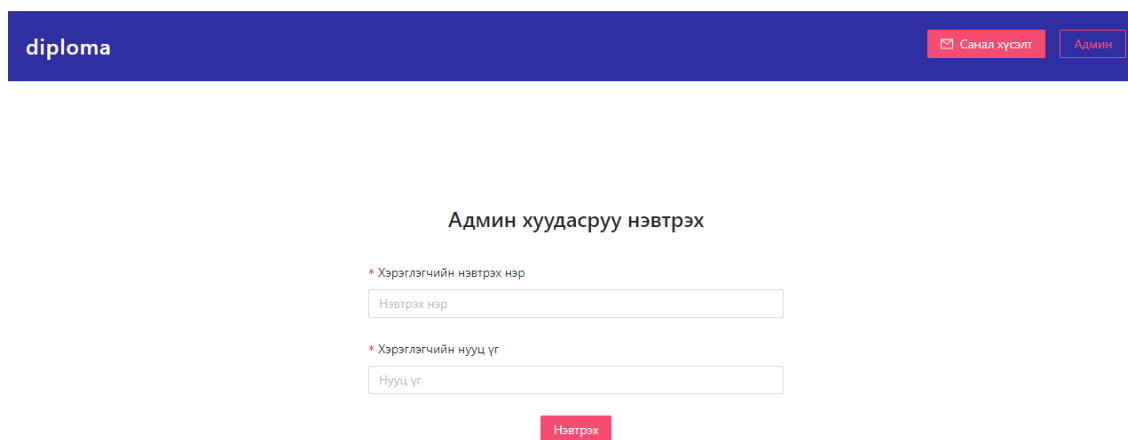
Тухайн цонх хэрэглэгчийн харьцах цонх бөгөөд тухайн цонхноос хэрэглэгч диплом сэдвээр хайх, хамгийн их үзэлттэй дипломуудыг харах боломжтой.



Зураг 4.1: Диплом хайх хэсэг

Нэвтрэх цонх

Системийн админуудын системрүү нэвтрэх цонх.

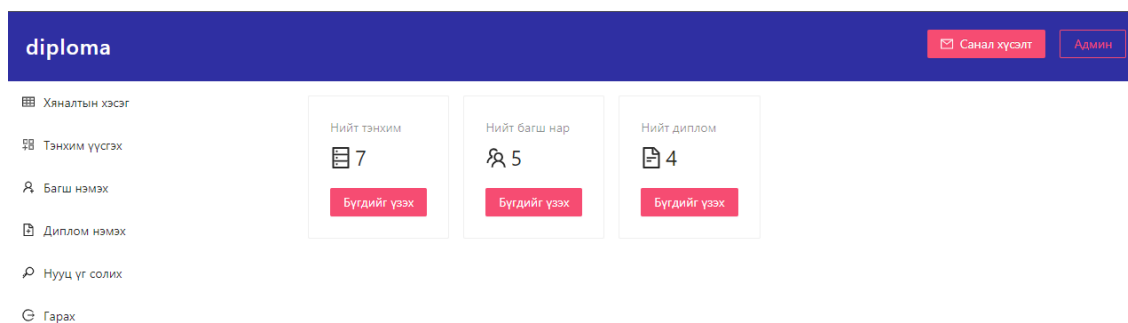


The screenshot shows a web interface for resetting an admin password. At the top, there is a dark blue header with the word 'diploma' on the left and two buttons on the right: 'Санал хүсэлт' (Feedback) and 'Админ' (Admin). Below the header, the title 'Админ хуудасруу нэвтрэх' (Admin login) is centered. The form contains two input fields: the first is labeled 'Хэрэглэгчийн нэвтрэх нэр' (Admin login name) with a placeholder 'Нэвтрэх нэр'; the second is labeled 'Хэрэглэгчийн нууц үг' (Admin password) with a placeholder 'Нууц үг'. A red 'Нэвтрэх' (Login) button is positioned below the second field.

Зураг 4.2: Нэвтрэх цонх

Админ цонх

Админ системрүү нэвтэрсэнийн дараа админд харагдах цонх бөгөөд админ нийт тэнхим, багш , нэмсэн дипломуудыг харах боломжтой бөгөөд тэнхим үүсгэх, багш нэмэх , диплом үүсгэх, нууц үгээ солих боломжтой.



Зураг 4.3: Админ цонх

Багш нэмэх

Тухайн сургуулийн багш нэмэх цонх.

Зураг 4.4: Багш нэмэх цонх

Диплом нэмэх хуудас

Тухайн хуудсыг ашиглан диплом нэмнэ.

diploma

Санал хүсэлт

Админ

Хяналтын хэсэг

Тэнхим үүсгэх

Багш нэмэх

Диплом нэмэх

Нууц үг солих

Гарах

* Дипломын сэдэв

Дипломын сэдэв

* Удирдагч багш сонгох

Удирдсан багш нарыг сонгон...

* Тэнхим ээ сонгоно уу

Тэнхим

* Бичигдсэн он

Он

* Файл сонгох(pdf файл оруулна уу)

Choose File

No file chosen

* Гүйцэтгэгчийн овог

Гүйцэтгэгчийн овог

* Гүйцэтгэгчийн нэр

Гүйцэтгэгчийн нэр

* Гүйцэтгэгчийн оюутны код

Гүйцэтгэгчийн оюутны код

Тайлбар

Тайлбар

Диплом нэмэх

Зураг 4.5: Диплом нэмэх хуудас

Адмын нэмэх болон сургууль нэмэх хуудас

Системрүү супер админ нэвтрэх үед хажуу талын цэсэнд хэрэглэгч нэмэх цэс гарч ирэх бөгөөд тухайн цэсрүү орсоноор админ болон сургууль үүсгэж болно.

Зураг 4.6: Админ нэмэх хуудас

4.2 Бүлгийн дүгнэлт

Энэхүү бүлэг сэдвээр дээрх бүлгүүдэд тодорхойлсон шинжилгээ ба зохиомжийн дагуу хөгжүүлэлтийг хийсэн. Хөгжүүлэлтийн явцад тулгарсан асуудлуудыг шийдвэрлэсэн.

Зураг 4.7: Тэнхим үүсгэх болон санал хүсэлт илгээх хуудас

diploma

Санал хүсэлт

Админ

Хяналтын хэсэг

Тэнхим үүсгэх

Багш нэмэх

Диплом нэмэх

Нууц үг солих

Гарах

id	name	description	created_at	universityId	Action
1	Department 1		2022-04-22T11:51:33.249Z	23	
2	2 Dep	Hello guys	2022-04-22T12:01:38.884Z	23	
3	3 Dep	ello guys	2022-04-22T12:01:58.179Z	23	
4	4 Department	Hello world	2022-04-22T12:13:30.533Z	23	
5	4 Department	Hello world	2022-04-22T12:20:03.586Z	23	
6	6 dep	en baina	2022-04-22T12:22:29.431Z	23	
7	desiredSalary	en	2022-04-22T12:32:57.372Z	23	

<

1

>

Зураг 4.8: Өгөгдөл харах хуудас

Дүгнэлт

Бакалаврын судалгааны ажлын хүрээнд ”Дипломын ажлын бүртгэлийн систем”-ийг хийхээр зорьсон. Тухайн сэдвийн гол зорилго нь олон сургуулиуд өөрсдийн бакалаврын судалгааны ажлыг бүртгэх, тухайн судалгааны ажлыг аль ч сургуулийн оюутан хаанаас ч хэзээ ч үзэх боломжийг олгодог систем юм.

Уг зорилгийн хүрээнд системийн шаардлага хэрэгцээг тодорхойлж, зохиомжийг боловсруулан тухайн зохиомжийн дагуу хөгжүүлэлтийг хийж гүйцэтгэсэн. Хөгжүүлэлтийн явцад зохиомжийг хэт хийсвэрлэлтийн өндөр түвшинд хийсэн байдлаас болж хөгжүүлэлт болон зохиомж хоорондын зөрөлдөөн үүсэх, кодын замбараагүй байдлаас болж хөгжүүлэлт удааших гэж мэт асуудлууд гарч байсан.

Цаашдаа ямар нэгэн програм хангамжийг гүйцэтгэхдээ кодын болон системийн зөв архитектурыг хэрэгжүүлэх нь маш чухал зүйл гэдгийг ойлгож авлаа.

Bibliography

- [1] NextJS Documentation <https://nextjs.org/>
- [2] NestJS Documentation <https://docs.nestjs.com/>
- [3] Prisma Documentation <https://www.prisma.io/>
- [4] Муис цахим каталог <https://catalog.num.edu.mn/cgi-bin/koha/opac-search.pl>

A. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

```
1 // This is your Prisma schema file,
2 // learn more about it in the docs: https://pris.ly/d/prisma-schema
3
4 generator client {
5   provider = "prisma-client-js"
6 }
7
8 datasource db {
9   provider = "postgresql"
10  url       = env("DATABASE_URL")
11 }
12
13 model Admin {
14   id          Int          @id @default(autoincrement())
15   email       String       @unique
16   firstname   String
17   lastname    String
18   phonenumber Int
19   role        String?      @default(value: "admin")
20   password    String
21   created_at  DateTime      @default(now())
22   university  University? @relation(fields: [universityId], references
    : [id])
23   universityId Int?         @unique
24
25   @@unique([email, password])
26 }
27
28 model University {
29   id          Int          @id @default(autoincrement())
30   name        String
31   address     String?
32   phonenumber Int          @unique
33   description  String?
34   created_at  DateTime      @default(now())
35   Admin       Admin?
36   departments Department[]
37 }
38
39 model Department {
40   id          Int          @id @default(autoincrement())
41   name        String
42   description  String?
43   created_at  DateTime      @default(now())
44   University  University? @relation(fields: [universityId], references
    : [id])
45   universityId Int
46   teachers    Teacher[]
```

```

47 }
48
49 model Teacher {
50   id          Int          @id @default(autoincrement())
51   firstname    String
52   lastname     String
53   degree       String
54   photo_url    String?
55   phonenumber  Int?
56   created_at   DateTime    @default(now())
57   department   Department  @relation(fields: [departmentId],
    references: [id])
58   departmentId Int
59   diplomas     TeacherDiploma[]
60 }
61
62 model Diploma {
63   id          Int          @id @default(autoincrement())
64   title        String
65   written_year DateTime
66   file_url     String
67   writer_firstname String?
68   writer_lastname String?
69   writer_stud_id String?
70   description   String?
71   watched_count Int          @default(0)
72   created_at    DateTime    @default(now())
73   teachers      TeacherDiploma[]
74 }
75
76 model TeacherDiploma {
77   id          Int          @id @default(autoincrement())
78   teacherId   Int
79   diplomaId   Int
80   created_at  DateTime    @default(now())
81   Teacher     Teacher    @relation(fields: [teacherId], references: [id])
82   Diploma     Diploma   @relation(fields: [diplomaId], references: [id])
83 }
84
85 model Feedback {
86   id          Int          @id @default(autoincrement())
87   text        String
88   created_at  DateTime    @default(now())
89 }

```

Код A.1: Prisma модел (schema.prisma файл)

```

1
2 import { Injectable } from '@nestjs/common';
3 import { JwtService } from '@nestjs/jwt';
4 import { IsEmail } from 'class-validator';
5 import { PrismaService } from 'src/prisma.service';

```

```

6 import { UsersService } from 'src/users/users.service';
7 import { LoginDto } from './authentication.dto';
8 import * as bcrypt from 'bcrypt';
9 import { generate } from 'generate-password';
10 import { RegisterAuthDto, SuperRegisterDto } from './authentication.dto';
11
12 @Injectable()
13 export class AuthenticationService {
14
15     constructor(private prisma: PrismaService, private usersService:
16         UsersService, private jwtTokenService: JwtService) { }
17
18     async validateUser(loginDto: LoginDto): Promise<any> {
19         const user = await this.usersService.findOne(loginDto.email);
20         if (user) {
21             let isMatch = await bcrypt.compare(loginDto.password, user.
22                 password);
23             if (isMatch) {
24                 const { password, ...result } = user;
25                 return result;
26             }
27             return null;
28         }
29         return null;
30     }
31
32     async login(user: any) {
33         const payload = { username: user.email, userId: user.id, university
34             : user.universityId };
35
36         return {
37             role: user.role,
38             access_token: this.jwtTokenService.sign(payload),
39         };
40     }
41
42     async create(registerAuthDto: RegisterAuthDto) {
43
44         let password = generate({ length: 10, numbers: true });
45         let hash = await bcrypt.hash(password, 10);
46
47         let university = await this.prisma.university.create({
48             data: {
49                 name: registerAuthDto.university_name,
50                 address: registerAuthDto.address,
51                 phonenumber: +registerAuthDto.university_number,
52                 description: registerAuthDto.description || "",
53             }
54         }).catch(err => { console.log(err); return undefined; });
55
56         if (university) {

```

```

54
55     let admin = await this.prisma.admin.create({
56       data: {
57         firstname: registerAuthDto.firstname,
58         lastname: registerAuthDto.lastname,
59         email: registerAuthDto.email,
60         phonenumber: +registerAuthDto.phonenumber,
61         role: registerAuthDto.role,
62         university: { connect: { id: university.id } },
63         password: hash
64       }
65     }).catch(err => { console.log(err); return undefined; });
66
67     if (admin) {
68       return {
69         success: true, message: "Admin created successfully",
70         password: password
71       };
72     } else {
73       return { success: false }
74     }
75   }
76
77   async signUp(superRegisterDto: SuperRegisterDto) {
78
79     let hash = await (bcrypt.hash(superRegisterDto.password, 10));
80     this.prisma.admin.create({
81       data: {
82         email: superRegisterDto.email,
83         firstname: superRegisterDto.firstname,
84         lastname: superRegisterDto.lastname,
85         phonenumber: 88508688,
86         role: "super",
87         password: hash,
88       }
89     }).catch(err => { return err; });
90
91     return { success: true, message: "Admin created successfully" };
92
93   }
94
95
96 }

```

Код A.2: Authentication Service (Хэрэглэгч шалгах)

```

1 import { Injectable } from '@nestjs/common';
2 import { PrismaService } from 'src/prisma.service';
3 import { RegisterAuthDto } from 'src/authentication/authentication.dto'
4 ;
5 import * as bcrypt from 'bcrypt';

```

```

5 import { generate } from 'generate-password';
6 import { ChangePasswordDto } from './users.dto';
7 @Injectable()
8 export class UsersService {
9
10   constructor(private prisma: PrismaService) { }
11
12   async changePassword(username: any, changePasswordDto:
13     ChangePasswordDto) {
14
15     let user = await this.findOne(username);
16     if (user) {
17       let isMatch = await bcrypt.compare(changePasswordDto.oldPassword,
18         user.password);
19       if (isMatch) {
20         let hash = await bcrypt.hash(changePasswordDto.newPassword, 10)
21         ;
22         let result = this.prisma.admin.update({
23           where: {
24             email: username
25           },
26           data: {
27             password: hash
28           }
29         }).catch(err => { console.log(err); return undefined; });
30
31         if (result) {
32           return { success: true, message: "Password changed
33             successfully" };
34         } else {
35           return { success: false, message: "Something went wrong" }
36         }
37       } else {
38         return { success: false, message: "Old password is incorrect" }
39       }
40     } else {
41       return { success: false, message: "Invalid user" }
42     }
43   }
44
45   async findOne(email: string) {
46     return await this.prisma.admin.findUnique({
47       where: {
48         email: email
49       }
50     })
51   }
52 }

```

Код A.3: Users Service

```

1 import { Injectable, UseGuards } from '@nestjs/common';
2 import { PrismaService } from 'src/prisma.service';
3 import { CreateDepartmentDto } from './department.dto';
4 @Injectable()
5 export class DepartmentService {
6
7     constructor(private prisma: PrismaService) { }
8
9     async create(userId: string, username: string, university: number,
10         department: CreateDepartmentDto) {
11
12         let departmentCreated = await this.prisma.department.create({
13             data: {
14                 name: department.name,
15                 description: department.description,
16                 University: {
17                     connect: {
18                         id: +university
19                     }
20                 },
21             },
22         }).catch(err => { console.log(err); return undefined; });
23         return departmentCreated;
24     }
25
26     async findAll(university: any) {
27
28         let deps = [];
29         if (university) {
30
31             deps = await this.prisma.department.findMany({
32                 where: {
33                     universityId: university
34                 }
35             });
36
37             // console.log(deps, "deps");
38             return deps;
39         }
40         return deps;
41     }
42 }

```

Код A.4: Department Service

```

1 import { Injectable } from '@nestjs/common';
2 import { PrismaService } from 'src/prisma.service';
3 import { CreateDiplomaDto } from './diploma.dto';
4 @Injectable()
5 export class DiplomaService {
6     constructor(private prisma: PrismaService) { }

```

```

7
8   async create(createDiploma: CreateDiplomaDto, req: any) {
9
10      console.log(createDiploma, "createDiploma");
11      let diploma = await this.prisma.diploma.create({
12          data: {
13              title: createDiploma.title,
14              written_year: createDiploma.year.toString(),
15              file_url: createDiploma.file,
16              description: createDiploma.description,
17              departmentId: createDiploma.departmentId,
18              writer_firstname: createDiploma.firstname,
19              writer_lastname: createDiploma.lastname,
20              writer_stud_id: createDiploma.studentId,
21              teachers: {
22                  create: createDiploma.teachers.map(teacherId => {
23
24                      return {
25                          Teacher: {
26                              connect: {
27                                  id: +teacherId
28                              }
29                          }
30                      }
31                  })
32              }
33          }
34      }).then(res => { return { success: true, diploma: res } }).
35      catch(err => { console.log(err, "error at diploma register")
36          ; return { success: false, err: err } });
37
38      return diploma
39  }
40
41  async count(req: any) {
42
43      return await this.prisma.diploma.count({
44          where: {
45              teachers: {
46                  some: {
47                      Teacher: {
48                          department: {
49                              universityId: req.user.university
50                          }
51                      }
52                  }
53              }
54          })
55
56  }

```

```

57   async findAll(req: any) {
58
59       if (req.user.university) {
60           let diplomas = await this.prisma.diploma.findMany({
61               where: {
62                   teachers: {
63                       some: {
64                           Teacher: {
65                               department: {
66                                   universityId: req.user.university
67                               }
68                           }
69                       }
70                   }
71               }
72           }).catch(err => { console.log(err, "error at diploma
73                               findAll"); return { success: false, message: "error at
74                               diploma findAll" } }).then(diplomas => { return {
75                               success: true, diplomas: diplomas } });
76
77           return diplomas;
78       }
79       return { success: true, message: "university bish baina",
80               diplomas: [] }
81   }
82 }

```

Код A.5: Diploma Service

```

1  import { Injectable } from '@nestjs/common';
2  import { PrismaService } from "src/prisma.service"
3  import { CreateTeacherDto } from './teacher.dto';
4  @Injectable()
5  export class TeacherService {
6
7       constructor(private prisma: PrismaService) { }
8
9       async create(createTeacherDto: CreateTeacherDto, req: any) {
10
11           let teacher = this.prisma.teacher.create({
12               data: {
13                   firstname: createTeacherDto.firstname,
14                   lastname: createTeacherDto.lastname,
15                   degree: createTeacherDto.degree,
16                   photo_url: createTeacherDto.photo_url,
17                   phonenumber: +createTeacherDto.phonenumber,
18                   email: createTeacherDto.email,
19                   department: {
20                       connect: {
21                           id: createTeacherDto.departmentId
22                       }
23                   }
24               }
25           })
26       }
27   }

```



```

24     }
25     }).catch(err => { console.log(err, "error at teacher create");
      return { success: false, message: "error at teacher create"
      } }).then(teacher => { return { success: true, teacher:
      teacher } });
26
27     return teacher;
28 }
29
30 async findAll(req: any) {
31
32     if (req.user.university) {
33         let teachers = this.prisma.teacher.findMany({
34             where: {
35                 department: {
36                     universityId: req.user.university
37                 }
38             }
39         }).catch(err => { console.log(err, "error at teacher
      findAll"); return { success: false, message: "error at
      teacher findAll" } }).then(teachers => { return {
      success: true, teachers: teachers } });
40
41         return teachers;
42     }
43     return { teachers: [] }
44 }
45 }

```

Код А.6: Teacher Service

```

1  import { Row, Col, Input, Button, Form, Typography, message } from "
      antd"
2  import { LoginService } from "../Services/lib/login";
3  import { useUser } from "../Context/user";
4  import { UseApi } from "../Hooks/useApi";
5  import { useEffect } from "react";
6  const AdminPage = () => {
7
8
9      const [{ data, isLoading, error }, fetch] = UseApi({ service:
      LoginService });
10     const { userAction } = useUser();
11     const { Title } = Typography;
12
13     const loginHandler = (values: any) => {
14
15         const { email, password } = values;
16
17         // console.log(email, "email", password, "passowrd");
18         fetch({ email: email, password: password });
19     }

```

```

20
21   useEffect(() => {
22
23       if (data) {
24
25           if (data.data.statusCode === 401) {
26               message.error("                .");
27           } else {
28               userAction({ action: "login", state: data.data.
29                   access_token, role: data.data.role })
30               // console.log(data, "logindata");
31           }
32       }
33       if (error) {
34           console.log(error, "login error");
35       }
36   }, [data, error])
37
38   return (
39       <Row>
40         <Col className=" login-container justify-content-center d-
41             flex flex-direction-column align-items-center" span={24}>
42           <div className="login">
43             <Title level={3} className="mb-2" > </
44               Title>
45             <br />
46             <Form onFinish={loginHandler} layout="vertical" >
47               <Form.Item name="email" label=" "
48                 required >
49                 <Input placeholder=" " />
50               </Form.Item>
51               <Form.Item name="password" label=" "
52                 required >
53                 <Input placeholder=" " type="password" />
54               </Form.Item>
55               <Form.Item>
56                 <Button htmlType="submit" className="
57                   primary-button" ></Button>
58               </Form.Item>
59             </Form>
60           </div>
61         </Col>
62       </Row>
63     )
64   }
65   export default AdminPage;

```

Код А.7: Нэвтрэх хуудсын хэрэгжүүлэлт

```

1 import { Col, Table, Space } from "antd"
2 import { useRouter } from "next/router";

```

```

3 import Cookies from "universal-cookie"
4 import axios from "axios"
5 import { DeleteOutlined } from "@ant-design/icons"
6 import WithAuth from "../HOC";
7 const DataTable = ({ data }: any) => {
8
9     const router = useRouter();
10    const deleteItem = (itemId: any, type: any) => {
11        console.log(itemId, type);
12    }
13    if (!data && data.length == 0) return <Col span={18} > </Col>;
14
15    let columns = [];
16
17    for (const [key, value] of Object.entries(data[0])) {
18
19        columns.push({
20            title: key,
21            dataIndex: key,
22            key: key,
23        });
24
25    }
26    columns.push({
27        title: 'Action',
28        key: 'action',
29        render: (text: any, record: any) => (
30            <Space align="center" > <DeleteOutlined className="color-
31                button" onClick={() => deleteItem(record.id, router.
32                    query.type)} /> </Space>
33        )
34    })
35    return (
36        <Col span={18} className="mt-1" >
37            <Table scroll={{ x: 500 }} bordered dataSource={data}
38                columns={columns} />
39        </Col>
40    )
41 }
42
43 DataTable.getInitialProps = async ({ query }: any) => {
44
45     const cookie = new Cookies();
46     // let result = [];
47     let config = {
48         headers: {
49             "Authorization": `Bearer ${cookie.get("token_diploma")}`
50         }
51     }
52
53     // console.log(query, "query");
54     if (query.type === "department") {

```

```

52     const result = await axios.get("http://localhost:7000/
53         department/get-all", config)
54     return {
55         data: result
56     }
57 } else if (query.type === "teachers") {
58     const result = await axios.get("http://localhost:7000/teacher/
59         get-all", config)
60
61     if (result.success) {
62         return {
63             data: result.teachers
64         }
65     }
66 } else if (query.type === "diploma") {
67     const result = await axios.get("http://localhost:7000/diploma/
68         get-all", config)
69     if (result.success) {
70         return {
71             data: result.diplomas
72         }
73     }
74 }
75
76 export default WithAuth(DataTable);

```

Код A.8: Өгөгдөл харах хуудас хэрэгжүүлэлт

```

1  import { Col, Row, Form, message, Input, Button, DatePicker, Select }
2      from "antd"
3  import WithAuth from "../HOC"
4  import Cookies from "universal-cookie";
5  import axios from "axios";
6  import { UseApi } from "../Hooks/useApi";
7  import { UploadService } from "../Services/lib/upload";
8  import { CreateDiplomaService } from "../Services/diploma";
9  import { useState, useEffect } from "react"
10 const AddDiplomaPage = ({ departments, teachers }: any) => {
11
12     const [{ data: uploadData, error: uploadError }, uploadFetch] =
13         UseApi({ service: UploadService })
14
15     const [{ data: diplomaData, error: diplomaError },
16         createDiplomaFetch] = UseApi({ service: CreateDiplomaService })
17
18     const [file, setFile] = useState<any>(undefined);
19     const [form] = Form.useForm();
20     const { TextArea } = Input;

```

```

21 const handleFile = (e: any) => {
22     setFile(e.target.files[0]);
23 }
24 const addDiploma = (values: any) => {
25
26     const { title, teacher, department, year, lastname, firstname,
27           id, description } = values;
28
29     let data = {
30
31
32
33     if (file) {
34         let formdata = new FormData();
35         formdata.append('file', file);
36         uploadFetch(formdata);
37     } else {
38         message.error("PDF                !");
39     }
40
41     console.log(data, "values");
42 }
43
44 useEffect(() => {
45
46     if (uploadData) {
47
48         console.log(form.getFieldValue("teacher"), "teachers");
49         createDiplomaFetch({
50             title: form.getFieldValue("title"),
51             year: new Date(form.getFieldValue("year")).getFullYear
52                 (),
53             file: uploadData.data.filename,
54             firstname: form.getFieldValue("firstname"),
55             lastname: form.getFieldValue("lastname"),
56             studentId: form.getFieldValue("id"),
57             description: form.getFieldValue("description"),
58             departmentId: form.getFieldValue("department"),
59             teachers: form.getFieldValue("teacher")
60         })
61     }
62
63     if (uploadError) {
64         console.log(uploadError, "uploadError at diploma.tsx");
65     }
66 }, [uploadData, uploadError])
67
68 useEffect(() => {
69
70     if (diplomaData) {

```

```

71         if (diplomaData.data.success) {
72             message.success("                !");
73         }
74         else {
75             message.error("                !");
76         }
77     }
78     if (diplomaError) {
79         message.error("                !");
80         console.log(diplomaError, "diploma error");
81     }
82 }, [diplomaData, diplomaError])
83
84 return (
85     <Col span={18} className="mt-1" >
86         <Form layout="vertical" form={form} onFinish={addDiploma}
87             >
88             <Form.Item name="title" rules={[{ required: true,
89                 message: "                " }]} label="                " >
90                 <Input placeholder="                " />
91             </Form.Item>
92             <Row gutter={16} >
93                 <Col span={6} >
94                     <Form.Item name="teacher" label="                "
95                         rules={[{ required: true, message: "
96                             " }]} >
97                         <Select optionLabelProp="label" placeholder=
98                             "                " mode="multiple" >
99                             {teachers && teachers.map((teacher: any
100                                 ) => <Select.Option key={teacher.id}
101                                     label={`_${teacher.lastname.slice(0,
102                                         3)}.${teacher.firstname}`} value={
103                                         teacher.id} >{teacher.lastname.slice
104                                             (0, 3)}. {teacher.firstname}</Select
105                                             .Option>)}
106                             </Select>
107                         </Form.Item>
108                 </Col>
109                 <Col span={6} >
110                     <Form.Item name="department" label="
111                         " rules={[{ required: true, message: "
112                             " }]} >
113                         <Select placeholder="                " >{departments &&
114                             departments.map((department: any) => <
115                                 Select.Option key={department.id} value
116                                 ={department.id} >{department.name}</
117                                 Select.Option>)}</Select>
118                     </Form.Item>
119                 </Col>
120                 <Col span={6} >
121                     <Form.Item label="                " name="year" rules={[{
122                         required: true, message: "                " }]} >

```

```

105         <DatePicker placeholder=" " picker="year"
106             />
107     </Form.Item>
108 </Col>
109 <Col span={6} >
110     <Form.Item label="(pdf)" name="
111         file" rules={[{ required: true, message: "
112             " }]} >
113         <Input type="file" onChange={handleFile} />
114     </Form.Item>
115 </Col>
116 </Row>
117 <Row gutter={16} >
118     <Col span={6} >
119         <Form.Item name="lastname" label="
120             "
121             rules={[{ required: true, message: "
122                 " }]} >
123             <Input placeholder="
124                 " />
125         </Form.Item>
126     </Col>
127     <Col span={6} >
128         <Form.Item name="firstname" label="
129             "
130             rules={[{ required: true, message: "
131                 " }]} >
132             <Input placeholder="
133                 " />
134         </Form.Item>
135     </Col>
136     <Col span={6} >
137         <Form.Item name="id" label="
138             " rules
139             = {[{ required: true, message: "
140                 " }]} >
141             <Input placeholder="
142                 " />
143         </Form.Item>
144     </Col>
145 </Row>
146 <Form.Item name="description" label="
147     " >
148     <TextArea rows={4} placeholder="
149         " />
150 </Form.Item>
151 <Form.Item>
152     <Button className="primary-button" htmlType="submit"
153         " > </Button>
154 </Form.Item>
155 </Form>
156 </Col>
157 )
158 }
159
160 AddDiplomaPage.getInitialProps = async (ctx: any) => {
161
162     const cookie = new Cookies();
163     let config = {
164         headers: {

```

```

147         "Authorization": `Bearer ${cookie.get("token_diploma")}`
148     }
149 }
150
151 const departments = await axios.get("http://localhost:7000/
    department/get-all", config);
152 const teachers = await axios.get("http://localhost:7000/teacher/get
    -all", config);
153 console.log(departments, teachers, "fatufak")
154 return {
155     departments: departments,
156     teachers: teachers.success ? teachers.teachers : []
157 }
158 }
159
160 export default WithAuth(AddDiplomaPage);

```

Код А.9: Диплом бүртгэх хуудас