

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

Баянжаргалын Энх-Амгалан

Вэб хөгжүүлэлт
(Web developing)

Мэдээллийн технологи (D061303)
Үйлдвэрлэлийн дадлагын тайлан

Улаанбаатар

2025 оны 01 сар

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

Вэб хөгжүүлэлт
(Web developing)

Мэдээллийн технологи (D061303)
Үйлдвэрлэлийн дадлагын тайлан

Удирдагч:	_____	Др. Ч.Алтангэрэл
Хамтран удирдагч:	_____	Н.Од-Эрдэнэ
Гүйцэтгэсэн:	_____	Б.Энх-Амгалан (21B1NUM0344)

Улаанбаатар
2025 оны 01сар

Зохиогчийн баталгаа

Миний бие Баянжаргалын Энх-Амгалан ”Вэб хөгжүүлэлт” 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	3
1. БАЙГУУЛЛАГЫН ТАНИЛЦУУЛГА	3
1.1 Товч Танилцуулга	3
1.2 Ямар Үйлчилгээ Үзүүлдэг Вэ?	3
1.3 Ямар Систем Дээр Голчлон Төвлөрдөг Вэ?	3
2. СИСТЕМИЙН ШААРДЛАГА	5
2.1 Хэрэглэгчийн шаардлага	5
2.2 Use Case диаграм	8
3. АШИГЛАХ ТЕХНОЛОГИ	9
3.1 Gitlab	9
3.2 NextJS	9
3.3 Golang	9
3.4 PostgreSQL	10
3.5 Postman	11
3.6 Docker	11
4. ХЭРЭГЖҮҮЛЭЛТ	12
4.1 GORM хэрэгжүүлэлт	12
4.2 Golang Test хэрэгжүүлэлт	18
4.3 Front-End хэрэгжүүлэлт	23
4.4 Back-End хэрэгжүүлэлт	36
5. ДҮГНЭЛТ	43
5.1 Үр дүн	43
5.2 Үр дүнгийн тайлан	45

НОМ ЗҮЙ	47
ХАВСРАЛТ.....	48
А. УДИРДАГЧИЙН ҮНЭЛГЭЭ	48
В. КОДЫН ХЭРЭГЖҮҮЛЭЛТ.....	49
В.1 GORM.....	49
В.2 Тест	51
В.3 Front-end	55
В.4 Back-End	62

ЗУРГИЙН ЖАГСААЛТ

2.1	Use Case диаграм	8
5.1	Хэрэглэгчидын жагсаалт	43
5.2	Хэрэглэгч засах	44
5.3	Өдөр сонгох календар	44
5.4	Ажилтны эрх сонгох хэсэг	45

ХҮСНЭГТИЙН ЖАГСААЛТ

1	Төлөвлөгөө гаргасан байдал	2
---	----------------------------------	---

Кодын жагсаалт

4.1	Get all users function	13
4.2	Create user function	13
4.3	Update user function	14
4.4	Delete user function	15
4.5	Get user by ID function	15
4.6	Check if login ID exists function	16
4.7	Test Database Setup	18
4.8	Test Create User Function	19
4.9	Test Get All Users Function	20
4.10	Test Update User Function	21
4.11	UserFormData интерфэйсийг үүсгэсэн байдал	24
4.12	UserFormPage функц (Main component)	24
4.13	Хувьсагч	24
4.14	initialFormData хоосон утга оноосон байдал	25
4.15	Төлөв хадгалах	25
4.16	Хэрэглэгчийн мэдээлэл татах useEffect	26
4.17	handleChange	27
4.18	handleSubmit	27
4.19	UI бүтэц	28
4.20	CreateUser Controller	38
4.21	CreatUser Controller	39
4.22	UpdateUser Controller	40
4.23	DeleteUser Controller	41
4.24	CheckLoginIDExists Controller	42

УДИРТГАЛ

Миний бие Б.Энх-Амгалан нь үйлдвэрлэлийн дадлагын хугацаанд Next.js, Golang гэсэн технологиуд дээр голчлон ажилласан ба уг технологиуд ямар шалтгаанаар үүссэн, цаана нь технологийн ямар дэвшил, хөгжүүлэлтийн арга барил ашигладаг, компаниуд хэрхэн үүн дээр хөгжүүлэлт хийж эцсийн бүтээгдэхүүнийг гаргадаг талаар судлахын тулд front-end хөгжүүлэлт дээрээ Next.js фрэймворк болон back-end хөгжүүлэлт дээрээ Golang ашигладаг компани болох "ОНДО" ХХК-г сонгон авч мэргэжлийн дадлагаа гүйцэтгэлээ.

Зорилго Next.js болон Golang технологийн талаар судалж, компанийн хөгжүүлэлтийн арга барилтай танилцах

Зорилт Удирдагчийн зааварчилгааны дагуу алхам алхмаар судалгаа хийж өгсөн шаардлагын хүрээнд хэрэгжүүлэлт хийх

Table 1: Төлөвлөгөө гаргасан байдал

№	Гүйцэтгэх ажил	Хугацаа	Биелэлт	Үнэлгээ
1	<i>SQL-ийг ORM-руу шилжүүлсэн</i>	<i>4 хоног</i>		
2	<i>Unite Test</i>	<i>3 хоног</i>		
3	<i>Admin Front-End хөгжүүлэлт</i>	<i>7 хоног</i>		
4	<i>Admin Back-End хөгжүүлэлт</i>	<i>7 хоног</i>		

Баталсан: Албан байгууллагын дадлага удирдагч:.....

Төлөвлөгөө боловсруулсан: Оюутан

1. БАЙГУУЛЛАГЫН ТАНИЛЦУУЛГА

ONDO бол цаг үедээ шинэ давлагаа. Монголын хэрэглэгчдэд үйлчилгээ үзүүлж ашиг олох гэхээсээ илүүтэй Монголд хийсэн технологио дэлхийн зах зээлд хүргэхээр ажиллаж буй компани юм. Stakeholder economy буюу төсөлд хамааралтай бүх хүн өгөөжийг хүртдэг эдийн засгийн тогтолцоог бий болгохоор зорьж, ёс зүйтэй, нийгмийн бүх хэсгүүдэд хүрч үйлчлэх тэр дундаа интернэтийг шаардлагатай бүлгийн иргэдэд таатай үнээр хүргэхээр ажилладаг. Нийт 120 залуусаас ONDO гэр бүл бүрддэг. Зах зээлд нэвтрээд 2 жилийн хугацаанд бид 385 мянган хэрэглэгчидтэй болжээ.

1.1 Товч Танилцуулга

2018 онд анх ONDO компани үүсгэн байгуулагдсан. Энэ үедээ ONDO нэрээ авч амжаагүй байсан бөгөөд үйл ажиллагаагаа албан ёсоор 2021 онд явуулж эхлэсэн. Хамгийн анхны бүтээгдэхүүнээ 2022 оны 3 сард зах зээлд нэвтрүүлсэн бөгөөд интернэтийн үйлчилгээг ХУГАЦААГҮЙ болон ХЯЗГААРГҮЙ байх нөхцлийг хэрэглэгчиддээ олгосон. Цаашдаа дэлхийн зах зээлд гарахаар зорин ажиллаж байна.

1.2 Ямар Үйлчилгээ Үзүүлдэг Вэ?

ONDO нь шинэ үеийн үүрэн холбооны үйлчилгээ үзүүлэгч компани. Тус компани нь хэрэглэгчдэдээ дэвшилтэт технологи, хурдан сүлжээ, мобайл дата, уян хатан үйлчилгээний багцуудыг санал болгодог. Хэрэглэгчид хангах заавал салбар оролгүй хүссэн газраасаа үйлчилгээ авах боломжыг бүрдүүлдэг.

1.3 Ямар Систем Дээр Голчлон Төвлөрдөг Вэ?

ONDO нь дижитал платформд суурилсан, хэрэглэгч төвтэй системүүд дээр ажилладаг

1.3. ЯМАР СИСТЕМ ДЭЭР ГОЛЧЛОШ ТӨЛӨРДӨЙ БЭЛЭЛЛЭГЭЙН ТАНИЛЦУУЛГА

бөгөөд хэрэглэгчдэд хурдан, найдвартай үйлчилгээ үзүүлэхэд анхаардаг.

2. СИСТЕМИЙН ШААРДЛАГА

Миний хувьд дадлагын 21 хоногийн хугацаа дахь сүүлийн долоо хоногт уг системийн мерчант вэб хэсэг дээр ажиллаж эхэлсэн ба гүйцэт биш ч тодорхой хэмжээнд системийн талаар судалгаа хийснээ хүргэж байна. Мөн уг систем дээр ажиллахын тулд нууцлалын гэрээ хийсэн учир дэлгэрэнгүй тайлбарлах боломжгүй.

Online аппыг бодит амьдрал дээрх олон дэлгүүр байршдаг нэг барилга гэж үзэж болох ба та тус апп руу орсноор яг л нэг молл дотор дэлгүүр хэсэж байгаа юм шиг хамгийн түрүүнд Online дээр гэрээтэй дэлгүүрүүдийн жагсаалт харагдана. Уг байдлаараа монголд хөгжиж буй бусад e-commerce-уудаас ялгаатай билээ. Харин миний ажилласан мерчант системийн гол үүрэг нь тухайн апп доторх дэлгүүрийг дэлгүүрийн эзэн буюу мерчант админ бүрэн удирддаг байх шаардлагатай.

2.1 Хэрэглэгчийн шаардлага

2.1.1 Хэрэглэгчид

- Мерчант админ гэсэн ердөө ганц хэрэглэгчээс бүрдэнэ.

2.1.2 Функционал хэрэглэгчийн шаардлагууд

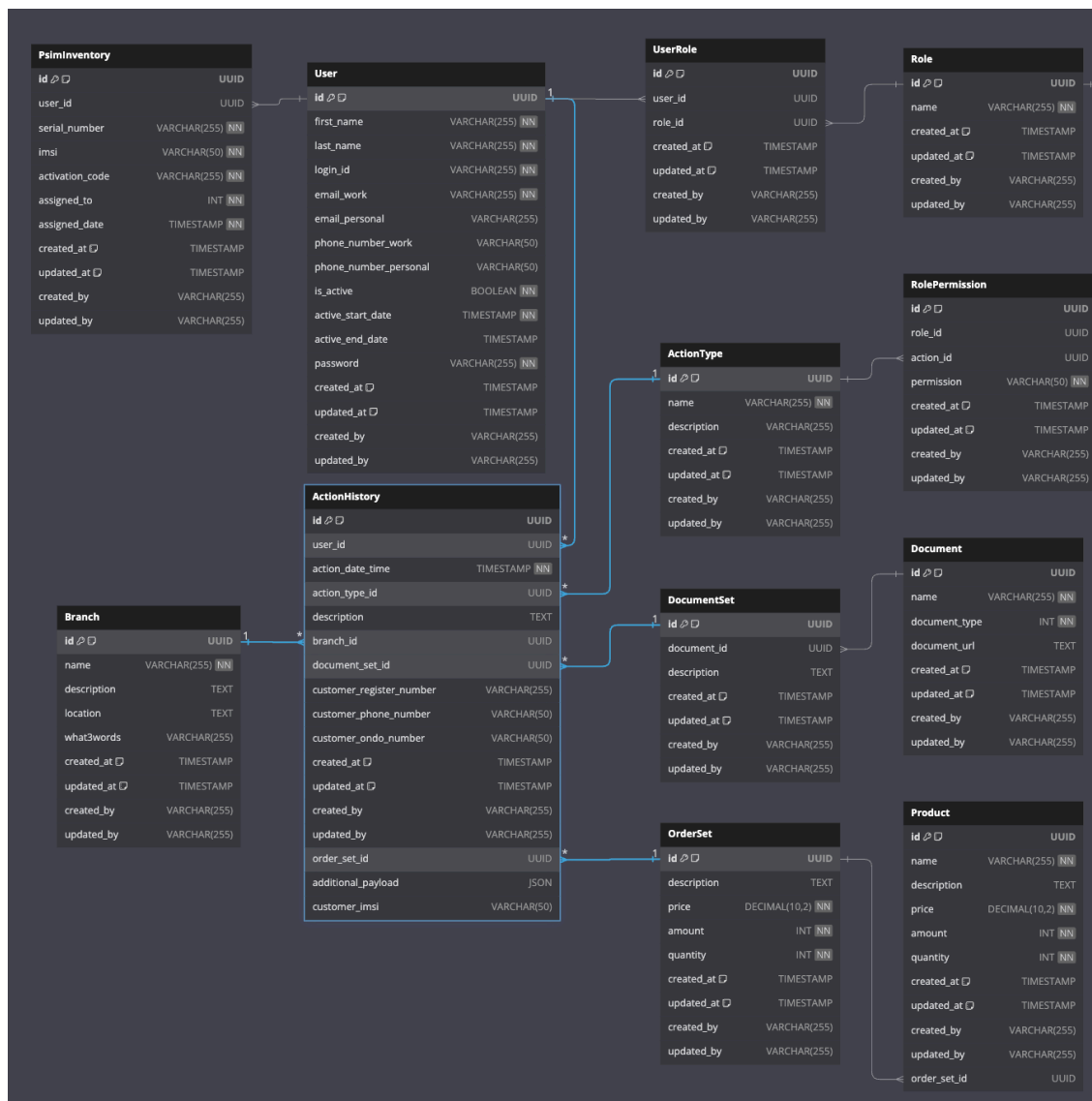
- Хэрэглэгч зөвхөн имэйлээр ирсэн холбоосоор нэвтэрдэг байх
- Хэрэглэгч өөрийн хаяг дээрээ олон дэлгүүр үүсгэх боломжтой байх
- Дэлгүүр бүр ажиллах цагийн хуваарьтай байх
- Дэлгүүр бүр cover болон icon зурагтай байх
- Дэлгүүр бүр бүтээгдэхүүний ангилалтай байх ёстой
- Бүтээгдэхүүний цуглуулга буюу коллекцийг үүсгэдэг байх

- Коллекц заавал cover болон icon зураг авдаг байх
- Хямдрал зарлах боломжтой байх
- Хямдралыг төрлөөр нь хувь, үнэ гэсэн хоёр хуваадаг байх
- Бүх бүтээгдэхүүн дээр хямдрал зарлах боломжтой байх
- Зөвхөн коллекци дээр хямдрал зарлах боломжтой байх
- Хэрэглэгчийн сервертэй харьцсан бүх үйлдэлд Toast-р хариу өгдөг байх
- Бүтээгдэхүүн үүсгэж болдог байх
- Бүтээгдэхүүн заавал нэг буюу түүнээс дээш зурагтай байх
- Бүтээгдэхүүний оруулж буй бүх зураг 1:1 хэмжээтэй байх
- Бүтээгдэхүүн заавал үнэтэй байх
- Мерчант дээр хуулагдаж буй бүх зураг alt таг авдаг байх
- Үүсгэсэн бүтээгдэхүүнээ засдаг байх
- Нүүр хэсэгт тухайн сонгосон дэлгүүрийн борлуулалтын статистикуыг харуулдаг байх
- Нүүр хэсэгт тухайн сонгосон дэлгүүр дээрх барааны жагсаалт харагддаг байх

2.1.3 Функционал бус шаардлага

- Вэбсайт нь хэрэглэхэд хялбар байх
- Бүх хуудасны арын суурь өнгө ижил байх
- Responsive дизайнтай байх
- Бүтээгдэхүүн үүсгэх хэсэг хялбар байх

- Цагийн хуваарь болон дэлгүүрийн бусад мэдээлэл оруулахад ойлгомжтой байх
- Бүх хуудас тусламж авах хэсэгтэй байх



Зураг 2.1: Use Case диаграм

2.2 Use Case диаграм

Мерчант вэб болон Oneline дээр хийгдсэн Use Case диаграм

3. АШИГЛАХ ТЕХНОЛОГИ

3.1 Gitlab

GitLab нь Git дээр суурилсан DevOps платформ бөгөөд програм хангамж хөгжүүлэх, тест хийх, нэвтрүүлэх процессыг автоматжуулан нэг дор удирдах боломжийг олгодог. GitLab нь CI/CD (Continuous Integration/Continuous Deployment) багтаасан байдаг ба энэ нь хөгжүүлэгчид өөрсдийн кодыг турших, хянах, шинэчлэлтүүдийг хурдан бөгөөд найдвартай нэвтрүүлэхэд тусалдаг. Хөгжүүлэгчид өөрсдийн төслийг GitLab дээр байршуулж, хамтын ажиллагааг хөнгөвчилж, кодын чанарыг сайжруулах боломжтой байдаг.

3.2 NextJS

Next.js нь React-д суурилсан, сервер талын рендэрлэг SSR (Server-side rendering), статик сайт үүсгэгч (Static Site Generation), API маршрут боловсруулагч зэрэг олон төрлийн функцуудыг агуулсан хүчирхэг framework юм. Next.js нь хурдан, SEO (search engine optimization)-д ээлтэй вэб аппликейшнуудыг бүтээхэд туслах зорилготой. Энэ нь хуудасны динамик маршрут, API endpoint-ууд, автомат статик зэрэг олон төрлийн функцуудыг агуулдаг. Next.js нь хөгжүүлэгчидэд илүү хялбар бөгөөд хурдан хөгжүүлэлтийг хангахын зэрэгцээ, вэб аппликейшны гүйцэтгэлийг сайжруулдаг. Мөн Next.js нь TypeScript-ийг дэмжиж, хөгжүүлэгчидэд илүү найдвартай, уншигдахуйц код бичих боломжийг олгодог.

3.3 Golang

Golang буюу Go нь Google-ээс гаргасан, хурдан, найдвартай, энгийн синтакс бүхий програмчлалын хэл юм. Go нь системийн програмчлал, сүлжээний програмчлал, cloud computing, микросервис архитектур зэрэг олон төрлийн хэрэглээнд тохиромжтой. Go хэл нь статик төрөлтэй, garbage

collection-тэй, multi-threading-ийг хялбаршуулсан goroutine-уудтай байдаг. Энэ нь хөгжүүлэгчидэд өндөр гүйцэтгэлтэй, масштабдах боломжтой програмуудыг бүтээх боломжийг олгодог. Go нь мөн стандарт сангуудын багцтай байдаг бөгөөд энэ нь хөгжүүлэгчидэд нэмэлт сангуудгүйгээр олон төрлийн ажиллагааг хэрэгжүүлэх боломжийг олгодог.

Go хэл нь мөн GORM гэдэг объектын холбоосын (ORM) санг дэмждэг бөгөөд энэ нь Go хэл дээрх өгөгдлийн сантай ажиллах процессыг хялбаршуулдаг. GORM нь хөгжүүлэгчдэд SQL болон өгөгдлийн сан холбохыг хялбаршуулж, объектуудыг шууд өгөгдлийн сангийн хүснэгтүүдэ хамааруулан ажиллах боломжийг олгодог. Энэ нь CRUD (Create, Read Update, Delete) үйлдлүүдийг гүйцэтгэх, өгөгдлийн миграци, холбол хийх зэрэг үүргийг гүйцэтгэх боломжийг бүрдүүлдэг. GORM-ийн ашиглах нь Go-ийн синтаксийн энгийн байдал, хурдтай холбогдсон, мөн өгөгдлийн сангийн бүхий л үйлдлүүдийг энгийн, хурдан гүйцэтгэх боломжоор хангадаг.

3.4 PostgreSQL

PostgreSQL нь хүчирхэг, нээлттэй open source объект хандлагат database бөгөөд сүүлийн 30 гаруй жилийн турш идэвхтэй хөгжиж байгаа найдвартай ажиллагаатай мэдээллийн сангийн програм юм. 1986 онд Калифорни их сургуулийн “Postgres” төслийн нэг хэсэг болж хөгжсөн түүхтэй. PostgreSQL ихэнх үйлдлийн системүүд дээр ажилладаг ба PostGIS гэсэн орон зайн мэдээллийн сангийн нэмэлт хэрэгсэлтэйгээрээ бусад мэдээллийн сангийн програмуудаас давуу талтай. Мэдээллийн сан бол таны мэдээллийг найдвартай хадгалах үүрэгтэй. PostgreSQL мэдээллийн сан нь SQL хэлний дийлэнх стандартууд болон орчин үед хэрэглэгдэж байгаа ихэнх мэдээллийн сангийн цогц асуулгууд буюу queries, гадаад түлхүүр буюу foreign keys, триггер командууд, өөрчлөлт хийх боломжтой view, мэдээ дамжуулах хурд, гэх мэт боломжуудыг өөртөө агуулсан байдаг. Мөн хэрэглэгч нь мэдээний төрөл, функцууд, үйлдлүүд, индексийн арга зэргийг өөрийн хэрэгцээнд зориулан хөгжүүлэх боломжтой нээлттэй эх үүсвэрийн мэдээллийн сан юм. PostgreSQL системийн үндсэн архитектурыг харвал клиент / серверийн загварыг

ашигладаг.

3.5 Postman

Postman нь API хөгжүүлэлт, тест хийх, баримтжуулалт хийхэд ашиглагддаг хэрэгсэл юм. Энэ нь хөгжүүлэгчидэд API-уудыг хялбархан турших, өөрчлөлт оруулах, багийн гишүүдтэй хамтран ажиллах боломжийг олгодог. Postman нь REST, GraphQL, SOAP зэрэг олон төрлийн API-уудыг дэмждэг бөгөөд хөгжүүлэгчидэд API-ийн хүсэлт, хариуг хялбархан үүсгэх, илгээх, хариу үйлдлийг шалгах боломжийг олгодог. Мөн Postman нь автомат тест үүсгэх, тест сценари бичих, API-ийн гүйцэтгэлийг шалгах боломжийг олгодог. Энэ нь хөгжүүлэлтийн явцыг хурдасгах, API-ийн найдвартай байдлыг нэмэгдүүлэхэд тусалдаг.

3.6 Docker

Docker нь аппликейшнуудыг контейнер хэлбэрээр багцлах, ажиллуулах, түгээхэд ашиглагддаг платформ юм. Docker нь аппликейшны хөгжүүлэлт, тест хийх, production орчинд хүргэх явцыг хялбаршуулдаг. Контейнерчилал нь аппликейшны бүх шаардлагатай нөөцийг (код, санууд, системийн хэрэгслүүд) нэг дор багцлах боломжийг олгодог бөгөөд энэ нь аппликейшныг ямар ч орчинд ажиллуулах боломжийг олгодог. Docker нь хөгжүүлэгчидэд аппликейшны орчны тогтвортой байдлыг хангах, аппликейшныг хурдан, найдвартайгаар хүргэх боломжийг олгодог. Мөн Docker нь микросервис архитектурт тохиромжтой бөгөөд энэ нь орчин үеийн cloud-native аппликейшнуудыг хөгжүүлэхэд тусалдаг.

4. ХЭРЭГЖҮҮЛЭЛТ

4.1 GORM хэрэгжүүлэлт

Уг функцуудыг шаардлагын дагуу алхам алхмаар хэрэгжүүлсэнээр GORM-ийг ашиглах анхан шатны туршлагатай болох зорилготой байсан ба цаашид ажилласан төсөл дээр GORM-ийн дотоод логик, өгөгдлийн сангийн холболт, тайлбар хийх, өгөгдөл хадгалах болон шинэчлэх процессуудын ажиллагааг ойлгох, тодорхой хэмжээнд практик мэдлэгийг цуглуулж чадсан.

Сурах ур чадвар:

- GORM-ийг ашиглах чадвараа нэмэгдүүлэх
- Өгөгдлийн сантай харьцах үйлдлүүд (GET, POST, UPDATE, DELETE) болон тэдгээрийн процессийг ойлгох
- Хэрэглэгчийн логийг бүртгэх болон алдааг барих чадвар эзэмших
- CRUD үйлдлүүдийг GORM ашиглан бичих
- Горм-ийг ашиглахдаа data validation болон password hashing-ийн талаар мэдлэгтэй болох

Функцийн шаардлага:

- GORM ашиглан хэрэглэгчийн мэдээллийг авах, нэмэх, засах, устгах
- Хэрэглэгчийн нэвтрэх ID-г шалгах функц бичих
- Алдааг бүртгэх болон лог хийх функц ашиглах
- GORM-ийн дотоод функцууд болон query үйлдлийг ойлгох
- Хэрэглэгчийн мэдээлэл устгах үед хамааралтай мэдээллийг устгах
- Controller давхаргыг зөв зохион байгуулах

4.1.1 User Management Functions

Энэхүү хэсэгт хэрэглэгчийн мэдээллийг удирдах хэд хэдэн функцийг оруулсан болно. Үүнд хэрэглэгчийн мэдээллийг авах, шинэчлэх, устгах, болон логины ID-г шалгах үйлдлүүд орно.

```
1 func GetAllUsers(r *http.Request) ([]models.User, error) {
2     var users []models.User
3     if err := config.DB.Find(&users).Error; err != nil {
4         log.Error("Failed to fetch users", map[string]interface{}{"error":
5             err.Error()}, r)
6         return nil, err
7     }
8     return users, nil
9 }
```

Код 4.1: Get all users function

Энэ функц нь бүх хэрэглэгчдийн мэдээллийг өгөгдлийн сангаас авч, ямар нэгэн алдаа гарсан тохиолдолд тухайн алдааг логлош болно.

```
1 func CreateUser(user models.User, r *http.Request) (models.User, error)
2     {
3     existingUser := config.DB.Where("login_id=?", user.LoginID).First(&
4         models.User{})
5     if existingUser.RowsAffected > 0 {
6         return models.User{}, errors.New("user already exists")
7     }
8
9     hashedPassword, err := utils.HashPassword(user.Password)
10    if err != nil {
```

```
9     return models.User{}, err
10 }
11
12 user.Password = hashedPassword
13
14 if err := config.DB.Create(&user).Error; err != nil {
15     return models.User{}, err
16 }
17 return user, nil
18 }
```

Код 4.2: Create user function

Шинэ хэрэглэгчийг үүсгэх үед, хэрэглэгчийн ID өмнө нь байгаа эсэхийг шалгаад, байсан бол алдаа буцаана. Мөн хэрэглэгчийн нууц үгийг хуулаад өгөгдлийн сан дээр хадгална.

```
1 func UpdateUser(id string, user models.User, r *http.Request) (models.
   User, error) {
2     hashedPassword, err := utils.HashPassword(user.Password)
3     if err != nil {
4         return models.User{}, err
5     }
6
7     user.Password = hashedPassword
8     if err := config.DB.Model(user).Where("id=␣?", id).Updates(user).
       Error; err != nil {
9         return models.User{}, err
10    }
11    return user, nil
12 }
```

Код 4.3: Update user function

Энэ функц нь хэрэглэгчийн мэдээллийг шинэчилж, нууц үгийг шинэчлэх шаардлагатай бол шинэчилнэ.

```

1 func DeleteUser(id string, r *http.Request) error {
2     if err := config.DB.Where("user_id=?", id).Delete(&models.UserRole
3         {})).Error; err != nil {
4         return err
5     }
6     if err := config.DB.Where("id=?", id).Delete(&models.User{}).Error;
7         err != nil {
8         return err
9     }
10    return nil
11 }

```

Код 4.4: Delete user function

Хэрэглэгчийг устгах үед түүний хэрэглэгчийн үүрэг болон хэрэглэгчийн мэдээллийг устгана.

```

1 func GetUserByID(id string, r *http.Request) (*models.User, error) {
2     log.Info("Fetching user by ID from the database", map[string]
3         interface{}{"id": id}, r)
4     var user models.User
5
6     err := config.DB.Where("id=?", id).First(&user).Error
7     if err != nil {

```

```
8     if errors.Is(err, gorm.ErrRecordNotFound) {
9         log.Warn("User_not_found", map[string]interface{}{"id": id}, r)
10        return nil, nil
11    }
12    log.Error("Database_query_error", map[string]interface{}{"error":
13        err.Error()}, r)
14    return nil, err
15
16    log.Info("User_fetched_successfully", map[string]interface{}{"id": id
17        }, r)
18    return &user, nil
19 }
```

Код 4.5: Get user by ID function

Энэхүү функц нь хэрэглэгчийн ID-г ашиглан мэдээллийг өгөгдлийн сангаас авдаг. Хэрэв хэрэглэгч олдвол түүний мэдээллийг буцаана.

```
1 func CheckLoginIDExists(loginID string, r *http.Request) (bool, error)
2 {
3     var user models.User
4     result := config.DB.Where("login_id=?", loginID).First(&user)
5
6     if result.Error != nil {
7         if errors.Is(result.Error, gorm.ErrRecordNotFound) {
8             return false, nil
9         }
10        log.Error("Failed_to_check_login_id", map[string]interface{}{"error":
11            result.Error.Error()}, r)
12    }
```



```
10     return false, result.Error
11 }
12
13 return true, nil
14 }
```

Код 4.6: Check if login ID exists function

Энэ функц нь өгөгдлийн сан дээрх login ID-гийн байршлыг шалгадаг. Хэрэв байвал true, байхгүй бол false буцаана.

4.2 Golang Test хэрэгжүүлэлт

Энэхүү хэсэгт бид Go хэл дээрх үйлдлүүдийг тест хийх зориулалттай функцуудыг хэрэгжүүлсэн болно. Тестүүд нь хэрэглэгчийн мэдээллийг нэмэх, авах, шинэчлэх, устгах зэрэг CRUD үйлдлүүдийг шалгана.

4.2.1 Тестийн орчин

Тестийн орчин байгуулахдаа өгөгдлийн санг анхны нөхцөлд нь оруулж, дараа нь алдаагүйгээр бүх функцуудыг туршина. Энэ зорилгоор дараах функцүүдийг ашиглав:

```
1 func setupTestDB() *gorm.DB {  
2     test.ConnectTestDatabase()  
3     config.DB.Exec("TRUNCATE TABLE users RESTART IDENTITY CASCADE")  
4     return config.DB  
5 }
```

Код 4.7: Test Database Setup

Энэхүү функц нь тестийн өгөгдлийн санг тохируулж, хэрэглэгчийн хүснэгтийг шинэчилнэ.

4.2.2 Тестийн функцууд

- `TestCreateUser`: Шинэ хэрэглэгч үүсгэх үйлдлийг шалгана.
- `TestGetAllUsers`: Бүх хэрэглэгчийн мэдээллийг авах функцийг туршина.
- `TestUpdateUser`: Хэрэглэгчийн мэдээллийг шинэчлэх үйлдлийг шалгана.
- `TestDeleteUser`: Хэрэглэгчийн мэдээллийг устгах үйлдлийг туршина.
- `TestGetUserByID`: Хэрэглэгчийн ID-ээр мэдээлэл авах үйлдлийг шалгана.

TestCreateUser

Энэхүү тест нь шинэ хэрэглэгчийг үүсгэж, үүсгэх үед гарах алдааг шалгах бөгөөд хэрэглэгчийн нууц үгийг хашлах явцыг шалгана.

```
1 func TestCreateUser(t *testing.T) {
2     db := setupTestDB()
3     defer test.CloseTestDatabase()
4
5     loginID := "johndoe"
6     dropExistingUser(db, loginID)
7
8     r := httptest.NewRequest(http.MethodPost, "/", nil)
9     user := createSampleUser(loginID)
10
11     createdUser, err := services.CreateUser(user, r)
12
13     assert.NoError(t, err, "Error creating user")
14     assert.NotZero(t, createdUser.ID, "User ID should not be zero")
15     assert.NotEqual(t, "password123", createdUser.Password, "Password
16         should be hashed")
17
18     var fetchedUser models.User
19     err = db.First(&fetchedUser, "id=?", createdUser.ID).Error
20     assert.NoError(t, err, "Error fetching user")
21
22     assert.Equal(t, "John", fetchedUser.FirstName, "First name mismatch")
23
24     assert.Equal(t, "Doe", fetchedUser.LastName, "Last name mismatch")
25     assert.NotEqual(t, "password123", fetchedUser.Password, "Password
```

```
24         should_be_hashed")
    }
```

Код 4.8: Test Create User Function

TestGetAllUsers

Энэхүү тест нь өгөгдлийн сан дахь бүх хэрэглэгчийн мэдээллийг авах үйлдлийг шалгана.

Мөн хэрэглэгчдийн тоог болон мэдээллийг баталгаажуулна.

```
1 func TestGetAllUsers(t *testing.T) {
2     db := setupTestDB()
3     defer test.CloseTestDatabase()
4
5     db.Exec("TRUNCATE TABLE users RESTART IDENTITY CASCADE")
6
7     users := []models.User{
8         {FirstName: "John", LastName: "Doe"},
9         {FirstName: "Jane", LastName: "Smith"},
10    }
11    for _, user := range users {
12        db.Create(&user)
13    }
14
15    r := httptest.NewRequest(http.MethodGet, "/users", nil)
16    allUsers, err := services.GetAllUsers(r)
17
18    assert.NoError(t, err, "Error fetching all users")
19    assert.Len(t, allUsers, len(users), "Unexpected number of users fetched")
}
```

```
20
21     for i, fetchedUser := range allUsers {
22         assert.True(t, i < len(users), "Index_%d_out_of_range_for_
23             expected_users", i)
24         assert.Equal(t, users[i].FirstName, fetchedUser.FirstName,
25             "First_name_mismatch._Expected:_%s,_Got:_%s", users[i].
26                 FirstName, fetchedUser.FirstName)
27         assert.Equal(t, users[i].LastName, fetchedUser.LastName,
28             "Last_name_mismatch._Expected:_%s,_Got:_%s", users[i].
                LastName, fetchedUser.LastName)
29     }
30 }
```

Код 4.9: Test Get All Users Function

TestUpdateUser

Энэхүү тест нь хэрэглэгчийн мэдээллийг шинэчлэх үйлдлийг шалгадаг.

```
1 func TestUpdateUser(t *testing.T) {
2     db := setupTestDB()
3     defer test.CloseTestDatabase()
4
5     // Create sample user
6     originalUser := createSampleUser("johndoe")
7     db.Create(&originalUser)
8
9     // Prepare updated user data
10    updatedUserData := models.User{
11        FirstName: "Jane",
```

```
12     LastName:  "Smith",
13     Password:  "newpassword123",
14 }
15
16 r := httptest.NewRequest(http.MethodPut, "/user", nil)
17 updatedUser, err := services.UpdateUser(originalUser.ID,
18     updatedUserData, r)
19
20 assert.NoError(t, err, "Error updating user")
21 assert.Equal(t, updatedUserData.FirstName, updatedUser.FirstName, "
22     First_name_mismatch")
23
24 assert.Equal(t, updatedUserData.LastName, updatedUser.LastName, "
25     Last_name_mismatch")
26
27 assert.NotEqual(t, "newpassword123", updatedUser.Password, "
28     Password_should_be_hashed")
29
30 var fetchedUser models.User
31 err = db.First(&fetchedUser, "id=?", originalUser.ID).Error
32 assert.NoError(t, err, "Error fetching updated user")
33
34 assert.Equal(t, updatedUserData.FirstName, fetchedUser.FirstName, "
35     First_name_mismatch_in_database")
36
37 assert.Equal(t, updatedUserData.LastName, fetchedUser.LastName, "
38     Last_name_mismatch_in_database")
39
40 assert.NotEqual(t, "newpassword123", fetchedUser.Password, "
41     Password_should_be_hashed_in_database")
42 }
```

Код 4.10: Test Update User Function

4.3 Front-End хэрэгжүүлэлт

Энэхүү төсөл нь Next.js фреймворкийг ашиглан хэрэглэгчийн бүртгэлд засвар хийх хуудас үүсгэсэн хэсгийг харуулж байна. Хэрэглэгчийн өгөгдлийг хүлээн авч, сервер рүү илгээх, мөн серверээс хариу авах процессыг оролцуулан бичигдсэн. Үндсэн зорилго нь хэрэглэгчийн бүртгэл болон мэдээллийн санах оруулах үндсэн үйл ажиллагааг вэб хуудас дээр хэрэгжүүлэх явдал юм.

Сурах ур чадвар:

- Next.js -ийн талаар үндсэн ойлголт авах
- API-тай харьцах
- Формыг хянаж алдаатай тохиолдолд хэрэглэгчид мэдэгдэх
- UI/UX сайжруулалт
- State Management ба өгөгдлийн хөрвүүлэлт хувийн мэдээлэл болон огноо зэргийг хэрэглэгчид ойлгогдохуйц хөрвүүлэх, хадгалах
- Аюулгүй байдал, баталгаажуулалт token ашиглан хамгаалалттай API хүсэлт илгээх

Формын шаардлага:

- Next.js ашигласан байх
- Хэрэглэгчийн мэдээлэл удирдах
- Форм хэлбэртэй байх
- API хүсэлт илгээдэг байх
- Token ашиглан хамгаалалттай өгөгдөл илгээх
- Хэрэглэгчийн үүргийг тохируулах

4.3.1 Front-end-ийн ерөнхий бүх элементийг агуулсан хэсэг болох формд засвар хийх код

Хэрэглэгчийн мэдээллийг хадгалах интерфейс

```
1 interface UserData {  
2   first_name: string;  
3   last_name: string;  
4   login_id: string;  
5   email_work: string;  
6   email_personal: string;  
7   phone_number_work: string;  
8   phone_number_personal: string;  
9   is_active: boolean;  
10  active_start_date: string;  
11  active_end_date: string;  
12  password?: string;  
13  created_by: string;  
14  updated_by: string;  
15 }
```

Код 4.11: UserData интерфейсийг үүсгэсэн байдал

Энэ функц нь хэрэглэгчийн мэдээллийг нэмэх, засах формыг харуулна

```
1 export default function UserFormPage() {}
```

Код 4.12: UserFormPage функц (Main component)

Доорх функцууд UserFormPage дотор агуулагдах функцууд. Эхлээд хувьсагчаа тодорхойлж өгнө.

```
1 const searchParams = useSearchParams();  
2 const id = searchParams.get("id");  
3 const router = useRouter();
```



```
4  const { data: session } = useSession();
```

Код 4.13: Хувьсагч

UserFormData тохируулж өгнө. Created by болон updated by нь нэвтэрсэн хэрэглэгчийн id-гаар автоматаар бөглөгдөнө.

```
1  const initialFormData: UserFormData = {
2    first_name: "",
3    last_name: "",
4    login_id: "",
5    email_work: "",
6    email_personal: "",
7    phone_number_work: "",
8    phone_number_personal: "",
9    is_active: false,
10   active_start_date: "",
11   active_end_date: "",
12   password: "",
13   created_by: session?.user?.id || "",
14   updated_by: session?.user?.id || "",
15 };
```

Код 4.14: initialFormData хоосон утга оноосон байдал

Төлөв хадгалах

```
1  const [formData, setFormData] = useState<UserFormData>(
    initialFormData);
2  const [loading, setLoading] = useState(false);
3  const [errors, setErrors] = useState<Record<string, string>>({});
```

Код 4.15: Төлөв хадгалах

Хэрэглэгчийн мэдээллийг татах

```
1  useEffect(() => {
2      const fetchUserData = async () => {
3          if (!id || !session?.user?.token) return;
4
5          try {
6              setLoading(true);
7              const response = await req.GET(
8                  `/admin/users?id=${id}`,
9                  session.user.token
10             );
11             setFormData({
12                 ...response,
13                 updated_by: session.user.id || "",
14             });
15         } catch (error) {
16             toast({
17                 title: "",
18                 description: "          ",
19                 variant: "destructive",
20             });
21         } finally {
22             setLoading(false);
23         }
24     };
25
26     fetchUserData();
27 }, [id, session?.user?.id, session?.user?.token]);
```

Код 4.16: Хэрэглэгчийн мэдээлэл татах useEffect

Оролтын өөрчлөлт хянах handleChange функц

```
1  const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {  
2    const { name, value, type, checked } = e.target;  
3    setFormData((prev) => ({  
4      ...prev,  
5      [name]: type === "checkbox" ? checked : value,  
6    }));  
7  };
```

Код 4.17: handleChange

Хүсэлт илгээх handleSubmit функц

```
1  const handleSubmit = async (e: React.FormEvent) => {  
2    e.preventDefault();  
3  
4    try {  
5      setLoading(true);  
6      const response = await req.PUT(  
7        `/admin/users/${id || ""}`,  
8        session?.user?.token || "",  
9        {  
10         ...formData,  
11         active_start_date: formData.active_start_date || null,  
12         active_end_date: formData.active_end_date || null,  
13       }  
14     );  
15  }
```

```

16     if (response) {
17         toast({
18             title: "      ",
19             description: "      □      ",
20         });
21     }
22 } catch (error) {
23     console.error("Error□submitting□form:", error);
24     toast({
25         title: "      ",
26         description: "      □ □      ",
27         variant: "destructive",
28     });
29 } finally {
30     setLoading(false);
31 }
32 };

```

Код 4.18: handleSubmit

Формын UI бүтэц

```

1  return (
2      <div className="py-8□px-4">
3          <Card>
4              <CardHeader>
5                  <CardTitle className="text-2xl">
6                      {id ? "      □      " : "      □      □      "}
7                  </CardTitle>
8              </CardHeader>

```

```

9      <CardContent>
10      <form onSubmit={handleSubmit} className="space-y-4">
11      <div className="space-y-6">
12      <h3 className="text-lg font-medium" > </h3>
13      <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
14      <div>
15      <Label htmlFor="first_name" ></Label>
16      <Input
17      id="first_name"
18      name="first_name"
19      type="text"
20      value={formData.first_name}
21      onChange={handleChange}
22      className={errors.first_name ? "border-red-500" : "
      "
      "}/>
23      />
24      {errors.first_name && (
25      <p className="text-red-500 text-sm mt-1">
26      {errors.first_name}
27      </p>
28      )}
29      </div>
30      <div>
31      <Label htmlFor="last_name" ></Label>
32      <Input
33      id="last_name"
34      name="last_name"
35      type="text"

```

```

36         value={formData.last_name}
37         onChange={handleChange}
38         className={errors.last_name ? "border-red-500" : ""}
39     }
40 />
41 {errors.last_name && (
42     <p className="text-red-500 text-sm mt-1">
43         {errors.last_name}
44     </p>
45 )}
46 </div>
47 <div>
48     <Label htmlFor="login_id"    > ID</Label>
49     <Input
50         id="login_id"
51         name="login_id"
52         type="text"
53         value={formData.login_id}
54         onChange={handleChange}
55         className={errors.login_id ? "border-red-500" : ""}
56     />
57 {errors.login_id && (
58     <p className="text-red-500 text-sm mt-1">
59         {errors.login_id}
60     </p>
61 )}
62 </div>
63 {!id && (

```

```

63         <div>
64             <Label htmlFor="password" > </Label>
65             <Input
66                 id="password"
67                 name="password"
68                 type="password"
69                 value={formData.password}
70                 onChange={handleChange}
71                 className={errors.password ? "border-red-500" : "
72                     "}
73             />
74             {errors.password && (
75                 <p className="text-red-500 text-sm mt-1">
76                     {errors.password}
77                 </p>
78             )}
79         </div>
80     )}
81 </div>
82 <div className="space-y-2">
83     <h3 className="text-lg font-medium" > </h3>
84     <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
85         <div>
86             <Label htmlFor="email_work" > </Label>
87             <Input
88                 id="email_work"
89                 name="email_work"

```

```

90         type="email"
91         value={formData.email_work}
92         onChange={handleChange}
93         className={errors.email_work ? "border-red-500" : "
          "}
94     />
95     {errors.email_work && (
96         <p className="text-red-500 text-sm mt-1">
97             {errors.email_work}
98         </p>
99     )}
100 </div>
101 <div>
102     <Label htmlFor="email_personal" > </Label>
103     <Input
104         id="email_personal"
105         name="email_personal"
106         type="email"
107         value={formData.email_personal}
108         onChange={handleChange}
109         className={errors.email_personal ? "border-red-500"
          : ""}
110     />
111     {errors.email_personal && (
112         <p className="text-red-500 text-sm mt-1">
113             {errors.email_personal}
114         </p>
115     )}

```



```

116         </div>
117     <div>
118         <Label htmlFor="phone_number_work" > </Label>
119         <Input
120             id="phone_number_work"
121             name="phone_number_work"
122             type="tel"
123             value={formData.phone_number_work}
124             onChange={handleChange}
125         />
126     </div>
127     <div>
128         <Label htmlFor="phone_number_personal" > </Label>
129         <Input
130             id="phone_number_personal"
131             name="phone_number_personal"
132             type="tel"
133             value={formData.phone_number_personal}
134             onChange={handleChange}
135         />
136     </div>
137 </div>
138 </div>
139 <div className="space-y-2">
140     <h3 className="text-lg font-medium" ></h3>
141     <div className="grid grid-cols-1 md:grid-cols-2 gap-4
142         items-center">

```

```

143     <input
144         id="is_active"
145         name="is_active"
146         type="checkbox"
147         checked={formData.is_active}
148         onChange={handleChange}
149         className="h-4 w-4 border-gray-300 rounded"
150     />
151     <Label htmlFor="is_active" > </Label>
152 </div>
153 <div className="space-x-2">
154     <Label > :</Label>
155     <DatePickerWithRange
156         value={{
157             from: formData.active_start_date ? new Date(
158                 formData.active_start_date) : undefined,
159             to: formData.active_end_date ? new Date(formData.
160                 active_end_date) : undefined,
161         }}
162         onChange={(date) => {
163             setFormData((prev) => ({
164                 ...prev,
165                 active_start_date: date?.from ? date.from.
166                     toISOString() : "",
167                 active_end_date: date?.to ? date.to.toISOString
168                     () : "",
169             }));
170         }}
171     />

```

```
167         />
168     </div>
169 </div>
170 </div>
171 <UserRole userId={id!} />
172 <div>
173     <Button type="submit" className="w-full" disabled={
174         {loading ? " " : id ? " " : " "}
175     >
176     </Button>
177 </div>
178 </form>
179 </CardContent>
180 </Card>
181 </div>
182 );
```

Код 4.19: UI бүтэц

4.4 Back-End хэрэгжүүлэлт

Энэ код нь Gin framework ашиглан хэрэглэгчийн бүртгэл, шинэчлэл, устгал, шалгалт хийх RESTful API-г зохион байгуулсан бөгөөд лог бичлэг болон аль болох тодорхой мэдээлэл авах боломжтой Controller бичих зорилготой.

Сурах ур чадвар:

- Golang хэлний мэдлэг
- Gin framework ашиглах
- GORM ORM ашиглах
- Өгөгдлийн сангийн зохицуулалт хийх
- Hashing ба нууцлал
- Алдаа удирдах
- Лог хөтлөх
- HTTP хүсэлттэй ажилллах
- Кодын бүтэц зохион байгуулалт

Кодны шаардлага:

- Golang хэлний синтакс, өгөгдлийн төрөл, функцийн ажиллах зарчим мэдэх
- REST API зохион байгуулах, маршрутын менежмент хийх чадвартай байх.
- GORM ашиглан өгөгдлийн сангийн хүсэлтүүд боловсруулах чадвартай байх.
- CRUD үйлдлүүдийг бүтээх
- JSON өгөгдлийг боловсруулах

- Нууц үгийн хашлалат, хэрэглэгчийн мэдээллийг хамгаалах
- API-д гарч болох алдааг зөв удирдаж, тохирсон HTTP код буцаах чадвартай байх
- Controller давхаргыг зөв зохион байгуулах
- API тест хийх

4.4.1 Controller

Энэ хэсэгт хэрэглэгчийн мэдээлэлтэй холбоотой CRUD үйлдлүүдийг GORM ашиглан хийнэ. CRUD(create, Read, Update, Delete) үйлдлүүд нь хэрэглэгчийн мэдээллийг үүсгэх, унших, шинэчлэх, устгах боломжийг олгодог.

4.4.2 GetAllUsers

Хэрэглэгчийн мэдээллийг Postgres өгөгдлийн сангаас авах API-ийг хэрэгжүүлэх хэсэг. GetAllUsers функц нь өгөгдлийн сангаас бүх хэрэглэгчийн мэдээллийг авахад зориулагдсан бөгөөд хэрэглэгчийн жагсаалтыг буцаадаг.

```
1 func CreateUser(c *gin.Context) {
2     var user models.User
3     if err := c.ShouldBindJSON(&user); err != nil {
4         log.Error("Failed to bind user JSON", map[string]interface{}{"error":
5             err.Error()}, c)
6         c.JSON(http.StatusBadRequest, gin.H{"error": "Invalid request payload"})
7         return
8     }
9     log.Info("Creating user", map[string]interface{}{"user": user}, c)
10
11     newUser, err := services.CreateUser(user, c.Request)
12     if err != nil {
13         log.Error("Failed to create user", map[string]interface{}{"user":
14             user, "error": err.Error()}, c)
15         c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
16         return
17     }
18 }
```

```

16 }
17
18 log.Info("User created successfully", map[string]interface{}{"userID"
    : newUser.ID}, c)
19 c.JSON(http.StatusCreated, newUser)
20 }

```

Код 4.20: CreateUser Controller

4.4.3 CreateUser

CreateUser функц нь хэрэглэгчийн мэдээллийг JSON хэлбэрээр хүлээн авч, models.User объектоор хувиргадаг. Ингэснээр хэрэглэгчийн мэдээллийг сервер рүү илгээх боломжтой болно.

```

1 func GetAllUsers(r *http.Request) ([]models.User, error) {
2     var users []models.User
3     if err := config.DB.Find(&users).Error; err != nil {
4         log.Error("Failed to fetch users", map[string]interface{}{"error"
            : err.Error()}, r)
5         return nil, err
6     }
7     return users, nil
8 }

```

Код 4.21: CreatUser Controller

4.4.4 UpdateUser

UpdateUser функц нь хэрэглэгчийн ID болон шинэ мэдээллийг хүлээн авч, models.User объектоор шинэчилнэ.

```
1 func UpdateUser(c *gin.Context) {
2     id := c.Param("id")
3     var user models.User
4
5     if err := c.ShouldBindJSON(&user); err != nil {
6         log.Error("Failed to bind user JSON", map[string]interface{}{"error":
7             err.Error()}, c)
8         c.JSON(http.StatusBadRequest, gin.H{"error": "Invalid request
9             payload"})
10        return
11    }
12
13    log.Info("Updating user", map[string]interface{}{"userID": id, "user":
14        user}, c)
15    updatedUser, err := services.UpdateUser(id, user, c.Request)
16    if err != nil {
17        log.Error("Failed to update user", map[string]interface{}{"userID":
18            id, "error": err.Error()}, c)
19        c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed to
20            update user"})
21        return
22    }
23
24    log.Info("User updated successfully", map[string]interface{}{"userID":
25        updatedUser.ID}, c)
26    c.JSON(http.StatusOK, updatedUser)
27 }
```


Код 4.22: UpdateUser Controller

4.4.5 DeleteUser

DeleteUser функц нь хэрэглэгчийн ID-ийг URL параметрээс авч, тухайн хэрэглэгчийг устгах үйлдлийг гүйцэтгэдэг.

```
1 func DeleteUser(c *gin.Context) {
2     id := c.Param("id")
3
4     if err := services.DeleteUser(id, c.Request); err != nil {
5         log.Error("Failed to delete user", map[string]interface{}{"userID": id, "error": err.Error()}, c)
6         c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed to delete user"})
7         return
8     }
9
10    log.Info("User deleted successfully", map[string]interface{}{"userID": id}, c)
11    c.JSON(http.StatusOK, gin.H{"message": "User deleted successfully"})
12 }
```

Код 4.23: DeleteUser Controller

4.4.6 CheckLoginIDExists

CheckLoginIDExists функц нь хэрэглэгчийн login id-ийг URL-ийн query параметрээс авч, энэ login id-ийн оршин байгаа эсэхийг шалгадаг.

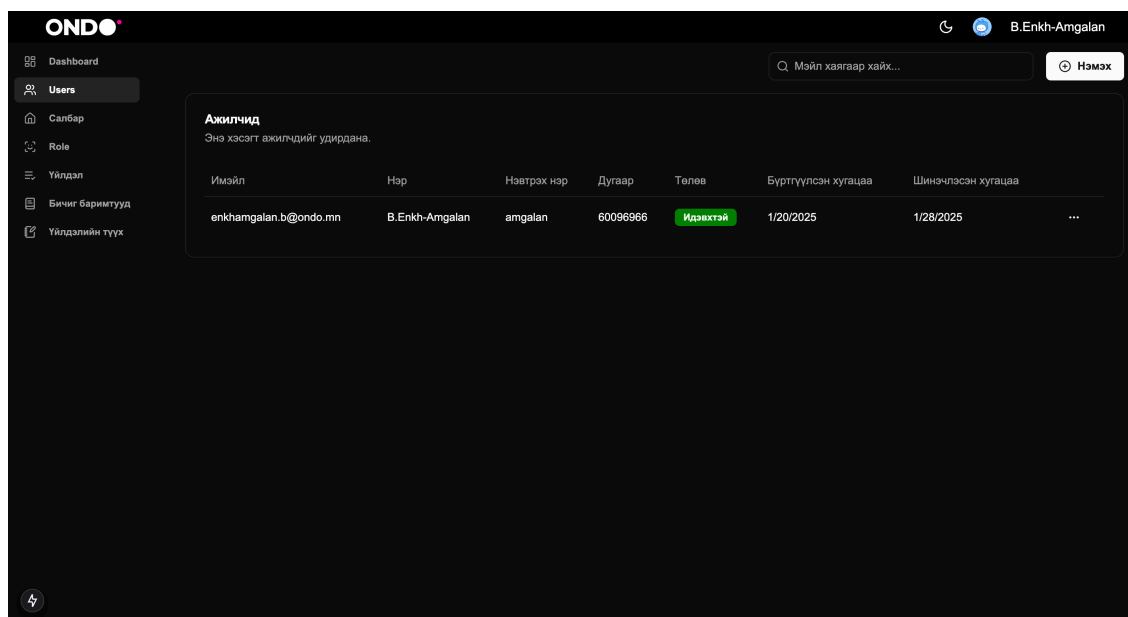
```
1 func CheckLoginIDExists(c *gin.Context) {
2     loginID := c.Query("login_id")
3     if loginID == "" {
4         log.Error("login_id_is_required", map[string]interface{}{}, c)
5         c.JSON(http.StatusBadRequest, gin.H{"error": "login_id_is_required"
6             })
7         return
8     }
9
10    exists, err := services.CheckLoginIDExists(loginID, c)
11    if err != nil {
12        log.Error("Failed_to_check_login_id", map[string]interface{}{"error
13            ": err.Error()}, c)
14        c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed_to_
15            check_login_id"})
16        return
17    }
18
19    log.Info("Checked_login_id_existence", map[string]interface{}{"
20        login_id": loginID, "exists": exists}, c)
21    c.JSON(http.StatusOK, gin.H{"exists": exists})
22 }
```

Код 4.24: CheckLoginIDExists Controller

5. ДҮГНЭЛТ

5.1 Үр дүн

5.1.1 *Front харагдах байдал*



Зураг 5.1: Хэрэглэгчидын жагсаалт

The screenshot shows a user registration form titled "Хэрэглэгч засах" (Edit User) in the ONDO system. The form is divided into several sections: "Хувийн мэдээлэл" (Personal Information), "Холбоо барих" (Contact Information), and "Төлөв" (Status). The "Хувийн мэдээлэл" section includes fields for Name (Enkh-Amgalan), Surname (Bayanjargal), and Username (amgalan). The "Холбоо барих" section includes fields for Email (enkh-amgalan.b@ondo.mn), Mobile Number (60096966), and Password (amgalan.dbb@gmail.com). The "Төлөв" section includes a checkbox for "Идэвхтэй эсэх" (Active) and a date range for "Ажиллах хугацаа" (Working Period) from 2024-11-11 to 2025-11-11. A "Засах" (Save) button is at the bottom right.

Хэрэглэгч засах

Хувийн мэдээлэл

Нэр: Enkh-Amgalan

Овог: Bayanjargal

Нэвтрэх ID: amgalan

Холбоо барих

Ажлын имэйл: enkh-amgalan.b@ondo.mn

Хувийн имэйл: amgalan.dbb@gmail.com

Ажлын дугаар: 60096966

Хувийн дугаар: 95560899

Төлөв

☒ Идэвхтэй эсэх

Ажиллах хугацаа: 2024-11-11 - 2025-11-11

Ажилтаны эрх

Засах

Зураг 5.2: Хэрэглэгч засах

This screenshot is identical to the previous one, but it includes a calendar widget for date selection. The calendar shows November 2024 and December 2024. The date 11 is selected in November. The "Засах" (Save) button is at the bottom right.

Хэрэглэгч засах

Хувийн мэдээлэл

Нэр: Enkh-Amgalan

Овог: Bayanjargal

Нэвтрэх ID: amgalan

Холбоо барих

Ажлын имэйл: enkh-amgalan.b@ondo.mn

Ажлын дугаар: 60096966

Төлөв

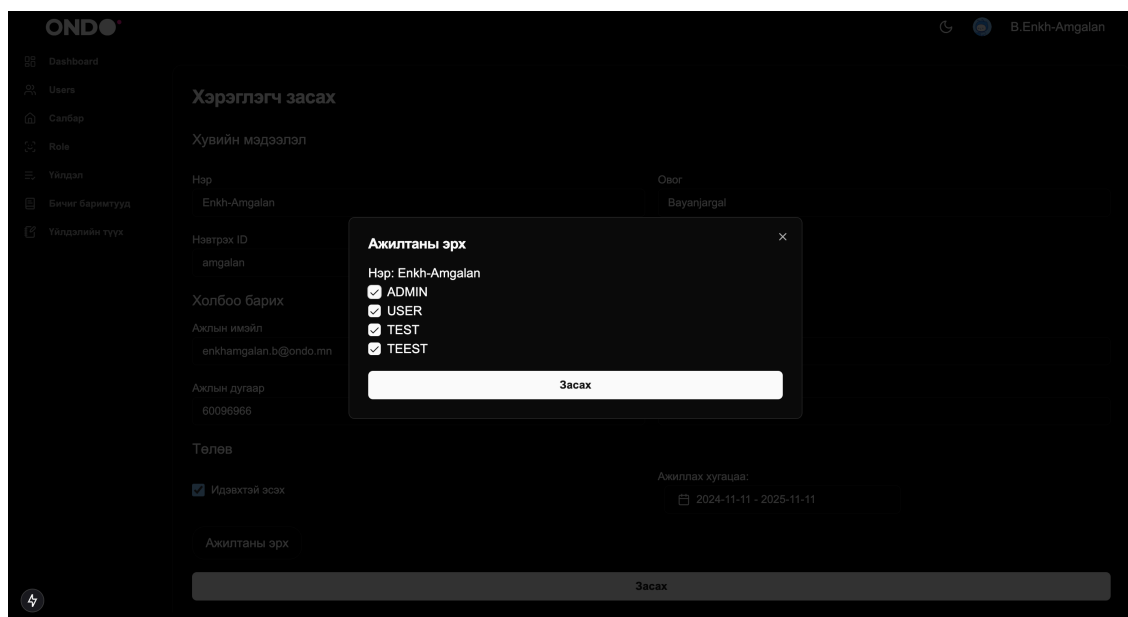
☒ Идэвхтэй эсэх

Ажиллах хугацаа: 2024-11-11 - 2025-11-11

Ажилтаны эрх

Засах

Зураг 5.3: Өдөр сонгох календар



Зураг 5.4: Ажилтны эрх сонгох хэсэг

5.2 Үр дүнгийн тайлан

Миний бие Хуур Мюзик Групп ХХК-д 21 хоногийн хугацаатай мэргэжлийн дадлагыг амжилттай гүйцэтгэж дуусгалаа. Уг хугацаанд хичээлийн хүрээнд үзсэн онолын ойлголтуудыг практик дээр туршиж, хэрэгжүүлсэн ба хөгжүүлэлт голчилсон технологийн компанийн ерөнхий үйл ажиллагаа, баг хооронд зохицон ажиллах чадвар, хөгжүүлэлтийн шинэ арга барилуудыг амжилттай эзэмшсэн гэж дүгнэж байна.

Continuous Integration/Continuous Deployment, GIT дээрх Feature Branch, ашиглаж буй програмчлалын хэлнийхээ давуу талыг судлан уг хэлээрээ сэтгэж бичих, том асуудлыг олон болгон хувааж багаар, алхам дэс дараатай асуудлыг шийдвэрлэх мөн ашиглаж буй сан, технологийнхоо гарын авлага буюу documentation-тай илүү сайн танилцаж уг технологийнхоо цаана нь буй концептийг хялбараар ойлгох гэх мэт чадваруудыг эзэмшсэн. Үүнийгээ цаашид илүү хөгжүүлж мэргэшсэн фронт-энд хөгжүүлэгч болохоор зорьж байна. Дадлага хийсэн компани маань хэрэгжүүлж буй төсөлдөө үргэлж технологийн шинэ туршилтын хувилбаруудыг төвөгшөөлгүйгээр хэрэглэж, түүнийхээ алдааг илрүүлж, ажлын бус цагаараа хамтдаа шийдлийг хайж олон улсын

нээлттэй эхийн төсөлд гар бие оролцдог нь бусад компаниудаас онцлог. Үүний үр дүнд манай дадлагын удирдагч болох С. Дөлмандах нь React-Native-н core contributor болж, 2019 онд болсон React-Native EU гэх олон улсын хөгжүүлэгчдийн эвентэд илтгэл тавьж байсан удаатай. Би цаашид өөрийн чөлөөт цагаа ашиглан дотоодын болон олон улсын нээлттэй эхийн төсөлд хувь нэмрээ оруулж гадны чадварлаг хөгжүүлэгчдийн арга барил, код бичих туршлага, тухайн асуудлыг хэрхэн шийдсэн гэх мэт үнэтэй мэдлэгүүдийг хуримтлуулж бусад хүмүүст мөн нээлттэй эхийн төсөлд оролцохын давуу талуудыг танилцуулж уриалахаар төлөвлөсөн байгаа.

Дадлагын эхэн үед React болон Next.js технологийн талаар судлах, түүнийгээ хэрэгжүүлэх, хөгжүүлэлтийн арга барилуудтай танилцах зорилготой байсан ба цаашид мэдээлэл технологийн ямар чиглэлээр мэргэшиж, түүндээ хүрэхийн тулд хэрхэн чадварлаг болох ёстойг ойлгосон тул зорилгодоо бүрэн хүрсэн гэдэгт итгэлтэй байна.

Bibliography

- [1] Declarative програмчлал болон Imperative програмчлалын ялгаа
<https://codeburst.io/declarative-vs-imperative-programming-a8a7c93d9ad2>
- [2] Material-ui Beta v5 хувилбарын Card ашиглах заавар
<https://next.material-ui.com/api/card/>

А. УДИРДАГЧИЙН ҮНЭЛГЭЭ

Хавсралт 3

МУИС, ХШУИС-ИЙН МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

Програм хамгаалчийн АНГИЙН
ОЮУТАН Б.Бат-Өлзий -ЫН
ДАДЛАГЫН АЖЛЫН УДИРДАГЧИЙН ТОДОРХОЙЛОЛТ

2021 оны .. сарын ..

Програм хамгаалчийн ангийн 1861num3474 кодтой оюутан Б.Бат-Өлзий нь манай байгууллагад 2021 оны 06 сарын 02-ны өдрөөс 06 сарын 06-ны өдөр хүртэл мэргэшүүлэх дадлагыг батлагдсан удирдамж, ажлын төлөвлөгөөний дагуу гүйцэтгэлээ. Оюутан Б.Бат-Өлзий-ын удирдамжийн дагуу дадлагын ажлыг гүйцэтгэсэн байдал:

Б.Бат-Өлзий нь үйлдвэрлэлийн дадлагын хүрээнд React.js болон Next.js ашиглан хөгжүүлж байгаа вэб програм хангамжийн хөгжүүлэлтэд оролцсон. Өмнө нь React болон холбоотой технологиудыг ашиглаж байсан туршлагагүй тул судлах, суралцах зүйлс цөөнгүй бөгөөд өгсөн чиглэл, даалгаврын дагуу хичээл зүтгэлтэй, цаг тухай бүрт судалж бас суралцсан болно. Тэрээр React.js компонент болон hooks бичих зарчим, арга ажиллагаа мөн Material UI ашиглан хэрэглэгчийн интерфэйсийг бүтээхэд суралцаж бас хөгжүүлэлтэд оролцсон. Үүнээс гадна Continuous Integration хөгжүүлэлтийн аргачлал болон хэрэглэгчийн интерфэйсийн автоматжуулсан тест бичих зарчимтай танилцсан.

Бат-Өлзий нь өгсөн даалгаврыг цаг тухай бүрт нь биелүүлдэг, нээлттэй харилцаатай тул ойлгохгүй эсвэл чадахгүй байгаа зүйлээ чөлөөтэй асуудаг, тусламж авдаг давуу талтай, мөн UX сонирхдог тул бүтээгдэхүүний хэрэглэгчийн интерфэйс бүтээх ажилд бүтээлчээр оролцож олон үнэтэй санал өгч байсан. Дадлагын хугацаанд вэб хөгжүүлэлтийн тал дээр багагүй өсч хөгжсөн, туршлагатай болсон гэж дүгнэж байна.

Үнэлгээний санал: 10 оноо

Дадлагын удирдагч

Дадлагын удирдагчийн талаарх мэдээлэл:

В. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

В.1 GORM

В.1.1 Golang дээр Gorm ашигласан байдал

```
1 package services
2
3 import (
4     "branch-admin-service/config"
5     "branch-admin-service/log"
6     "branch-admin-service/models"
7     "branch-admin-service/utils"
8     "errors"
9     "net/http"
10
11     "github.com/gin-gonic/gin"
12     "gorm.io/gorm"
13 )
14
15 func GetAllUsers(r *gin.Context) ([]models.User, error) {
16     var users []models.User
17     if err := config.DB.Find(&users).Error; err != nil {
18         log.Error("Failed to fetch users", map[string]interface{}{"error":
19             : err.Error()}, r)
20         return nil, err
21     }
22     return users, nil
23 }
24
25 func CreateUser(user models.User, r *http.Request) (models.User,
26     error) {
27     existingUser := config.DB.Where("login_id=?", user.LoginID).First
28     (&models.User{})
29     if existingUser.RowsAffected > 0 {
30         return models.User{}, errors.New("user already exists")
31     }
32
33     hashedPassword, err := utils.HashPassword(user.Password)
34     if err != nil {
35         return models.User{}, err
36     }
37     user.Password = hashedPassword
38
39     if err := config.DB.Create(&user).Error; err != nil {
40         return models.User{}, err
41     }
42     return user, nil
43 }
```

```

42 func UpdateUser(id string, user models.User, r *http.Request) (models
    .User, error) {
43     if err := config.DB.Model(&user).Where("id=?", id).Updates(user).
        Error; err != nil {
44         return models.User{}, err
45     }
46     return user, nil
47 }
48
49 func DeleteUser(id string, r *http.Request) error {
50     if err := config.DB.Where("user_id=?", id).Delete(&models.
        UserRole{}).Error; err != nil {
51         return err
52     }
53     if err := config.DB.Where("id=?", id).Delete(&models.User{}).
        Error; err != nil {
54         return err
55     }
56     return nil
57 }
58
59 func GetUserByID(id string, r *gin.Context) (*models.User, error) {
60     log.Info("Fetching user by ID from the database", map[string]
        interface{}{"id": id}, r)
61
62     var user models.User
63     err := config.DB.Where("id=?", id).First(&user).Error
64     if err != nil {
65         if errors.Is(err, gorm.ErrRecordNotFound) {
66             log.Warn("User not found", map[string]interface{}{"id": id}, r)
67             return nil, nil
68         }
69         log.Error("Database query error", map[string]interface{}{"error":
            err.Error()}, r)
70         return nil, err
71     }
72
73     log.Info("User fetched successfully", map[string]interface{}{"id":
        id}, r)
74     return &user, nil
75 }
76
77 func CheckLoginIDExists(loginID string, r *gin.Context) (bool, error)
    {
78     var user models.User
79     result := config.DB.Where("login_id=?", loginID).First(&user)
80
81     if result.Error != nil {
82         if errors.Is(result.Error, gorm.ErrRecordNotFound) {
83             return false, nil
84         }

```

```

85     log.Error("Failed to check login_id", map[string]interface{}{"
86         error": result.Error.Error()}, r)
87     return false, result.Error
88 }
89 return true, nil
90 }

```

B.2 Тест

B.2.1 Ашигласан GORM дээрээ тест бичсэн байдал

```

1  package service_test
2
3  import (
4      "log"
5      "net/http"
6      "net/http/httptest"
7      "testing"
8      "time"
9
10     "branch-admin-service/config"
11     "branch-admin-service/models"
12     services "branch-admin-service/services/admin"
13     test "branch-admin-service/test/utils"
14
15     "github.com/stretchr/testify/assert"
16     "gorm.io/gorm"
17 )
18
19 func setupTestDB() *gorm.DB {
20     test.ConnectTestDatabase()
21     config.DB.Exec("TRUNCATE TABLE users RESTART IDENTITY CASCADE")
22     return config.DB
23 }
24
25
26 func dropExistingUser(db *gorm.DB, loginID string) {
27     var existingUser models.User
28     if err := db.Where("login_id=?", loginID).First(&existingUser).
29         Error; err == nil {
30         if err := db.Delete(&existingUser).Error; err != nil {
31             log.Fatal("Error deleting existing user:", err)
32         }
33     }
34 }
35
36 func createSampleUser(loginID string) models.User {
37     dummyDate := time.Date(2021, time.January, 1, 0, 0, 0, time.UTC)
38     emailPersonal := "personal@mail.com"

```

```

38     phoneNumberWork := "9876543210"
39     admin := "admin"
40
41     return models.User{
42         FirstName:      "John",
43         LastName:       "Doe",
44         LoginID:        loginID,
45         EmailWork:      "test@mail.com",
46         EmailPersonal:   &emailPersonal,
47         PhoneNumberWork: &phoneNumberWork,
48         PhoneNumberPersonal: &phoneNumberWork,
49         IsActive:       true,
50         ActiveStartDate: dummyDate,
51         ActiveEndDate:  &dummyDate,
52         Password:       "password123",
53         CreatedAt:      dummyDate,
54         UpdatedAt:      dummyDate,
55         CreatedBy:      &admin,
56         UpdatedBy:      &admin,
57     }
58 }
59
60 func TestCreateUser(t *testing.T) {
61     db := setupTestDB()
62     defer test.CloseTestDatabase()
63
64     loginID := "johndoe"
65     dropExistingUser(db, loginID)
66
67     r := httptest.NewRequest(http.MethodPost, "/", nil)
68     user := createSampleUser(loginID)
69
70     createdUser, err := services.CreateUser(user, r)
71
72     assert.NoError(t, err, "Error creating user")
73     assert.NotZero(t, createdUser.ID, "User ID should not be zero")
74     assert.NotEqual(t, "password123", createdUser.Password, "Password should be hashed")
75
76     var fetchedUser models.User
77     err = db.First(&fetchedUser, "id=?", createdUser.ID).Error
78     assert.NoError(t, err, "Error fetching user")
79
80     assert.Equal(t, "John", fetchedUser.FirstName, "First name mismatch")
81     assert.Equal(t, "Doe", fetchedUser.LastName, "Last name mismatch")
82     assert.NotEqual(t, "password123", fetchedUser.Password, "Password should be hashed")
83 }
84
85 func TestGetAllUsers(t *testing.T) {
86     db := setupTestDB()

```

```

87     defer test.CloseTestDatabase()
88
89     db.Exec("TRUNCATE TABLE users RESTART IDENTITY CASCADE")
90
91     users := []models.User{
92         {FirstName: "John", LastName: "Doe"},
93         {FirstName: "Jane", LastName: "Smith"},
94     }
95     for _, user := range users {
96         db.Create(&user)
97     }
98
99     r := httptest.NewRequest(http.MethodGet, "/users", nil)
100    allUsers, err := services.GetAllUsers(r)
101
102    assert.NoError(t, err, "Error fetching all users")
103    assert.Len(t, allUsers, len(users), "Unexpected number of users fetched")
104
105    for i, fetchedUser := range allUsers {
106        assert.True(t, i < len(users), "Index %d out of range for expected users", i)
107        assert.Equal(t, users[i].FirstName, fetchedUser.FirstName,
108            "First name mismatch. Expected: %s, Got: %s", users[i].
109                FirstName, fetchedUser.FirstName)
110        assert.Equal(t, users[i].LastName, fetchedUser.LastName,
111            "Last name mismatch. Expected: %s, Got: %s", users[i].LastName,
112                fetchedUser.LastName)
113    }
114
115    }
116
117    func TestUpdateUser(t *testing.T) {
118        db := setupTestDB()
119        defer test.CloseTestDatabase()
120
121        // Create sample user
122        originalUser := createSampleUser("johndoe")
123        db.Create(&originalUser)
124
125        // Prepare updated user data
126        updatedUserData := models.User{
127            FirstName: "Jane",
128            LastName: "Smith",
129            Password: "newpassword123",
130        }
131
132        r := httptest.NewRequest(http.MethodPut, "/user", nil)
133        updatedUser, err := services.UpdateUser(originalUser.ID,
            updatedUserData, r)

```

```

134     assert.NoError(t, err, "Error Updating user")
135     assert.Equal(t, updatedUserData.FirstName, updatedUser.FirstName, "
        First_name_mismatch")
136     assert.Equal(t, updatedUserData.LastName, updatedUser.LastName, "
        Last_name_mismatch")
137     assert.NotEqual(t, "newpassword123", updatedUser.Password, "
        Password_should_be_hashed")
138
139     var fetchedUser models.User
140     err = db.First(&fetchedUser, "id=?", originalUser.ID).Error
141     assert.NoError(t, err, "Error fetching updated user")
142
143     assert.Equal(t, updatedUserData.FirstName, fetchedUser.FirstName, "
        First_name_mismatch_in_database")
144     assert.Equal(t, updatedUserData.LastName, fetchedUser.LastName, "
        Last_name_mismatch_in_database")
145     assert.NotEqual(t, "newpassword123", fetchedUser.Password, "
        Password_should_be_hashed_in_database")
146 }
147
148 func TestDeleteUser(t *testing.T) {
149     db := setupTestDB()
150     defer test.CloseTestDatabase()
151
152     // Create sample user
153     user := createSampleUser("johndoe")
154     db.Create(&user)
155
156     r := httptest.NewRequest(http.MethodDelete, "/user", nil)
157     err := services.DeleteUser(user.ID, r)
158
159     assert.NoError(t, err, "Error deleting user")
160
161     var fetchedUser models.User
162     err = db.Where("id=?", user.ID).First(&fetchedUser).Error
163     assert.Error(t, err, "Expected error for fetching deleted user")
164     assert.Equal(t, gorm.ErrRecordNotFound, err, "Expected record not
        found error for deleted user")
165 }
166
167
168 func TestGetUserByID(t *testing.T) {
169     db := setupTestDB()
170     defer test.CloseTestDatabase()
171
172     db.Exec("TRUNCATE TABLE users RESTART IDENTITY CASCADE")
173
174     sampleUser := models.User{FirstName: "John", LastName: "Doe", ID: "
        123"}
175     db.Create(&sampleUser)
176
177     t.Run("Valid ID", func(t *testing.T) {

```

```

178     r := httptest.NewRequest(http.MethodGet, "/user/123", nil)
179     user, err := services.GetUserByID("123", r)
180
181     assert.NoError(t, err, "Error_fetching_user_with_valid_ID")
182     assert.NotNil(t, user, "User_should_not_be_nil_for_valid_ID")
183     assert.Equal(t, sampleUser.ID, user.ID, "User_ID_mismatch")
184     assert.Equal(t, sampleUser.FirstName, user.FirstName, "User_first
        _name_mismatch")
185 })
186
187 t.Run("Invalid_ID", func(t *testing.T) {
188     r := httptest.NewRequest(http.MethodGet, "/user/999", nil)
189     user, err := services.GetUserByID("999", r)
190
191     // Handle the error cleanly
192     assert.NoError(t, err, "Error_should_be_nil_for_non-existent_user
        _ID")
193     assert.Nil(t, user, "Fetched_user_should_be_nil_for_non-existent_
        ID")
194 })
195 }

```

B.3 Front-end

B.3.1 /admin/user/edit/page.tsx

```

1     "use_client";
2     import { useEffect, useState } from "react";
3     import { useRouter, useSearchParams } from "next/navigation";
4     import { Input } from "@components/ui/input";
5     import { Button } from "@components/ui/button";
6     import { Label } from "@components/ui/label";
7     import { toast } from "@hooks/use-toast";
8     import { Card, CardHeader, CardTitle, CardContent } from "@/
        components/ui/card";
9     import { req } from "@app/api";
10    import { useSession } from "next-auth/react";
11    import { DatePickerWithRange } from "@components/date-range-picker";
12    import UserRole from "../components/user-role";
13
14    interface UserFormData {
15        first_name: string;
16        last_name: string;
17        login_id: string;
18        email_work: string;
19        email_personal: string;
20        phone_number_work: string;
21        phone_number_personal: string;
22        is_active: boolean;
23        active_start_date: string;

```

```

24   active_end_date: string;
25   password?: string;
26   created_by: string;
27   updated_by: string;
28 }
29
30 export default function UserFormPage() {
31   const searchParams = useSearchParams();
32   const id = searchParams.get("id");
33   const router = useRouter();
34   const { data: session } = useSession();
35
36   const initialFormData: UserFormData = {
37     first_name: "",
38     last_name: "",
39     login_id: "",
40     email_work: "",
41     email_personal: "",
42     phone_number_work: "",
43     phone_number_personal: "",
44     is_active: false,
45     active_start_date: "",
46     active_end_date: "",
47     password: "",
48     created_by: session?.user?.id || "",
49     updated_by: session?.user?.id || "",
50   };
51
52   const [formData, setFormData] = useState<UserFormData>(
53     initialFormData);
54   const [loading, setLoading] = useState(false);
55   const [errors, setErrors] = useState<Record<string, string>>({});
56
57   useEffect(() => {
58     const fetchData = async () => {
59       if (!id || !session?.user?.token) return;
60
61       try {
62         setLoading(true);
63         const response = await req.GET(
64           `/admin/users?id=${id}`,
65           session.user.token
66         );
67         setFormData({
68           ...response,
69           updated_by: session.user.id || "",
70         });
71       } catch (error) {
72         toast({
73           title: "",
74           description: "❌ ❌ ❌ ❌ ❌",
75           variant: "destructive",

```



```

75     });
76     } finally {
77         setLoading(false);
78     }
79 };
80
81     fetchUserData();
82 }, [id, session?.user?.id, session?.user?.token]);
83
84 const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {
85     const { name, value, type, checked } = e.target;
86     setFormData((prev) => ({
87         ...prev,
88         [name]: type === "checkbox" ? checked : value,
89     }));
90 };
91
92 const handleSubmit = async (e: React.FormEvent) => {
93     e.preventDefault();
94
95     try {
96         setLoading(true);
97         const response = await req.PUT(
98             `/admin/users/${id || ""}`,
99             session?.user?.token || "",
100             {
101                 ...formData,
102                 active_start_date: formData.active_start_date || null,
103                 active_end_date: formData.active_end_date || null,
104             }
105         );
106
107         if (response) {
108             toast({
109                 title: "",
110                 description: "",
111             });
112         }
113     } catch (error) {
114         console.error("Error submitting form:", error);
115         toast({
116             title: "",
117             description: "",
118             variant: "destructive",
119         });
120     } finally {
121         setLoading(false);
122     }
123 };
124
125 return (
126     <div className="py-8 px-4">

```

```

127 <Card>
128   <CardHeader>
129     <CardTitle className="text-2xl">
130       {id ? " " : " " }
131     </CardTitle>
132   </CardHeader>
133   <CardContent>
134     <form onSubmit={handleSubmit} className="space-y-4">
135       <div className="space-y-6">
136         <h3 className="text-lg font-medium" > </h3>
137         <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
138           <div>
139             <Label htmlFor="first_name" ></Label>
140             <Input
141               id="first_name"
142               name="first_name"
143               type="text"
144               value={formData.first_name}
145               onChange={handleChange}
146               className={errors.first_name ? "border-red-500" :
147                 ""}
148             />
149             {errors.first_name && (
150               <p className="text-red-500 text-sm mt-1">
151                 {errors.first_name}
152               </p>
153             )}
154           </div>
155           <div>
156             <Label htmlFor="last_name" ></Label>
157             <Input
158               id="last_name"
159               name="last_name"
160               type="text"
161               value={formData.last_name}
162               onChange={handleChange}
163               className={errors.last_name ? "border-red-500" :
164                 ""}
165             />
166             {errors.last_name && (
167               <p className="text-red-500 text-sm mt-1">
168                 {errors.last_name}
169               </p>
170             )}
171           </div>
172           <div>
173             <Label htmlFor="login_id" > ID</Label>
174             <Input
175               id="login_id"
176               name="login_id"
177               type="text"
178               value={formData.login_id}

```

```

177         onChange={handleChange}
178         className={errors.login_id ? "border-red-500" : ""}
179     />
180     {errors.login_id && (
181       <p className="text-red-500 text-sm mt-1">
182         {errors.login_id}
183       </p>
184     )}
185   </div>
186   {!id && (
187     <div>
188       <Label htmlFor="password" > </Label>
189       <Input
190         id="password"
191         name="password"
192         type="password"
193         value={formData.password}
194         onChange={handleChange}
195         className={errors.password ? "border-red-500" : ""}
196       />
197       {errors.password && (
198         <p className="text-red-500 text-sm mt-1">
199           {errors.password}
200         </p>
201       )}
202     </div>
203   )}
204 </div>
205 </div>
206 <div className="space-y-2">
207   <h3 className="text-lg font-medium" > </h3>
208   <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
209     <div>
210       <Label htmlFor="email_work" > </Label>
211       <Input
212         id="email_work"
213         name="email_work"
214         type="email"
215         value={formData.email_work}
216         onChange={handleChange}
217         className={errors.email_work ? "border-red-500" : ""}
218       />
219       {errors.email_work && (
220         <p className="text-red-500 text-sm mt-1">
221           {errors.email_work}
222         </p>
223       )}
224     </div>
225   </div>

```

```

226     <Label htmlFor="email_personal" > </Label>
227     <Input
228       id="email_personal"
229       name="email_personal"
230       type="email"
231       value={formData.email_personal}
232       onChange={handleChange}
233       className={errors.email_personal ? "border-red
        -500" : ""}
234     />
235     {errors.email_personal && (
236       <p className="text-red-500 text-sm mt-1">
237         {errors.email_personal}
238       </p>
239     )}
240   </div>
241   <div>
242     <Label htmlFor="phone_number_work" > </Label>
243     <Input
244       id="phone_number_work"
245       name="phone_number_work"
246       type="tel"
247       value={formData.phone_number_work}
248       onChange={handleChange}
249     />
250   </div>
251   <div>
252     <Label htmlFor="phone_number_personal" > </
      Label>
253     <Input
254       id="phone_number_personal"
255       name="phone_number_personal"
256       type="tel"
257       value={formData.phone_number_personal}
258       onChange={handleChange}
259     />
260   </div>
261 </div>
262 </div>
263 <div className="space-y-2">
264   <h3 className="text-lg font-medium" ></h3>
265   <div className="grid grid-cols-1 md:grid-cols-2 gap-4
      items-center">
266     <div className="flex items-center space-x-2">
267       <input
268         id="is_active"
269         name="is_active"
270         type="checkbox"
271         checked={formData.is_active}
272         onChange={handleChange}
273         className="h-4 w-4 border-gray-300 rounded"
274       />

```

```

275         <Label htmlFor="is_active" > </Label>
276     </div>
277     <div className="space-x-2">
278         <Label > :</Label>
279         <DatePickerWithRange
280             value={{
281                 from: formData.active_start_date ? new Date(
282                     formData.active_start_date) : undefined,
283                 to: formData.active_end_date ? new Date(
284                     formData.active_end_date) : undefined,
285             }}
286             onChange={(date) => {
287                 setFormData((prev) => ({
288                     ...prev,
289                     active_start_date: date?.from ? date.from.
290                         toISOString() : "",
291                     active_end_date: date?.to ? date.to.
292                         toISOString() : "",
293                 }));
294             }}
295             />
296     </div>
297 </div>
298 </div>
299 <UserRole userId={id!} />
300 <div>
301     <Button type="submit" className="w-full" disabled={
302         loading}>
303         {loading ? " □ ..." : id ? " " : " "}
304     </Button>
305 </div>
306 </form>
307 </CardContent>
308 </Card>
309 </div>
310 );
311 }

```

B.4 Back-End

B.4.1 /Controller/Admin/userController.go

```

1 package controllers
2
3 import (
4     "branch-admin-service/log"
5     "branch-admin-service/models"
6     services "branch-admin-service/services/admin"
7     "net/http"
8
9     "github.com/gin-gonic/gin"
10 )
11
12 func GetAllUsers(c *gin.Context) {
13     id := c.Query("id")
14
15     if id != "" {
16         user, err := services.GetUserByID(id, c)
17         if err != nil {
18             log.Error("Failed to fetch user by ID", map[string]interface{}{
19                 "error": err.Error(), "id": id}, c)
20             c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed to fetch user"})
21             return
22         }
23         if user == nil {
24             log.Warn("User not found", map[string]interface{}{"id": id}, c)
25             c.JSON(http.StatusNotFound, gin.H{"error": "User not found"})
26             return
27         }
28         log.Info("User fetched successfully", map[string]interface{}{"id": id}, c)
29         c.JSON(http.StatusOK, user)
30         return
31     }
32
33     users, err := services.GetAllUsers(c)
34     if err != nil {
35         log.Error("Failed to fetch users", map[string]interface{}{"error": err.Error(), "id": id}, c)
36         c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed to fetch users"})
37         return
38     }
39     log.Info("Users fetched successfully", map[string]interface{}{"count": len(users)}, c)
40     c.JSON(http.StatusOK, users)
41 }

```

```

42 func CreateUser(c *gin.Context) {
43     var user models.User
44     if err := c.ShouldBindJSON(&user); err != nil {
45         log.Error("Failed to bind user JSON", map[string]interface{}{"
46             error": err.Error()}, c)
47         c.JSON(http.StatusBadRequest, gin.H{"error": "Invalid request
48             payload"})
49         return
50     }
51     log.Info("Creating user", map[string]interface{}{"user": user}, c)
52     newUser, err := services.CreateUser(user, c.Request)
53     if err != nil {
54         log.Error("Failed to create user", map[string]interface{}{"user":
55             user, "error": err.Error()}, c)
56         c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()
57             })
58         return
59     }
60     log.Info("User created successfully", map[string]interface{}{"
61         userID": newUser.ID}, c)
62     c.JSON(http.StatusCreated, newUser)
63 }
64
65 func UpdateUser(c *gin.Context) {
66     id := c.Param("id")
67     var user models.User
68
69     if err := c.ShouldBindJSON(&user); err != nil {
70         log.Error("Failed to bind user JSON", map[string]interface{}{"
71             error": err.Error()}, c)
72         c.JSON(http.StatusBadRequest, gin.H{"error": "Invalid request
73             payload"})
74         return
75     }
76     log.Info("Updating user", map[string]interface{}{"userID": id, "
77         user": user}, c)
78     updatedUser, err := services.UpdateUser(id, user, c.Request)
79     if err != nil {
80         log.Error("Failed to update user", map[string]interface{}{"userID
81             ": id, "error": err.Error()}, c)
82         c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed to
83             update user"})
84         return
85     }
86     log.Info("User updated successfully", map[string]interface{}{"
87         userID": updatedUser.ID}, c)
88     c.JSON(http.StatusOK, updatedUser)

```

```
83 }
84
85 func DeleteUser(c *gin.Context) {
86     id := c.Param("id")
87
88     if err := services.DeleteUser(id, c.Request); err != nil {
89         log.Error("Failed_to_delete_user", map[string]interface{}{"userID": id, "error": err.Error()}, c)
90         c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed_to_delete_user"})
91         return
92     }
93
94     log.Info("User_deleted_successfully", map[string]interface{}{"userID": id}, c)
95     c.JSON(http.StatusOK, gin.H{"message": "User_deleted_successfully"})
96 }
97
98 func CheckLoginIDExists(c *gin.Context) {
99     loginID := c.Query("login_id")
100     if loginID == "" {
101         log.Error("login_id_is_required", map[string]interface{}{}, c)
102         c.JSON(http.StatusBadRequest, gin.H{"error": "login_id_is_required"})
103         return
104     }
105
106     exists, err := services.CheckLoginIDExists(loginID, c)
107     if err != nil {
108         log.Error("Failed_to_check_login_id", map[string]interface{}{"error": err.Error()}, c)
109         c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed_to_check_login_id"})
110         return
111     }
112
113     log.Info("Checked_login_id_existence", map[string]interface{}{"login_id": loginID, "exists": exists}, c)
114     c.JSON(http.StatusOK, gin.H{"exists": exists})
115 }
```