

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

Баянжаргалын Энх-Амгалан

Веб хөгжүүлэлт
(Web developing)

Мэдээллийн технологи (D061303)
Үйлдвэрлэлийн дадлагын тайлан

Улаанбаатар

2025 оны 01 сар

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

Веб 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 ”Веб 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	2
1. БАЙГУУЛЛАГЫН ТАНИЛЦУУЛГА	2
1.1 Товч Танилцуулга	2
1.2 Үйлчилгээ	2
1.3 Гол Системүүд	2
2. АЖЛЫН ТӨЛӨВЛӨГӨӨ	3
3. СИСТЕМИЙН ШААРДЛАГА	4
3.1 Хэрэглэгчийн шаардлага	4
3.2 Use Case диаграмм	6
3.3 Entity Relationship диаграмм	7
4. АШИГЛАХ ТЕХНОЛОГИ	8
4.1 Gitlab	8
4.2 NextJS	8
4.3 Golang	8
4.4 PostgreSQL	9
4.5 Postman	10
4.6 Docker	10
5. ХЭРЭГЖҮҮЛЭЛТ	11
5.1 GORM хэрэгжүүлэлт	11
5.2 Golang Test хэрэгжүүлэлт	17
5.3 Front-End хэрэгжүүлэлт	22
5.4 Back-End хэрэгжүүлэлт	35
6. ДҮГНЭЛТ	42
6.1 Үр дүн	42

6.2 Дүгнэлт.....	45
НОМ ЗҮЙ	46
ХАВСРАЛТ.....	47
А. УДИРДАГЧИЙН ҮНЭЛГЭЭ	47
В. КОДЫН ХЭРЭГЖҮҮЛЭЛТ.....	48
В.1 GORM.....	48
В.2 Тест	50
В.3 Front-end	54
В.4 Back-End	61

ЗУРГИЙН ЖАГСААЛТ

3.1	Use Case диаграмм	6
3.2	Entity Relationship диаграмм	7
6.1	Хэрэглэгчдийн жагсаалт	42
6.2	Хэрэглэгч засах	43
6.3	Өдөр сонгох календарь	43
6.4	Ажилтны эрх сонгох хэсэг	44

ХҮСНЭГТИЙН ЖАГСААЛТ

2.1	Төлөвлөгөө гаргасан байдал	3
-----	----------------------------------	---

Кодын жагсаалт

5.1	Get All Users Service	12
5.2	Create User Service	12
5.3	Update User Service	13
5.4	Delete User Service	14
5.5	Get User By ID Service	14
5.6	Check if login ID exists Service	15
5.7	Test Database Setup	17
5.8	Test Create User Function	18
5.9	Test Get All Users Function	19
5.10	Test Update User Function	20
5.11	UserFormData интерфэйсийг үүсгэсэн байдал	23
5.12	UserFormPage функц (Main component)	23
5.13	Хувьсагч	23
5.14	initialFormData хоосон утга оноосон байдал	24
5.15	Төлөв хадгалах	24
5.16	Хэрэглэгчийн мэдээлэл татах useEffect	25
5.17	handleChange	26
5.18	handleSubmit	26
5.19	UI бүтэц	27
5.20	CreateUser Controller	37
5.21	CreatUser Controller	38
5.22	UpdateUser Controller	38
5.23	DeleteUser Controller	39
5.24	CheckLoginIDExists Controller	40

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

1. БАЙГУУЛЛАГЫН ТАНИЛЦУУЛГА

1.1 Товч Танилцуулга

ONDO компани 2018 онд байгуулагдаж, 2021 онд албан ёсоор үйл ажиллагаагаа эхлүүлсэн. 2022 оны 3 сард анхны бүтээгдэхүүнээ зах зээлд нэвтрүүлж, ХУГАЦААГҮЙ, ХЯЗГААРГҮЙ интернэтийн үйлчилгээг хэрэглэгчдэдээ санал болгосон. Тус компани дэлхийн зах зээлд гарахаар зорин ажиллаж байна. Мөн Stakeholder economy-ийг дэмжиж, ёс зүйтэй үйл ажиллагаа явуулдаг. Одоогийн байдлаар 120 залуусаас бүрдсэн ONDO компанийн хэрэглэгчид 385 мянга болсон.

1.2 Үйлчилгээ

ONDO нь дэвшилтэт технологи, хурдан сүлжээ, мобайл дата болон уян хатан үйлчилгээний багцуудыг санал болгодог. Хэрэглэгчид хүссэн газраасаа үйлчилгээ авах боломжоор хангадаг.

1.3 Гол Системүүд

ONDO нь хэрэглэгч төвтэй, дижитал платформд суурилсан системүүд ашигладаг:

- **CRM:** Хэрэглэгчдийн мэдээлэл, шаардлага, гомдол удирдах.
- **BSS:** Өдөр тутмын үйл ажиллагаа, төлбөр тооцоо.
- **OSS:** Сүлжээний үйл ажиллагаа, чанар сайжруулах.
- **Billing Systems:** Төлбөр тооцоог удирдах.

2. АЖЛЫН ТӨЛӨВЛӨГӨӨ

Table 2.1: 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

3. СИСТЕМИЙН ШААРДЛАГА

Дадлагын 21 хоногийн хугацаанд "Branch Admin" гэх дотоодын программ дээр Front-End болон Back-End хөгжүүлэлтийг хийсэн. Уг системийн товч тайлбарлавал дуудлагын төв болон хэрэглэгчид үйлчлэх төвд ажиллаж байгаа ажилтан боруулалт хийх болон хэрэглэгчид тулгарсан асуудалыг шийдвэрлэх процессыг хөнгөвчлөх зорилготой. Мөн борлуулалт хянах тайлан гаргах боломжтой систем юм.

3.1 Хэрэглэгчийн шаардлага

3.1.1 Хэрэглэгчид

- Админ болон салбарын ажилтнууд орно.

3.1.2 Функционал хэрэглэгчийн шаардлагууд

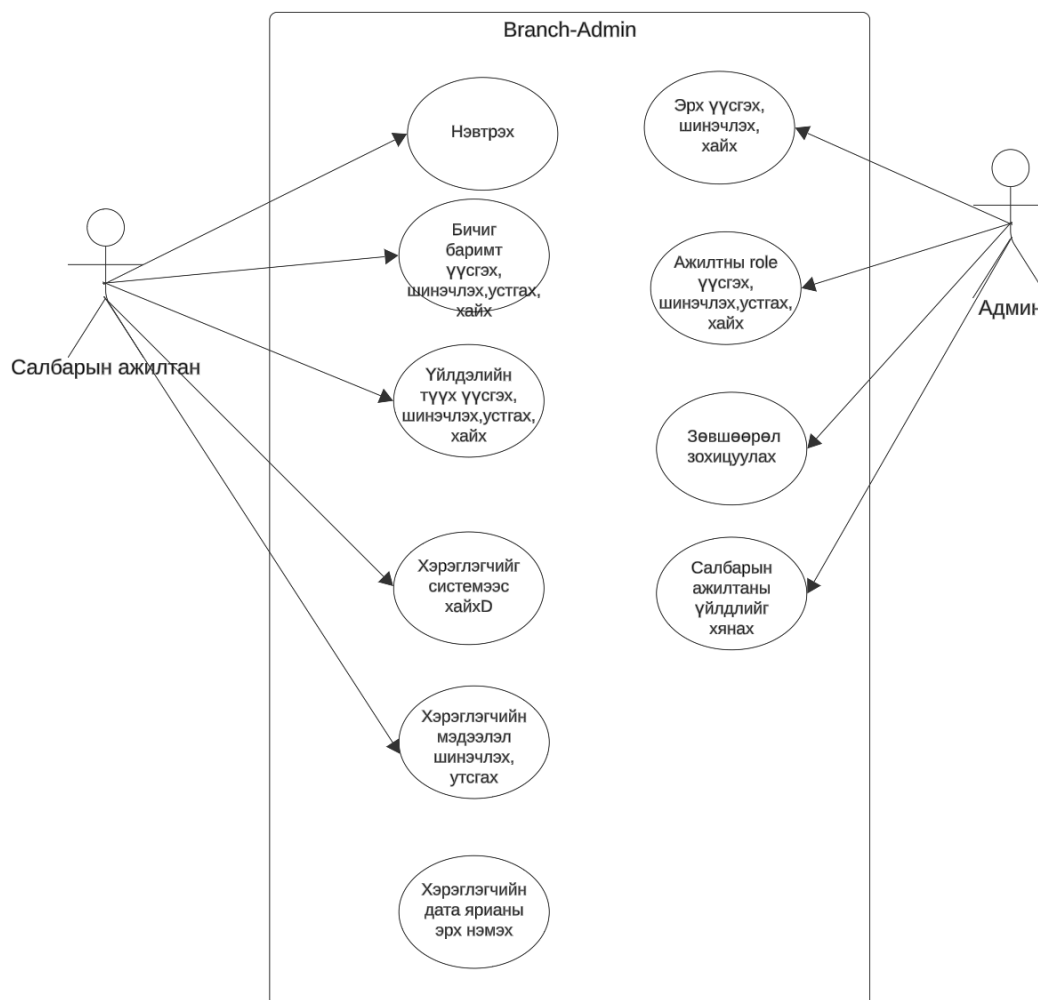
- Хэрэглэгчийн бүртгэл хийх, шинэчлэх, устгах боломжтой байх.
- Систем нь хэрэглэгчийн мэдээллийг найдвартай хадгалж, аюулгүй байдлыг хангах ёстой.
- Үйлчлүүлэгчийн үйлчилгээний мэдээллийг хянах, шинэчлэх, засварлах боломжтой байх.
- Систем нь олон хэрэглэгчийн нэгэн зэрэг оролцох боломжийг дэмжих ёстой.
- Систем нь хэрэглэгчийн хүсэлт, гомдлыг хүлээн авч, шийдвэрлэх ажиллагааг дэмжих.

3.1.3 Функционал бус шаардлага

- Систем нь хурдан ачаалагдах, бага хүлээх хугацаатай байх ёстой.
- Хэрэглэгчийн интерфэйс энгийн, ойлгомжтой, хэрэглэгчдэд хялбар байх.
- Систем нь олон хэрэглэгчийн нэгэн зэрэг оролцох чадвартай байх.

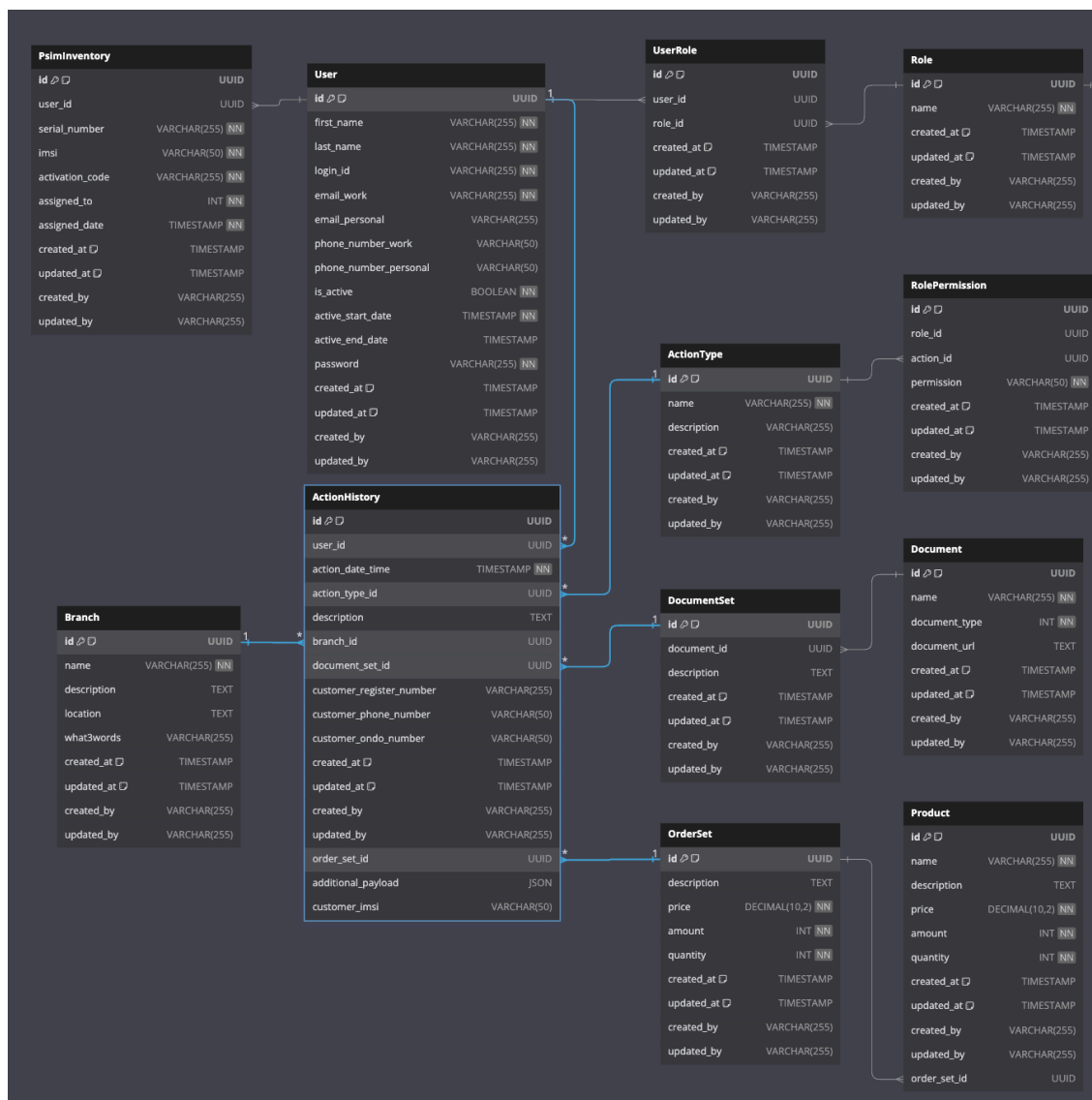
- Системийн аюулгүй байдал өндөр байх ёстой, түүнд нэвтрэх эрхгүй хүн орох боломжгүй байх.
- Систем нь өргөжүүлэлт хийх боломжтой, шинэ хэрэглэгчид болон үйлчилгээг дэмжих чадвартай байх.

3.2 Use Case диаграмм



Зураг 3.1: Use Case диаграмм

3.3 Entity Relationship диаграмм



Зураг 3.2: Entity Relationship диаграмм

4. АШИГЛАХ ТЕХНОЛОГИ

4.1 Gitlab

GitLab нь Git дээр суурилсан DevOps платформ бөгөөд программ 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.

4.2 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 веб аппликейшнуудыг бүтээхэд туслах зорилготой. Энэ нь хуудасны динамик маршрут, API endpoint-ууд, автомат статик зэрэг олон төрлийн функцуудыг агуулдаг. Next.js нь хөгжүүлэгчдэд илүү хялбар бөгөөд хурдан хөгжүүлэлтийг хангахын зэрэгцээ, веб аппликейшны гүйцэтгэлийг сайжруулдаг. Мөн Next.js нь TypeScript-ийг дэмжиж, хөгжүүлэгчдэд илүү найдвартай, уншигдахуйц код бичих боломжийг олгодог.

4.3 Golang

Golang буюу Go нь Google-ээс гаргасан, хурдан, найдвартай, энгийн синтакс бүхий программчлалын хэл юм. Go нь системийн программчлал, сүлжээний программчлал, 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 хөгжүүлэгчдэд өндөр гүйцэтгэлтэй программуудыг бүтээх боломжийг олгодог. Go нь мөн стандарт сангуудын багцтай байдаг бөгөөд энэ нь хөгжүүлэгчдэд нэмэлт сангүйгээр 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 хүснэгтүүдэд хамааруулан ажиллах боломжийг олгодог. Энэ нь CRUD (Create, Read Update, Delete) үйлдлүүдийг гүйцэтгэх, өгөгдлийн миграци, холболт 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.

4.4 PostgreSQL

PostgreSQL нь хүчирхэг, нээлттэй open source объект хандлагат database бөгөөд сүүлийн 30 гаруй жилийн турш идэвхтэй хөгжиж байгаа найдвартай ажиллагаатай мэдээллийн сангийн программ 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 программуудаас 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.

4.5 Postman

Postman нь API хөгжүүлэлт, тест хийх, баримтжуулалт хийхэд ашиглагддаг хэрэгсэл юм. Энэ нь хөгжүүлэгчдэд API-уудыг хялбархан турших, өөрчлөлт оруулах, багийн гишүүдтэй хамтран ажиллах боломжийг олгодог. Postman нь REST, GraphQL, SOAP зэрэг олон төрлийн API-уудыг дэмждэг бөгөөд хөгжүүлэгчдэд 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.

4.6 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 хөгжүүлэгчдэд 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.

5. ХЭРЭГЖҮҮЛЭЛТ

5.1 GORM хэрэгжүүлэлт

Энэ хэсэгт 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
- GORM-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

5.1.1 User Service

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

Энэ функц нь бүх хэрэглэгчдийн мэдээллийг өгөгдлийн сангаас авч, ямар нэгэн алдаа гарсан тохиолдолд тухайн алдааг лог болгон харуулна.

```

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 5.1: Get All Users Service

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

```

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 }

```

```
6
7 hashedPassword, err := utils.HashPassword(user.Password)
8 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 5.2: Create User Service

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

```
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     }
```

```

10 }
11 return user, nil
12 }

```

Код 5.3: Update User Service

Хэрэглэгчийг устгах үед түүний үүрэг болон хэрэглэгчийн мэдээллийг устгана.

```

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 5.4: Delete User Service

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

```

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 5.5: Get User By ID Service

Энэ функц нь өгөгдлийн сангаас login ID-гийн байгаа эсэхийг шалгадаг. Хэрэв байвал true, байхгүй бол false 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    }
11 }

```

```
8     }
9     log.Error("Failed to check login_id", map[string]interface{}{"error
    ": result.Error.Error()}, r)
10     return false, result.Error
11 }
12
13 return true, nil
14 }
```

Код 5.6: Check if login ID exists Service

5.2 Golang Test хэрэгжүүлэлт

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

5.2.1 Тестийн орчин

Тестийн орчин байгуулахдаа өгөгдлийн санг моклож, 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 5.7: Test Database Setup

5.2.2 Тестийн функцууд

- `TestCreateUser`: Шинэ хэрэглэгч үүсгэх үйлдлийг шалгана.
- `TestGetAllUsers`: Бүх хэрэглэгчийн мэдээллийг авах функцыг 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 5.8: 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 5.9: 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 5.10: Test Update User Function

5.3 Front-End хэрэгжүүлэлт

Энэ хэсэгт Next.js фреймворкийг ашиглан хэрэглэгчийн бүртгэлд засвар хийх хуудсыг хийнэ. Хэрэглэгчийн өгөгдлийг хүлээн авч, сервер рүү илгээх, мөн серверээс хариу авах процессыг оролцуулан бичигдсэн. Үндсэн зорилго нь хэрэглэгчийн бүртгэл болон мэдээллийг санд оруулах үндсэн үйл ажиллагааг веб 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

5.3.1 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 5.11: UserData интерфейсийг үүсгэсэн байдал

Энэ функц нь хэрэглэгчийн мэдээллийг нэмэх, засах формыг харуулна

```

1 export default function UserFormPage() {}

```

Код 5.12: 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 5.13: 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 5.14: 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 5.15: 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 5.16: 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 5.17: 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 5.18: 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     />
112     {errors.email_personal && (
113         <p className="text-red-500 text-sm mt-1">
114             {errors.email_personal}
115         </p>
116     )}

```

```

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 5.19: UI бүтэц

5.4 Back-End хэрэгжүүлэлт

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.

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

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

20 }

Код 5.20: CreateUser Controller

5.4.2 CreateUser

CreateUser функц нь хэрэглэгчийн мэдээллийг JSON хэлбэрээр хүлээн авч, models.User объект болгон 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":
5             : err.Error()}, r)
6         return nil, err
7     }
8     return users, nil
9 }
```

Код 5.21: CreateUser Controller

5.4.3 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": err.Error()}, c)
7      c.JSON(http.StatusBadRequest, gin.H{"error": "Invalid request payload"})
8      return
9  }
10
11  log.Info("Updating user", map[string]interface{}{"userID": id, "user": user}, c)
12  updatedUser, err := services.UpdateUser(id, user, c.Request)
13  if err != nil {
14      log.Error("Failed to update user", map[string]interface{}{"userID": id, "error": err.Error()}, c)
15      c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed to update user"})
16      return
17  }
18
19  log.Info("User updated successfully", map[string]interface{}{"userID": updatedUser.ID}, c)
20  c.JSON(http.StatusOK, updatedUser)
21 }
```

Код 5.22: UpdateUser Controller

5.4.4 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 5.23: DeleteUser Controller

5.4.5 CheckLoginIDExists

CheckLoginIDExists функц нь хэрэглэгчийн login id-ийг URL-ийн query параметрээс авч, өгөгдлийн санд 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 }
```



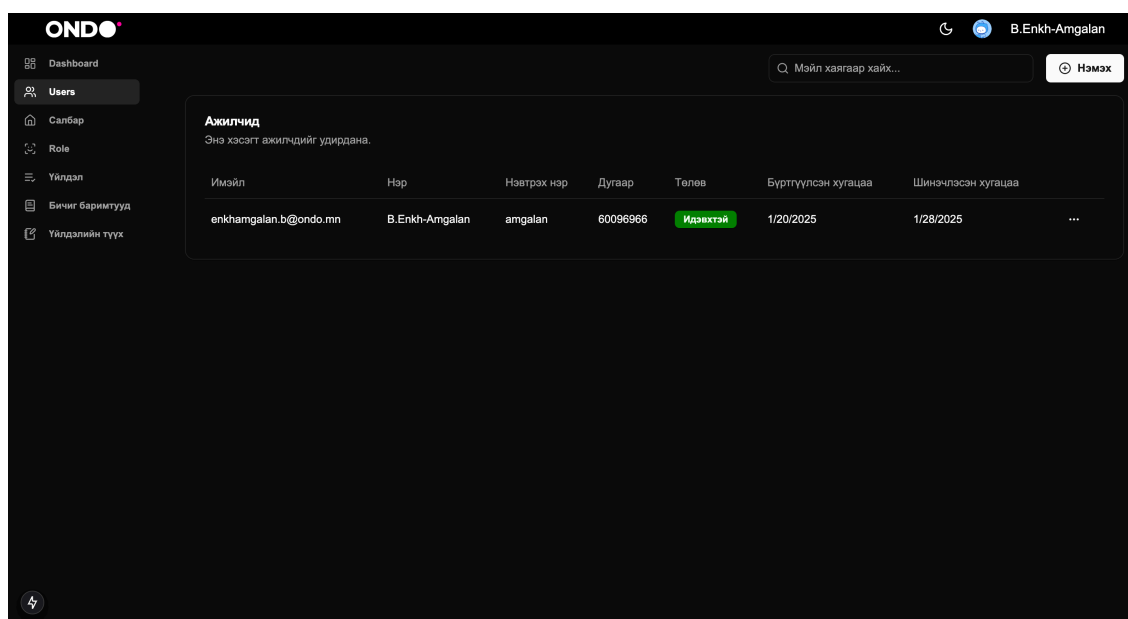
```
        })  
6      return  
7    }  
8  
9    exists, err := services.CheckLoginIDExists(loginID, c)  
10   if err != nil {  
11     log.Error("Failed to check login_id", map[string]interface{}{"error"  
12       ": err.Error()}, c)  
13     c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed to  
14       check login_id"})  
15     return  
16   }  
17  
18   log.Info("Checked login_id existence", map[string]interface{}{"  
19     login_id": loginID, "exists": exists}, c)  
20   c.JSON(http.StatusOK, gin.H{"exists": exists})  
21 }
```

Код 5.24: CheckLoginIDExists Controller

6. ДҮГНЭЛТ

6.1 Үр дүн

6.1.1 *Front харагдах байдал*



Зураг 6.1: Хэрэглэгчдийн жагсаалт

ONDO

Dashboard
Users
Салбар
Role
Үйлдэл
Бичиг баримтууд
Үйлдэлийн түүх

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 6.2: Хэрэглэгч засах

ONDO

Dashboard
Users
Салбар
Role
Үйлдэл
Бичиг баримтууд
Үйлдэлийн түүх

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

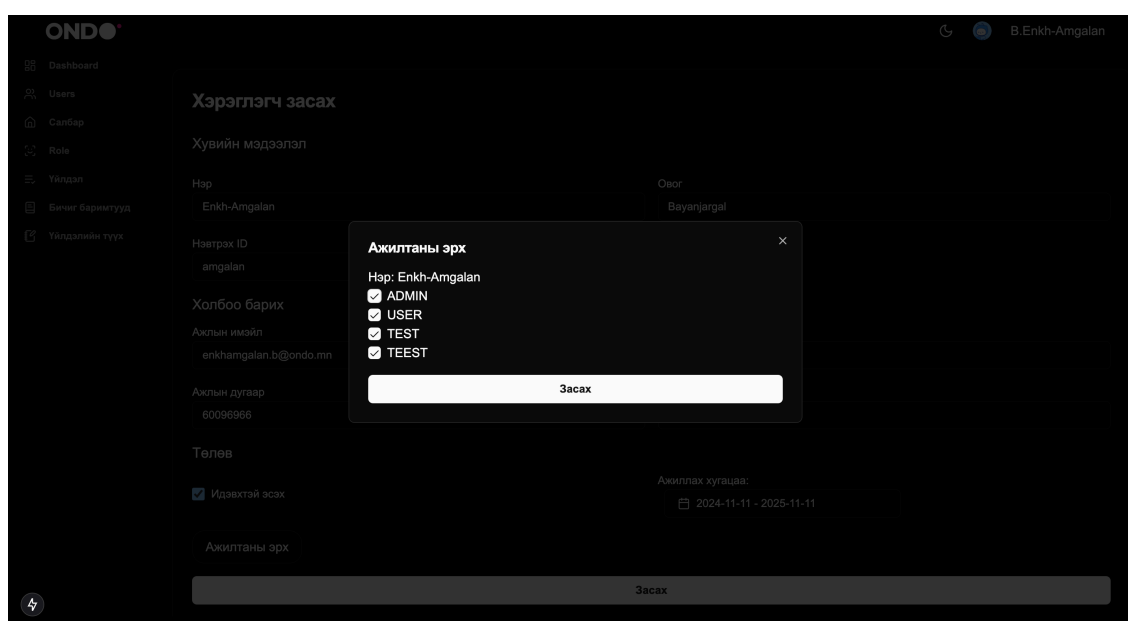
☒ Идэвхтэй эсэх

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

Ажилтаны эрх

Засах

Зураг 6.3: Өдөр сонгох календарь



Зураг 6.4: Ажилтны эрх сонгох хэсэг

6.2 Дүгнэлт

Миний бие ОНДО 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, GITLAB дээрх Feature Branch, Next.js болон Golang түүний сан GIN болон өгөгдлийн сантай ажиллах GORM зэрэг сантай ажиллаж 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 Full-Stack хөгжүүлэгч болохоор зорьж байна.

Дадлагын хүрээнд бүрэн хэмжээний хөгжүүлэлтийн талаар практик ойлголтыг олж авч богино хугацаанд өөрийн хэмжээнд багагүй туршлага хуримтлуулсан. Мөн хийсэн төсөл дээрээ тулгуурлан өөрийн тайланг бичих туршлагатай болсон. Цаашид сурсан зүйлсээ илүү бататган ирээдүйдээ ашиглахаар зорьж байна. Мэдлэгийн хүрээг минь тэлэхэд үнэтэй хувь нэмэр оруулсан МТЭС-МКУТ-даа болон ОНДО ХХК-д талархал илэрхийлье.

Bibliography

- [1] Golang Docs
<https://go.dev/doc/>
- [2] GORM Docs
<https://gorm.io/docs/index.html>
- [3] NextJS Docs
<https://nextjs.org/docs>
- [4] Go дээр тест бичих заавар
<https://go.dev/doc/tutorial/add-a-test>

А. УДИРДАГЧИЙН ҮНЭЛГЭЭ

МУИС, МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРЫН УХААНЫ ТЭНХИМ

Мугалийн технологийн АНГИЙН
ОЮУТАН Б.Энх-Амгалан -ийн

ҮЙЛДВЭРЛЭЛИЙН ДАДЛАГЫН АЖЛЫН УДИРДАГЧИЙН ТОДОРХОЙЛОЛТ

2025 оны 02 сарын 3

Мугалийн технологийн ангийн зивилимозүү кодтой оюутан Б.Энх-Амгалан нь манай байгууллагад 2025 оны сарын -ны өдрөөс сарын -ны өдөр хүртэл мэргэшүүлэх дадлагыг батлагдсан удирдамж, ажлын төлөвлөгөөний дагуу гүйцэтгэлээ. Оюутан Б.Энх-Амгалан -ын удирдамжийн дагуу дадлагын ажлыг гүйцэтгэсэн байдал: Б.Энх-Амгалан нь ОНҮО компаний нэг сарын хугацаанд дадлагаар ажиллахдаа өгөгдсөн ажлын даалгаварын хүрээнд мэдлэгээ нэмэгдүүлж, бие даан судлах даалгаварыг гүйцэтгэхийн тулд ОНҮО компаний мэдээлэл технологийн багтай уялдаж ажиллах байсан. Б.Энх-Амгалан -ийн хувьд мэргэжлийнхээ хүрээнд онолын мэдлэгээ, практиктай уялдуулан мэргэжлийн ур чадвараа нэмэгдүүлэх хүсэл эрмэлзэлтэйгээр дадлагын ажлыг амжилттай гүйцэтгэж дуусгалаа.

Үнэлгээний санал: Үйлдвэрлэлийн дадлагын хариуцсан удирдагчийн өгөх оноо
(10 онооноос өгнө)

Б.Энх-Амгаланг үйлдвэрлэлийн дадлагын хүрээнд арваас
арван оноо өгг байна. (10/10)

Үйлдвэрлэлийн дадлагын хариуцсан удирдагч

Н.Ог-Эрдэнэ

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, 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     <div>
300         <Button type="submit" className="w-full" disabled={
301             loading}>
302             {loading ? "  " : id ? "  " : "  "}
303         </Button>
304     </div>
305 </form>
306 </CardContent>
307 </Card>
308 </div>
309 );
310 }

```

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 }
```