

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

Анужингийн Сайнзолбоо

АЖИЛ ОЛГОГЧДЫН ӨГӨГДӨЛД СУУРИЛСАН ЧАТ БОТ

(Chat bot based on employers' data)

Мэдээллийн технологи (D061303)
Бакалаврын судалгааны ажил

Улаанбаатар

2022 он

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

АЖИЛ ОЛГОГЧДЫН ӨГӨГДӨЛД СУУРИЛСАН ЧАТ БОТ
(Chat bot based on employers' data)

Мэдээллийн технологи (D061303)
Бакалаврын судалгааны ажил

Удирдагч: _____ Б.Хуягбаатар доктор (Ph.D.)

Гүйцэтгэсэн: _____ А.Сайнзолбоо (18B1NUM1762)

Улаанбаатар

2022 он

Зохиогчийн баталгаа

Миний бие Анужингийн Сайнзолбоо “АЖИЛ ОЛГОГЧДЫН ӨГӨГДӨЛД СУУРИЛСАН ЧАТ БОТ” 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: _____

УДИРТГАЛ	1
БҮЛГҮҮД	2
1. СЭДВИЙН ТАНИЛЦУУЛГА	2
1.1 Оршил	2
1.2 Зорилго	2
1.3 Зорилт	2
1.4 Алсын хараа	3
2. СИСТЕМИЙН СУДАЛГАА	4
2.1 Системийн судалгаа	4
2.2 Ижил төстэй системүүд	7
2.3 Технологийн судалгаа	9
3. СИСТЕМИЙН ШИНЖИЛГЭЭ	15
3.1 Бизнесийн үйл ажиллагааны шинжилгээ	15
3.2 Хэрэглэгч	16
3.3 Функционал шаардлага	16
3.4 Функционал бус шаардлага	17
3.5 Use case диаграмм	18
4. СИСТЕМИЙН ЗОХИОМЖ	19
4.1 Өгөгдлийн сангийн диаграмм	19
4.2 Өгөгдлийн элемент	20
4.3 Өгөгдлийн сангийн холбоосын тайлбар	22
4.4 Класс диаграмм	22
5. ХЭРЭГЖҮҮЛЭЛТ, ҮР ДҮН	25
5.1 Хөгжүүлсэн байдал	25
5.2 Үр дүн	45

ДҮГНЭЛТ	52
ДҮГНЭЛТ	52
НОМ ЗҮЙ	53
ХАВСРАЛТ.....	54
А. ҮЕЧИЛСЭН ТӨЛӨВЛӨГӨӨ	54
В. КОДЫН ХЭРЭГЖҮҮЛЭЛТ.....	55
В.1 Өгөгдөл цугуулалт	55
В.2 Өгөгдөл нэгтгэх, цэвэрлэх	60
В.3 Өгөгдлийн сангийн холболт	61
В.4 Чатбот хөгжүүлэлт	65

ЗУРГИЙН ЖАГСААЛТ

2.1	Pizza Hut chat bot	7
2.2	WHO's chat bot	8
2.3	Өгөгдөл цуглуулалтын жишээний үр дүн	10
2.4	AWS-EC2 Server тасралтгүй ажиллах үр дүн	10
2.5	EC2 дээрх PostgreSQL өгөгдлийн сантай холбогдсон үр дүн	11
2.6	SQLAlchemy өгөгдлийн сантай холбогдох нь	12
2.7	Чатботын амьралын мөчлөг	13
3.1	BPMN-1	15
3.2	BPMN-2	16
3.3	Use Case диаграмм	18
4.1	Өгөгдлийн сангийн диаграмм	19
4.2	Класс диаграм	23
5.1	Үндсэн процесс зураглал	25
5.2	Data set	31
5.3	Виртуал машин дээрх crontab	32
5.4	Виртуал машин дээрх pm2	33
5.5	Өгөгдлийн статистик-1	34
5.6	Өгөгдлийн статистик-2	35
5.7	Өгөгдлийн статистик-3	36
5.8	Өгөгдлийн статистик-4	37
5.9	Өгөгдлийн статистик-5	38
5.10	Чатботын хариулж чадах асуултууд	39
5.11	Угтах тескт	46
5.12	<Байгууллага>-д <ажил> ажлын байрны мэдээлэл?	47
5.13	Бусад ажлын байрууд	48

5.14	<Байгууллага>-д нээлттэй ажлын байрууд?	49
5.15	<Ажиллах цаг> цагийн <ажил> ажлын байр?	50
5.16	<Цалин> цалинтай <ажил> ажлын байр?	51
A.1	Бакалаврын судалгааны ажлын үечилсэн төлөвлөгөө	54
B.1	Фолдерийн бүтэц	55

ХҮСНЭГТИЙН ЖАГСААЛТ

4.1	advertisement хүснэгт	21
4.2	category хүснэгт	22

Кодын жагсаалт

2.1	BeautifulSoup жишээ өгөгдөл цугуулалт	9
2.2	PostgreSQL-тэй харьцах объектын класс	11
5.1	Data Link crawling	27
5.2	Өгөгдөл цуглуулах	28
5.3	Upsert to PostgreSQL	29
5.4	Өгөгдөл цэвэрлэх функц	31
5.5	REST API controller	38
5.6	Question-understand классын кодын жишээ 1	39
5.7	Question-understand классын кодын жишээ 2	40
5.8	query бэлтгэх кодын жишнээ	40
5.9	JSON хүлээж авах кодын жишээ.	40
5.10	Хэрэглэгчийн UI дүрслэх кодын жишээ.	41
B.1	Бүх өгөгдлийг цуглуулах - dataScraping.py	55
B.2	Нэг зарын өгөгдлийг цуглуулах - adScrape.py	56
B.3	Өгөгдлийн төрөл - classTypes.py	59
B.4	Scrape хийх функц - scrape.py	60
B.5	Өгөгдөл цэвэрлэх - dataClean.py	60
B.6	Elastic IP-руу холбогдох - connection.py	61
B.7	Өгөгдлийн сангийн зохиомж болон upsert функц - upsert.py	63
B.8	чатбот класс - bot.js	65
B.9	Өгөгдлийн сангаас өгөгдөл авах - apiHelper.js	68
B.10	Хэрэглэгчийн асуултыг ойлгох query үүсгэх - questionUdnerstand.js	69
B.11	Хэрэглэгчид харагдах байдлыг угсрах - cardBuilder.js	70

УДИРТГАЛ

Мэдээллийн технологи эрчимтэй хөгжиж буй өнөөгийн нийгэмд байгууллага үйл ажиллагаа явуулж эхэлсэн цагаасаа эхлэн өгөгдлийг үйлдвэрлэсээр байдаг. Тэдгээр өгөгдлийг байнга хадгалах нь өгөгдлийн сангийн нөөцөд хортой байдаг тул өгөгдөлд шинжилгээ хийж, тэдгээрээс шаардлагатай өгөгдлүүдийг түүвэрлэн хадгалах нь чухал юм.

Өнөөдөр бид дэлхий нийтээрээ хурдтай амьдралын хэмнэлд ажиллаж, амьдарч байна. Мөн зах зээлийн хөгжил, ажил олгогчийн эрэлт хэрэгцээ ажил хайгчийн хүсэл сонирхлыг оновчтой бөгөөд хурдан холбож өгөх нь нэн шаардлагатай. Өнөөгийн байдлаар энэ эрэлт хэрэгцээг хангасан тодорхой шийдвэрлэсэн мэдээллийн систем хомс байна. Иймд энэхүү бакалаврын судалгааны ажлаар ажил олгогч болон ажил идэвхтэй хайгч хоёрыг түргэн шуурхай холбож өгөх чатбот системийг хөгжүүлж байна.

1. СЭДВИЙН ТАНИЛЦУУЛГА

1.1 Оршил

Энэхүү бакалаврын судалгааны ажлын хүрээнд “Ажил олгогчдын өгөгдөлд суурилсан чатбот” сэдвийн дагуу ажил хайгчдыг ажлын байрны мэдээллээр хангах чатбот системийг хөгжүүлнэ. Ажлын байрны мэдээллийг Data Scraping аргын тусламжтайгаар, системд шаардлагатай мэдээллийг өгөгдлийн сангийн хэлбэрт оруулан бүтэцтэйгээр нэгтгэн авах бөгөөд үүнээс ажил хайгчдын дунд байдаг түгээмэл асуултуудын хариултыг өгнө.

1.2 Зорилго

Ажлын хайгчдын хэрэгцээт асуултад хариулж, ажлын байрны хүртээмжийг нэмэгдүүлэхэд энэхүү системийн гол зорилго оршино.

1.3 Зорилт

Дээрх зорилгод хүрэхийн тулд дараах зорилтуудыг тавьсан. Үүнд:

- Ашиглагдах технологиудыг сонгох, судлах
- Ижил төстэй системийн судалгаа хийх
- Системийн шинжилгээ хийх
- Системийг зохиомжлох
- Системийг хөгжүүлэх, сайжруулалт хийх

1.4 Алсын хараа

Ажлын байрны дэлгэрэнгүй мэдээллийг цуглуулснаар цаашид тэдгээрт шинжилгээ хийж хамгийн их эрэлттэй, өндөр цалинтай ажлын байр гэх зэрэг мэдээллүүдийг систем хэрэглэгчдэд хүргэх боломжтой юм.

2. СИСТЕМИЙН СУДАЛГАА

2.1 Системийн судалгаа

Сонгосон сэдэв болох “Ажил олгогчдын өгөгдөлд суурилсан чатбот” сэдвийн хүрээнд судалгаа хийхдээ чатбот системийн талаар болон өгөгдөл цуглуулгын аргын талаар судалсан. Үүний дараа ижил төстэй системийн болон ашиглагдах технологийн талаар судалгааг хийсэн болно.

2.1.1 Чатбот систем

Чатбот систем нь ихэвчлэн хэрэглэгчийн асуултыг хиймэл оюун ухааны тусламжтайгаар ойлгож, хариултыг автоматжуулах үндсэн зорилготой компьютерийн програм хангамж юм. Орчин үед хэрэглэгчдэд туслах үндсэн үүргийн дагуу чатбот системийг байгууллагууд олон янзаар ашиглах болсон. Тэдгээрээс дурдвал,

- Цэс дээр суурилсан чатбот (Menu-based chatbot)
- Түлхүүр үгийг танихад суурилсан чатбот (Keyword recognition-based chatbot)
- Машин сургалтын чатбот (Machine learning chatbot)

Цэс дээр суурилсан чатбот

Өнөөгийн зах зээлд хэрэгжиж буй чатботуудын хамгийн энгийн бөгөөд түгээмэл хэлбэр юм.[1]

¹ Хэрэглэгчийн асууж болох асуултуудыг урьдаас таамаглан хариултуудыг мод хэлбэртэйгээр бүтэцлэн хадгалдаг. Хэрэглэгч хүссэн хариултаа авахын тулд системийн хадгалсан хариултаар аялах хэрэгтэй болдог. Бусад чатботтой харьцуулбал, хариулт хязгаарлагдмал бөгөөд хэрэглэгчээс олон асуулт асууж цаг их шаарддагаараа сул талтай байдаг.

¹<https://www.engati.com/blog/types-of-chatbots-and-their-applications>

Түлхүүр үгийг танихад суурилсан чатбот

Энэхүү чатбот нь хэрэглэгчийн бичсэнийг уншиж тохиромжтой хариултыг өгдөг. Ингэхдээ өгүүлбэрийг тодорхой дүрмийн дагуу боловсруулж түлхүүр үгийг таньж хариултыг өгдөг. Ижил төстэй олон асуултад хариулах эсвэл түлхүүр үг дутуу үед амжилтгүй болдог. Мөн хэрэглэгч хүссэн хариултаа олж чадахгүй байх болон үр дүн муутай хариулт өгсөн тохиолдолд цэс дээр суурилсан чатботыг хослуулан ашиглах нь найдвартай болдог бөгөөд түгээмэл шийдлүүдийн нэг байдаг.

Машин сургалтын чатбот

Энэ төрлийн чатбот нь өмнө хэрэглэгчийн харилцан яриан дээр хиймэл оюун ухаан болон машин сургалтын тусламжтайгаар шинжилгээ хийж, хэрэглэгчийн зан төлөв, асуултын хэв маягаас суралцдаг. Ингэснээрээ чатботод хэрэглэгчийн зарцуулах цаг эрчимтэйгээр буурах буюу хариултаа авах алхам багасгах ба хэрэглэгчийн туршлага (UX) нь түүнийгээ даган өсөх нь энэхүү чатботын үндсэн зорилго болно.

Чатботыг сонгох

Машин сургалтын чатбот нь илүү уян хатан хэрэглэгчдэд ээлтэй чатботыг бий болгодог боловч хөгжүүлэхэд цаг хугацаа их шаардагдах ба машин өөрөө суралцахад мөн хугацаа шаардагддаг. Иймд системийн нөөц, шаардлагыг харгалзан үзэж энэхүү судалгааны ажлаар түлхүүр үг танихад суурилсан чатботыг хэрэгжүүлэхийг зориод байна.

2.1.2 Өгөгдөл цуглуулгын арга

Өгөгдөл цуглуулах (data scraping) нь хэрэглэгчдэд харагдаж буй өгөгдлийг олон янзын сувгаас цуглуулан хувийн орчинд хадгалан цаашид ашиглах боломжийг олгодог хамгийн үр дүнтэй автомат өгөгдөл олборлох арга юм. Ихэвчлэн өгөгдөл цуглуулах арга нь вэбсайтаас шаардлагатай өгөгдлийг цуглуулахад ашигладаг. Өгөгдөл цуглуулж буй хүнээс хамааран олборлосон өгөгдлийг таслалаар тусгаарлагдсан утгын (Comma-Separated Values) файл эсвэл өгөгдлийн санд хадгалах боломжтой бөгөөд нэгэнт цуглуулсан их хэмжээний өгөгдөлд судалгаа

шинжилгээ хийх, худалдаа, борлуулалтын хэрэгсэл болгох зэрэг олон төрлийн боломжийг олгодог.

Вебсайтаас өгөгдлийг олборлох хамгийн түгээмэл арга нь HTML parsing буюу HTML-ийг задлан шинжлэх юм. Энэ нь вебсайтын HTML болох сайтын үндсэн бүтцийг агуулгынх нь хамтаар хуулах бөгөөд авах гэж буй өгөгдлийн зан төрхийг нь зааж өгснөөр доторх агуулгыг хамгийн хялбар бөгөөд автомат байдлаар цуглуулдаг юм. Цуглуулга хийх 2 үндсэн арга байдаг. Үүнд:

- Өгөгдлийг цуглуулж, задлах (Data scraping)
- Өгөгдлийг олж илрүүлж, хаягийг цуглуулах (Data crawling)

Өгөгдлийг цуглуулж, задлах

Нэг үгээр хэлбэл өгөгдлийг цуглуулж, задлах нь зааж өгсөн хаягийн дагуу шаардлагатай өгөгдлийг задалж, хэрэгтэй агуулгыг хөгжүүлэгчдэд өгдөг бөгөөд хүссэн өгөгдлөө задлан авах боломжийг олгодоогоороо давуу талтай. Өөрөөр хэлбэл өгөгдөл олборлох програм нь зорилго буюу даалгавраа мэдэж байгаа юм.

Өгөгдлийг олж илрүүлж, хаягийг цуглуулах

Энэхүү аргачлал нь хаяг тодорхой бус үед түүнийг олж илрүүлж шаардлагын дагуу хаягийг, зарим тохиолдолд өгөгдлийг цуглуулдаг. Системийн шаардлагын дагуу өгөгдлийг цуглуулах үед хаяг алгасах, дутуу өгөгдөл цуглуулахаас сэргийлдэг давуу талтай.

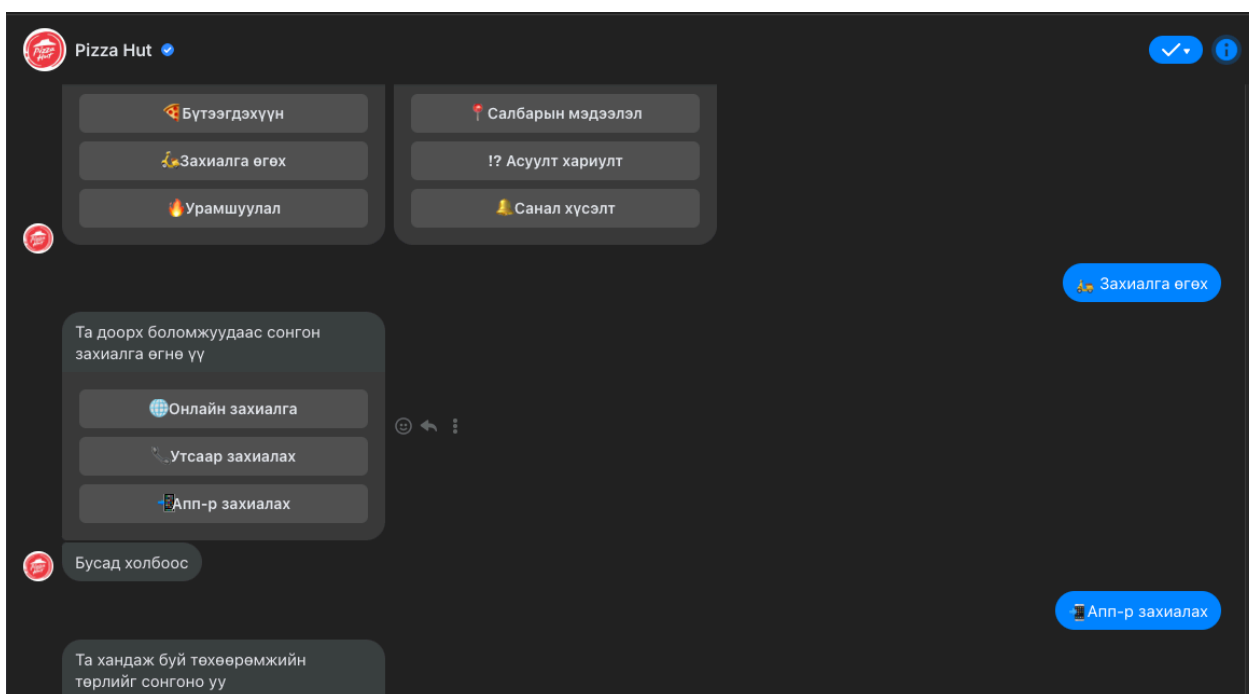
Ихэвчлэн энэхүү хоёр аргыг хослуулан ашигладаг бөгөөд шаардлагад нийцэх өгөгдлийг үлдээлгүй бүгдийг нь олоход *data crawling*-ийг ашиглах бол олсон өгөгдлийг задалж, шинжлэн өгөгдлийн санд хадгалах үйлдлийг *data scraping* хийдэг. Жишээлбэл, худалдааны сайтын бараа бүтээгдэхүүний өгөгдлийг цуглуулах гэж байгаа гэж үзвэл, барааны ангиллын хаягуудыг өөрчлөгдөх бүрд хадгалан өгөгдлийг цуглуулна. Өөрөөр хэлбэл нэг нь өөрчлөлт гарахыг ажиглаж вебсайтаар мөлхөж байх бол нөгөө нь шаардлагын дагуу бүх хэрэгтэй өгөгдлийг хэдийн цуглуулсан байна. Энэхүү бакалаврын судалгааны ажлын хүрээнд өгөгдлийг CSV файл үүсгэн хадгалж цаашид ашигласан болно.

2.2 Ижил төстэй системүүд

Гадаад ба дотоодын байгууллагуудын үйл ажиллагаандаа хэрэгжүүлдэг чатбот системүүдээс, *Domino's Pizza & Pizza Hut* болон WHO's Chat bot гэсэн гурван чатботыг сонгон авч судалгаанд оруулав.

2.2.1 *Domino's Pizza & Pizza Hut*

Domino's pizza хоолны газар нь захиалгын алхмаас эхлээд бүх мэдээллийг ганцхан *Facebook messenger chatbot* хангадаг. Чатбот эрчээ авч эхэлсэн шалтгаан нь хүмүүс, бусад хүмүүсийг хүлээлгүйгээр үйлчилгээ авах, тусламж авах зэрэг үйлчилгээг зэрэг нэвтрүүлсэнтэй холбоотой билээ. Үүний нэгэн адилаар Монголд үйл ажиллагаа явуулж буй Pizza Hut Mongolia юм.

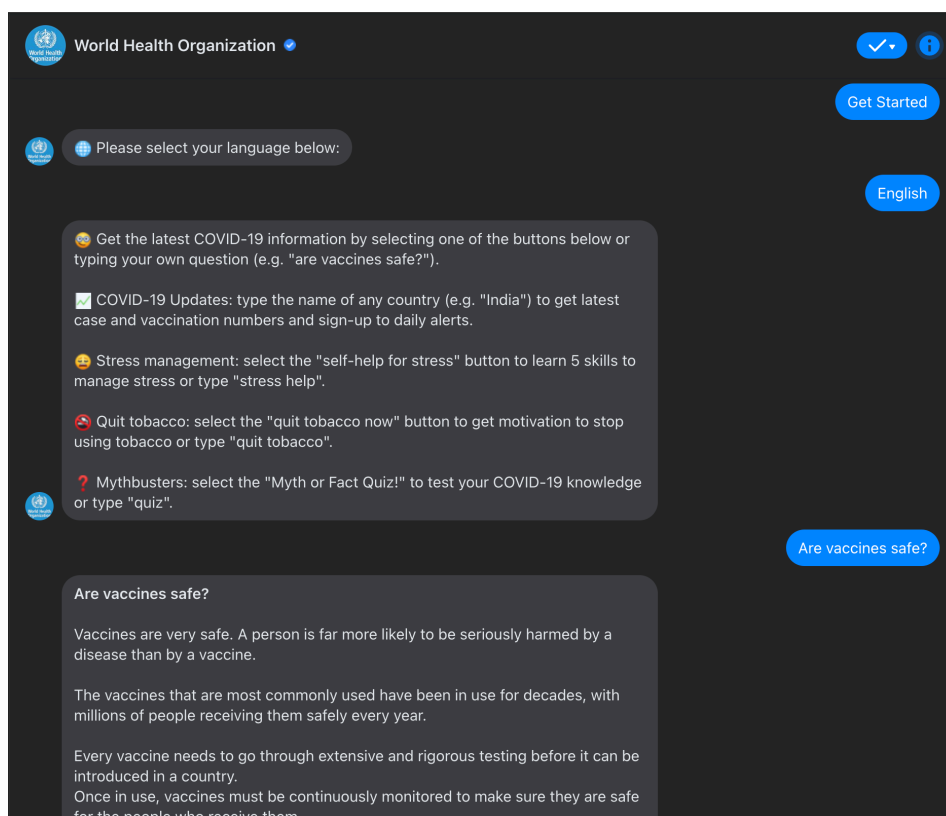


Зураг 2.1: Pizza Hut chat bot

Үйлчлүүлэгчдийн захиалга хүлээх хугацааг багасгахын тулд захиалгын үйл явцыг хурдасгаснаар тодорхой хэмжээнд нөлөөлж байгаа нь дээрх жишээнээс харагдаж байна.

2.2.2 World Health Organization's Chat bot

Цар тахал болох коронавирусын эрчимтэй тархаж байх үед дэлхийн өнцөг булан бүрд оршин суугаа хүмүүст цар тахлын мэдээлэл, урьдчилан сэргийлэх арга, баталгаатай эх сурвалжийн мэдээллээр хангах зорилготой чатбот юм. Дэлхий нийтээр вакцинжуулалтын хөдөлгөөн өрнөж байх үеэр вакцины талаарх мэдээлэл, архаг хууч өвчинд нөлөөлөх талаар найдвартай, хамгийн сүүлийн үеийн албан ёсны мэдээллийг өгдөг. Хэдий халдварын тоо буурч, нийгэм өөрөө дасан зохицож байгаа хэдий ч Дэлхийн Эрүүл Мэндийн байгууллага үүргээ гүйцэтгэж чухал эх сурвалжаар хангасаар байгаагийн шинж юм.



Зураг 2.2: WHO's chat bot

2.3 Технологийн судалгаа

АЖИЛ ОЛГОГЧДЫН ӨГӨГДӨЛД СУУРИЛСАН ЧАТ БОТ-ЫГ хөгжүүлэхдээ өгөгдөл цуглуулгыг *python* хэлний сан болох *BeautifulSoup* HTML өгөгдөл задлах технологийг ашигласан бөгөөд чатбот системийн үндсэн хөгжүүлэлтийг *Microsoft Bot Framework*-ийн тусламжтайгаар *javascript* хэл дээр хөгжүүлэлтийг хийж гүйцэтгэсэн. Чатбот болон өгөгдлийн сантай холбогдох буюу back-end хөгжүүлэлтийг *node.js*-ийн framework болон *express.js* ашигласан. Харин цуглуулсан өгөгдлийг AWS-EC2 сүлжээнд байршуулсан бөгөөд дүн шинжилгээ хийхдээ *CSV* файлд хадгалан ашигласан бөгөөд судалгааг дараах байдлаар хийсэн болно.

2.3.1 *BeautifulSoup*

Өгөгдөл цуглуулгын олон технологиудын нэг нь *BeautifulSoup* бөгөөд *python* програмчлалын хэлний сан юм. Энэ нь өгсөн вебсайтын хаяг (Url)-ийн дагуу бүх HTML өгөгдлийг агуулгын хамтаар нь хэрэглэгчид өгдөг. HTML хэл нь мод хэлбэртэй байдаг бөгөөд түүний хүүхэд элементүүдийн агуулгыг шаардлага болон түлхүүр үгийн дагуу цуглуулах зарчмаар ажилладаг. Бакалаврын судалгааны ажлын сэдвийн дагуу ажлын байр олгогчдын мэдээлэл болон ажлын байрны мэдээллийг **zangia.mn**-ээс *BeautifulSoup* ашиглан цуглуулсан. Доорх кодын жишээнд бүх ажлын байрны ангилал болон шүүлтүүрийн агуулгыг цуглуулсан бөгөөд жишээнд зориулж зөвхөн эхний ангиллын мэдээллийг харуулав.

```
1 from bs4 import BeautifulSoup
2 import requests
3 from urllib.error import HTTPError
4
5 url = 'https://zangia.mn/'
6 try:
7     response = requests.get(url)
8     response.raise_for_status()
```

```

9 except HTTPError as error:
10     print(error)
11 soup = BeautifulSoup(response.text, 'html.parser')
12 navigatorList = soup.find_all('div', class_='filter')
13 print(navigatorList[0])

```

Код 2.1: BeautifulSoup жишээ өгөгдөл цуглуулалт

```

<div class="filter">
  <h3>
    Онцлох
  </h3>
  <div>
    <a href="job/list/x.1">
      Удирдах албан тушаалын ажлын байр
    </a>
  </div>
  <div>
    <a href="job/list/x.3">
      Англи хэлний 100%-н мэдлэг шаардах ажлын байр
    </a>
  </div>
  <div>
    <a href="job/list/x.2">
      Ажлын туршлага шаардахгүй ажлын байр
    </a>
  </div>

```

Зураг 2.3: Өгөгдөл цуглуулалтын жишээний үр дүн

2.3.2 Amazon-Web-Services

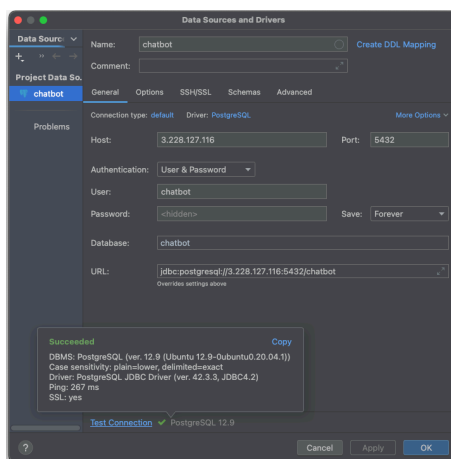
Амазон веб үйлчилгээ нь үүлэн тооцооллын системд суурилсан бөгөөд хувь хүн, байгууллагуудад сервер, нөөц хадгалах сан, сүлжээ, и-мэйл, алсын зайн удирдлага, виртуал машины удирдлага зэрэг үйлчилгээг өгдөг Амазоны охин компани юм. Чатбот системийн хувьд AWS-EC2 буюу үүлэн технологид суурилсан виртуал машиныг ашигласан үндсэн өгөгдөл хадгалах сан болгон ашигласан юм.

<input type="checkbox"/>	Name ▾	Instance ID	Instance state ▾	Instance type ▾	Status check
<input type="checkbox"/>	AWS_Server	i-03d487ded936fb1d5	🟢 Running 🔍	t2.micro	🟢 2/2 checks passed

Зураг 2.4: AWS-EC2 Server тасралтгүй ажиллах үр дүн

2.3.3 PostgreSQL

PostgreSQL нь үнэ төлбөргүй, нээлттэй эхийн *relational database management system* SQL-ийн ажиллах зарчимтай адил. Үүнийг маш олон веб, гар утасны аппликейшний үндсэн өгөгдлийн сан болгон ашигласан байдаг. Тэдгээрийн нэгэн адилаар бакалаврын судалгааны ажлын хүрээнд ажлын байр олгогчдын мэдээллийг AWS-EC2-д байрлах PostgreSQL өгөгдлийн санд хадгалах зарчмаар ажилласан юм.



Зураг 2.5: EC2 дээрх PostgreSQL өгөгдлийн сантай холбогдсон үр дүн

2.3.4 SQLAlchemy - ORM

Object-relational-mapping (ORM) нь систем хооронд өгөгдлийг дамжуулахад объект хандалтат програмчлалын хэлийг ашигладаг програмчлалын техник буюу арга юм. *SQLAlchemy* нь дээрх ORM аргыг хэрэгжүүлдэг Python хэлний нээлттэй эхийн SQL хэрэглүүр юм.

Өгөгдлийн сантай шууд объектоор харьцах боломжийг олгодгоороо давуу талтай бөгөөд чатбот системийн хувьд яг тохирсон технологи юм. Өөрөөр хэлбэл, өгөгдөл цуглуулгын явцад, өгөгдлийн объектууд үүсэх бөгөөд тэдгээрийг бүхлээр нь өгөгдлийн санруу оруулахад тохиромжтой юм. Жишээ болгож ангилал классыг доорх зурагт харуулав.

```
1 class PCategory(Base):  
2     __tablename__ = 'category'
```

```

23 def get_engine_from_settings():
24     keys = ['user', 'password', 'host', 'port', 'db']
25     if not all(key in keys for key in settings.keys()):
26         raise Exception('Bad config file')
27
28     return get_engine(settings['user'],
29                       settings['password'],
30                       settings['host'],
31                       settings['port'],
32                       settings['db'])

```

Зураг 2.6: SQLAlchemy өгөгдлийн сантай холбогдох нь

```

3
4 _id = Column(String, primary_key=True)
5 url = Column(String, nullable=False)
6 name = Column(String, nullable=False)
7 parent_id = Column(String,
8                  ForeignKey('category._id',
9                             onupdate='CASCADE',
10                             ondelete='CASCADE'),
11                  nullable=True)
12 category = relationship('PCategory')

```

Код 2.2: PostgreSQL-тэй харьцах объектын класс

2.3.5 Express.js

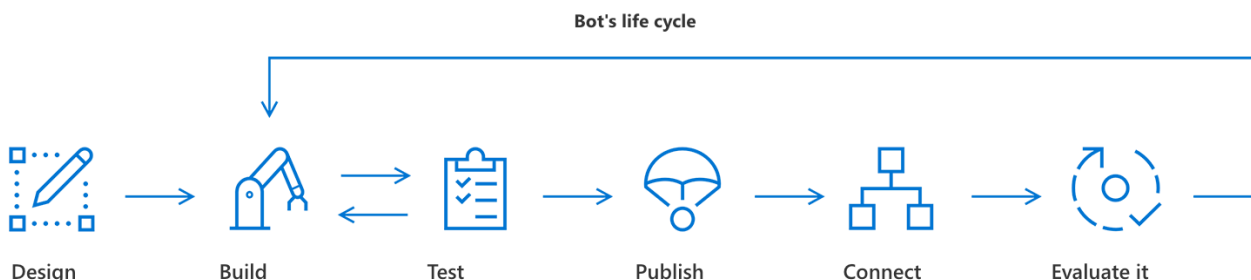
Express.js нь Node.js-ийн framework бөгөөд үнэ төлбөргүй MIT лицензийн нээлттэй эхийн програм хангамж юм. Энэ нь сервер болон веб програмуудыг бүтээхэд зориулагдсан бүх хэрэгслүүдийг хангасан хамгийн энгийн фреймворк юм. Бидний мэдэх request/response-ийг зохицуулах, чиглүүлэх үндсэн үүргийг гүйцэтгэдэг.

2.3.6 Comma Separated Values - CSV файл

CSV нь өгөгдлийн утгуудыг тусгаарлахад таслал ашигладаг текст файл юм. Файлын мөр бүр нь өгөгдөл байдаг бөгөөд харгалзах утгуудад текст файлыг бичих энгийн өгөгдөл хадгалах технологи юм. Их өгөгдөлтэй хялбар харьцах боломжийг олгодгоороо давуу талтай.

2.3.7 Microsoft Bot Framework

*Bot Framework*² нь Microsoft-ийн *Azure Bot Service*-ийн тусламжтайгаар чатботыг турших, үүсгэх, удирдах, хэрэглээнд нэвтрүүлэх гэх мэт боломжуудыг нэг дор хангаж өгдөг. Энэхүү боломжуудын хүрээнд асуулт хариултыг зохицуулах, хэрэглэгчид зориулсан User Interface бүтээх, Language Understanding аргыг ашиглах гэх мэт үйлдлүүдийг хийх боломжтой. Bot бүтээх үйл явцыг Azure Bot Service болон Bot Framework нь ихэд хөнгөвчилж өгдөг бөгөөд доорх зурагт үзүүлсэн дарааллын дагуу Bot системийг бүтээдэг.



Зураг 2.7: Чатботын амьралын мөчлөг

Design

Design буюу загварчлах нь төслийн төлөвлөгөөг гаргах юм. Өөрөөр хэлбэл, системийн зорилго, үйл явц, хэрэглэгчийн хэрэгцээг сайтар судлах нь амжилттай Bot систем бүтээх чухал хэсэг юм.

Build

Бот системийг угсрах буюу хөгжүүлэх үйл явц юм. Энэ алхамд хөгжүүлэгч хэрэглэгчийн

²<https://dev.botframework.com/>

харагдах хэсгийг загварчлах бөгөөд хөгжүүлэлтийн орчин нь *Azure Portal*, JavaScript, Python болон C програмчлалын хэлнүүдээс сонгож хөгжүүлэлтийг гүйцэтгэх явц юм. Мөн системийн шаардлагыг тодорхойлсны дагуу бот системийг өргөжүүлж ашиглах боломжтой бөгөөд тэдгээрээс дурдвал:

- Эх хэлний боловсруулалт (NLP)
- Асуулт хариултыг сайжруулан мэдлэгийн сан үүсгэх
- Хэрэглэгчийн интерфэйсийг сайжруулах

Test

Програм хангамжийн хөгжүүлэлтийн амьдралын мөчлөгийн адилаар тестийн үйл явцыг алгасаж болохгүй. Нэгэнт хэрэглэгчийн гарт бот системийг оруулахаас өмнө гарч болох алдаа дутагдлыг засан сайжруулах шаардлагатай. Иймд Bot системийг publish хийхээс өмнө заавал туршиж үзэх шаардлагатай. Энд Microsoft-ийн өөрсдийнх нь бие даасан програм болох *Bot emulator*-ийг ашиглан хөгжүүлэлтийн орчинд туршиж үзэх боломжийг олгодог.

Publish

Тестийн шатны дараа бот систем ашиглахад бэлэн болсон гэж үзсэн үед төсөл эсвэл чатботыг олон нийтэд ил болгох явдал юм.

Connect

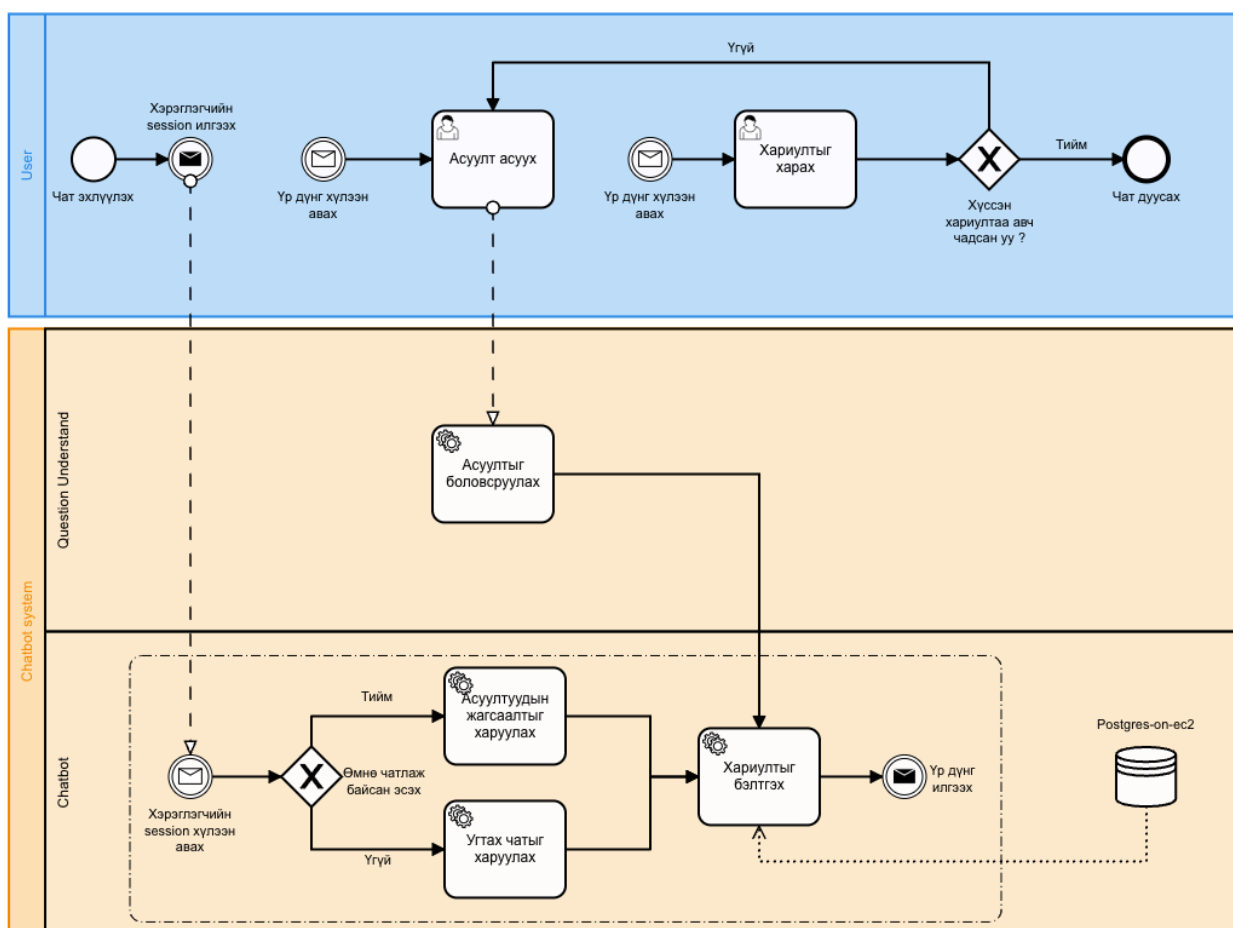
Bot системээ Facebook messenger, Microsoft Teams, Telegram, Skype гэх мэт чат сувгуудыг өөрийн Bot-той холбоно.

Ингэж бүх мөчлөгийг дууссаны дараа хөгжүүлэгч хэрэглэгчийн ашиглаж буй байдал дээр анализ хийж системийг дахин сайжруулах боломжтой бөгөөд буцаад угсрах үйл явцруу шилжих юм.

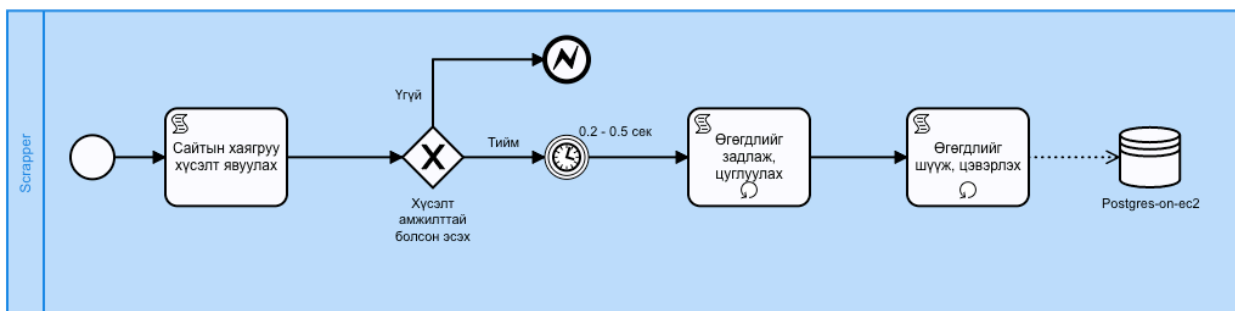
3. СИСТЕМИЙН ШИНЖИЛГЭЭ

3.1 Бизнесийн үйл ажиллагааны шинжилгээ

Бизнес процессийн модель нь чатбот системийн үндсэн процесс буюу үйл ажиллагааны явцыг BPMN-2.0 ашиглан дүрслэн харуулав [5]. Диаграммд дүрслэхдээ оролцогч талууд болох системүүдийг тус тусын *pool* дотор дүрсэлсэн бол дэд процесс буюу *subprocess*-ийг *lane*-д дүрсэлж хоорондын хамаарлыг харууллаа.



Зураг 3.1: BPMN-1



Зураг 3.2: BPMN-2

3.2 Хэрэглэгч

Чатбот системийг ямар ч хүн хэрэглэх боломжтой бөгөөд олон нийтэд нээлттэй байна. Системийн гол зорилго нь ажил хайж буй хэрэглэгчдэд ажлын байрны цогц мэдээллийг олгох зорилготой байх тул хэрэглэгчдийг дараах байдлаар тодорхойлж болно. Үүнд:

- Ажлын байр хайж буй хүн
- Хөгжүүлэгч

3.3 Функционал шаардлага

Дараах хэсэгт чатбот системд тавигдах функционал шаардлагуудыг харуулсан болно.

ФШ 1 Чатбот нь харилцан яриа эхэлмэгц хариу өгдөг байна.

ФШ 2 Чатбот нь ямар ч оролтод хариу өгнө.

ФШ 3 Хэрэв чатбот нь оролтод хариу өгч чадхааргүй байвал бусад асуултуудыг санал болгож ойлгомжгүй утга оруулсныг илэрхийлнэ.

ФШ 4 Чатботын санал болгох асуултууд нь текст хэлбэрээр харагдана.

ФШ 5 Чатботны хариулт нь текстэн хэлбэрээр хэрэглэгчид харагдана.

ФШ 6 Чатбот нь зөвхөн Монголоор бичсэн асуултад хариулт өгнө.

ФШ 7 Чатбот нь дэлгэрэнгүй мэдээллийг карт хэлбэрээр харуулан эх сурвалжийн хаягийг мөн харуулна.

3.4 Функционал бус шаардлага

Бэлэн болон найдвартай байдал (Availability & Reliability)

ФБШ 01 Чатбот систем өдрийн аль ч цагт 99.999% ажиллагаатай байх ёстой.

ФБШ 02 Ямар ч хүсэлт ирсэн чатбот 100% хариу өгдөг байна.

Гүйцэтгэлтэй байдал (Performance)

ФБШ 03 Чатботын байршуулсан сувагт, шаардлагаас хамаарч ямар ч төхөөрөмжөөс хандаж болно.

ФБШ 04 Зарим тохиолдолд чатботын гүйцэтгэл нь хэрэглэгчийн интернет болон төхөөрөмжийн үйлдлийн системийн хувилбараас хамаарч болно.

Дэмжих чадвар (Supportability)

ФБШ 05 Чатботын эх кодыг *github* дээр нээлттэй эхийн систем хэлбэрээр байршуулна.

Хэрэгцээт байдал (Usability)

ФБШ 06 Чатбот нь хэрэглэхэд хялбар, ойлгомжтой байна.

ФБШ 07 Чатботны цэс нь ойлгомжтой цөөн үгээр илэрхийлэгдсэн байна.

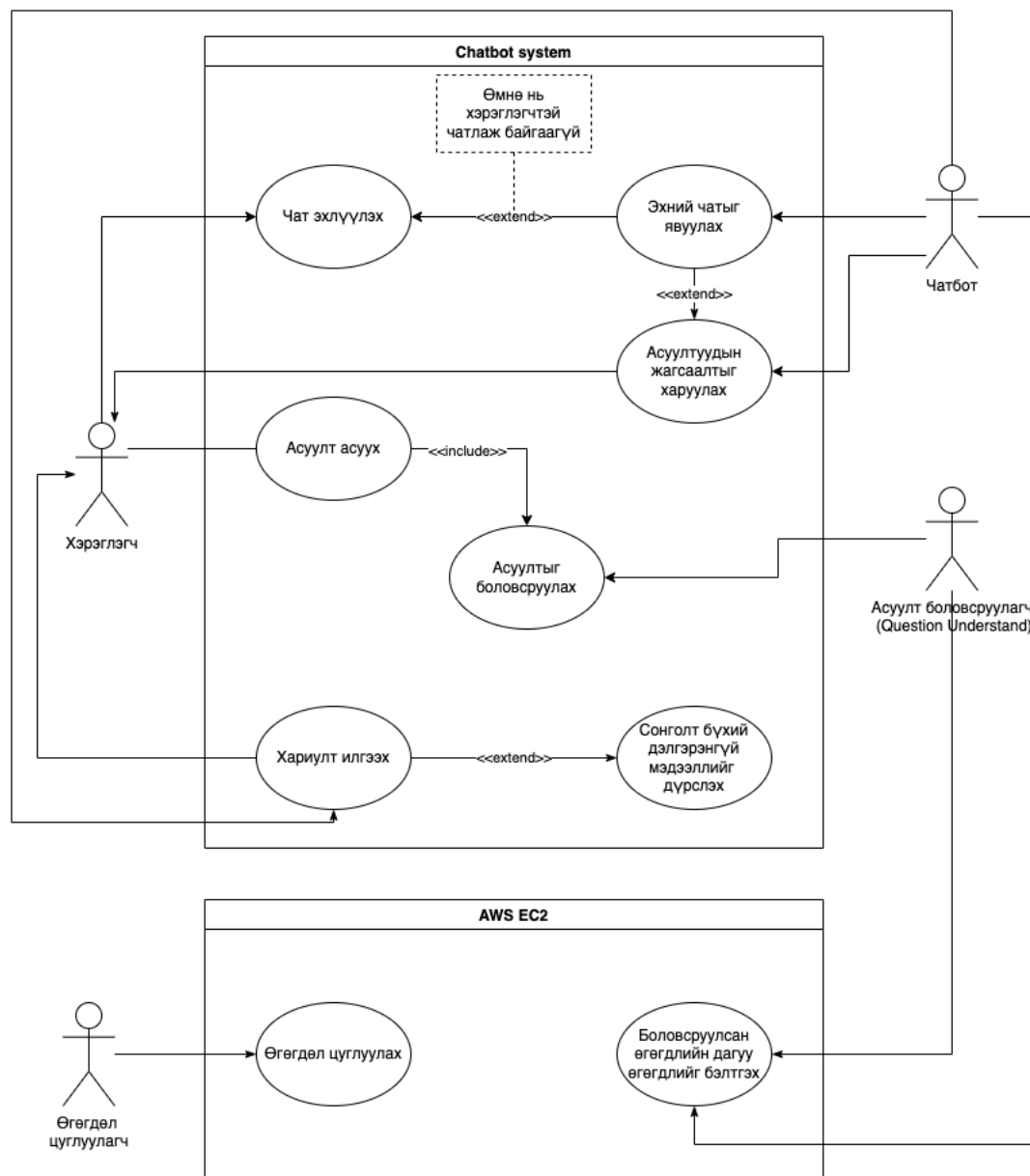
ФБШ 08 Чатботны цэсийн хэмжээ дарагдахуйц том байна.

Аюулгүй байдал (Security)

ФБШ 09 Чатбот системийн байршуулсан сувгийн стандартын дагуу хэрэглэгчийн мэдээллийг өгөгдлийн санд хадгалахгүй байна.

3.5 Use case диаграмм

Чатбот системийн use-case диаграммыг байдлаар тодорхойлов [4].



Зураг 3.3: Use Case диаграмм

4. СИСТЕМИЙН ЗОХИОМЖ

4.1 Өгөгдлийн сангийн диаграмм



Зураг 4.1: Өгөгдлийн сангийн диаграмм

4.2 Өгөгдлийн элемент

Чатбот системийн өгөгдлийн сангийн диаграммд харуулсан хүснэгтүүдэд агуулагдах мэдээлэл болон үүргийн талаар дэлгэрэнгүй тайлбарласан болно.

4.2.1 *advertisement* - Ажлын байрны зар

Ажлын байрны зар нь ямар категори буюу ангилалд, ямар холбоо барих хаягийн хамтаар хадгалагдаж буй мэдээлэл болон бусад дэлгэрэнгүй мэдээллийг харуулсан байна. Энд *level*

№	Баганын нэр	Түлхүүр өгөгдөл	Өгөгдлийн төрөл	Хоосон утга	Тайлбар
1	_id	PK	varchar	not null	Ажлын байрны зарын дахин давтагдашгүй дугаар
2	category_id	FK	varchar	not null	Ажлын байрны зард хамаарах ангилалын дугаар
3	url		varchar	not null	Ажлын байрны зарын хаяг
4	company		varchar	not null	Ажил олгогч компани / хувь хүн
5	title		varchar	not null	Ажлын зарын гарчиг
6	roles		varchar	null	Гүйцэтгэхүндсэн үүрэг
7	requirements		varchar	null	Ажлын байранд тавигдах шаардлага
8	additionalInfo		varchar	null	Нэмэлт мэдээлэл
9	city		varchar	null	Ажлын байрны харъяа хот
10	district		varchar	null	Ажлын байрны харъяа дүүрэг
11	exactAddress		varchar	null	Ажлын байрны бүтэн хаяг
12	level		level_types	null	Ажлын түвшин
13	type		workTime_type	null	Ажиллах цагийн төрөл

Table 4.1: advertisement хүснэгт

№	Баганын нэр	Түлхүүр өгөгдөл	Өгөгдлийн төрөл	Хоосон утга	Тайлбар
14	minSalary		float	null	Доод цалин
15	maxSalary		float	null	Дээд цалин
16	isDeable		boolean	null	Цалин тохиролцох эсэх
17	phoneNumber		varchar	null	Холбоо барих утасны дугаар
18	fax		varchar	null	Холбоо барих факсын дугаар
19	publishedDate		datetime	not null	Зар нийтэлсэн огноо

буюу ажлын түвшин, *type* буюу ажлын цагийн өгөгдлийн төрлийг тодорхойлохдоо дараах байдлаар зааж өгсөн.

Enum level_types буюу ажлын түвшний шаардлага нь дараах үндсэн 4 өгөгдлийн төрлөөс хамаарна:

- student - Оюутан / дадлагажигч
- professional - Мэргэжлийн
- occupationDoesntRequire - Мэргэжил шаардахгүй
- intermediateManagemet - Дунд шатны удирдлага
- topLevelManagemet - Дээд шатны удирдлага

workTime_type буюу ажиллах цагийн нөхцөл нь дараах үндсэн 4 өгөгдлийн төрлөөс хамаарна:

- shift - Ээлжийн
- fullTime - Бүтэн цагийн
- halfTime - Хагас цагийн

- contract - Гэрээт / зөвлөх
- seasonal - Улирлаар

4.2.2 category - Ангилал

Ажлын байрны зарын бүх ангиллуудын хаяг болон нэрийн мэдээллийг хадгалах хүснэгт юм. Ангиллууд нь дэд ангилал байж болох учир түүнийг эцэг ангиллын дугаарыг хадгалах байдлаар зохиомжлов.

Table 4.2: category хүснэгт

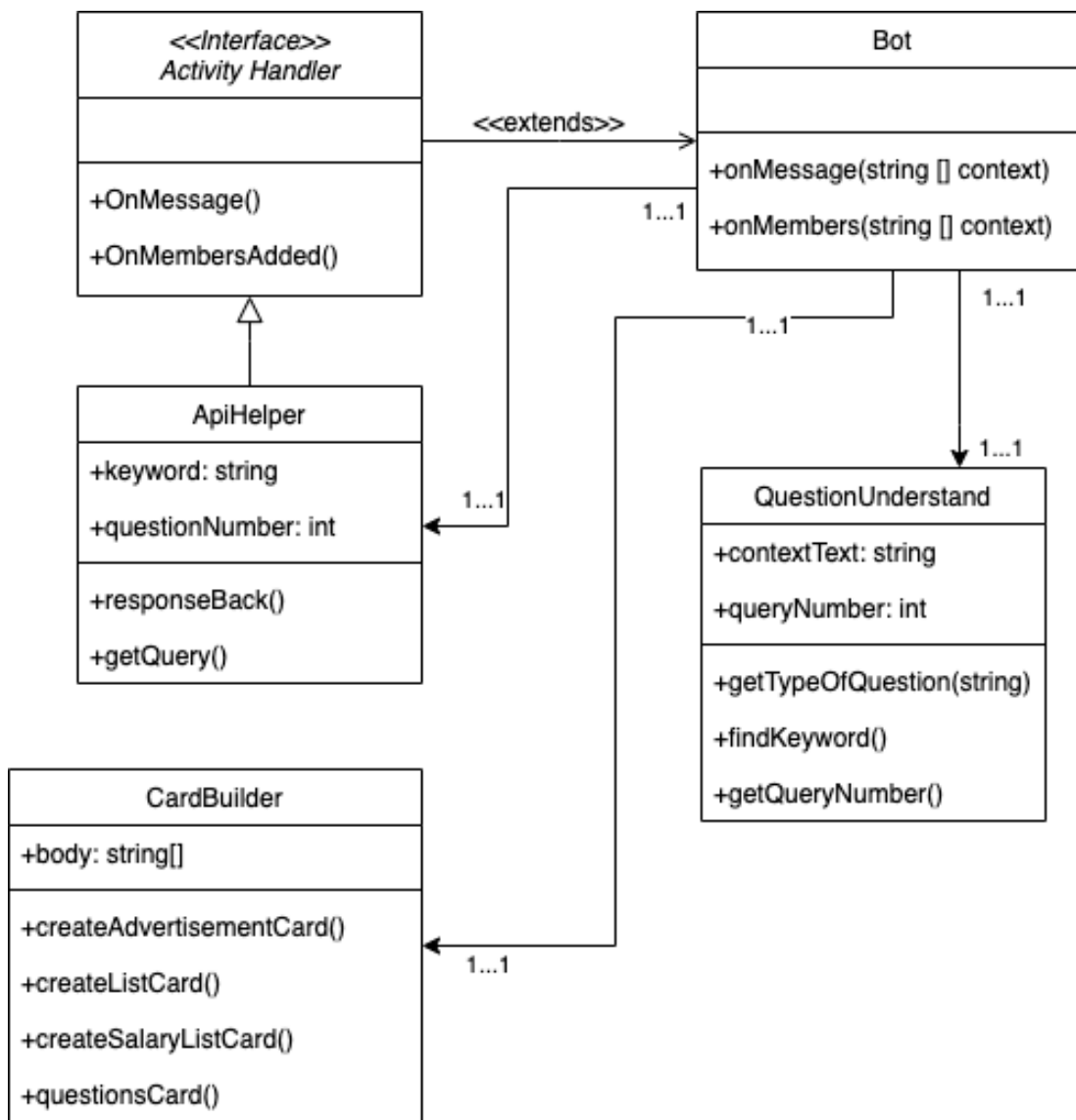
№	Баганын нэр	Түлхүүр өгөгдөл	Өгөгдлийн төрөл	Хоосон Утга	Тайлбар
1	id	PK	varchar	not null	Ажлын байрны зарын ангиллын дугаар
2	url		varchar	not null	Ангиллын хаяг
3	name		varchar	not null	Ангиллын нэр
4	parent_id	FK	varchar	null	Эцэг ангиллын дугаар

4.3 Өгөгдлийн сангийн холбоосын тайлбар

- Нэг ангилал буюу категорид олон ажлын байрны зар байж болно.
- Нэг ангилал буюу категорид олон категори байж болно.
- Нэг ажлын байрны зард нэг категори байна.

4.4 Класс диаграм

Програмын класс диаграмд классууд болон тэдгээрийн хоорондын хамаарлыг дараах байдлаар тодорхойлж үндсэн үүрэг, ажиллах зарчмуудыг тайлбарлав.



Зураг 4.2: Класс диаграм

4.4.1 Бот класс

Энэ классын үүрэг нь үндсэн Bot-ийн тохиргоог хийх, хэрэглэгчээс ирсэн асуултыг холбогдох класс-руу дамжуулна. Өөрөөр хэлбэл чатботыг ажиллуулах үндсэн эх бие нь юм. Мөн ActivityHandler интерфэйсд байх OnMessage, OnMembersAdded функцийг хэрэгжүүлэх бөгөөд эдгээр нь шинэ хэрэглэгч орж ирсэн талаарх session-г дамжуулдаг бөгөөд хэрэглэгчтэй харьцах цөм хэсгийг агуулна.

4.4.2 *QuestionUnderstand* класс

Энэхүү класс нь хэрэглэгчийн асуусан асуултыг боловсруулж түлхүүр үгийг таних үүрэгтэй. Урьдчилан бэлдсэн, хариулж чадах асуултын дагуу түлхүүр үгийг түүж аван, ямар төрлийн асуулт болохыг тодорхойлно. Улмаар үр дүнг *ApiHelper* классруу дамжуулна.

4.4.3 *ApiHelper* класс

Энэ классын үүрэг нь орж ирсэн түлхүүр үгийг ашиглан өгөгдлийн сантай холбогдон query илгээж үр дүнг хүлээн авна. Хүлээн авсан үр дүн JSON форматын өгөгдөл байх тул CardBuilder класс руу дамжуулна.

4.4.4 *CardBuilder* класс

Энэхүү класс нь орж ирсэн үр дүнг буюу хэрэглэгчдэд харуулах текстийг Microsoft Botframework-ийн AdaptiveCard-ийг ашиглан хэрэглэгчдэд бүтэцлэн харуулах үүрэгтэй.

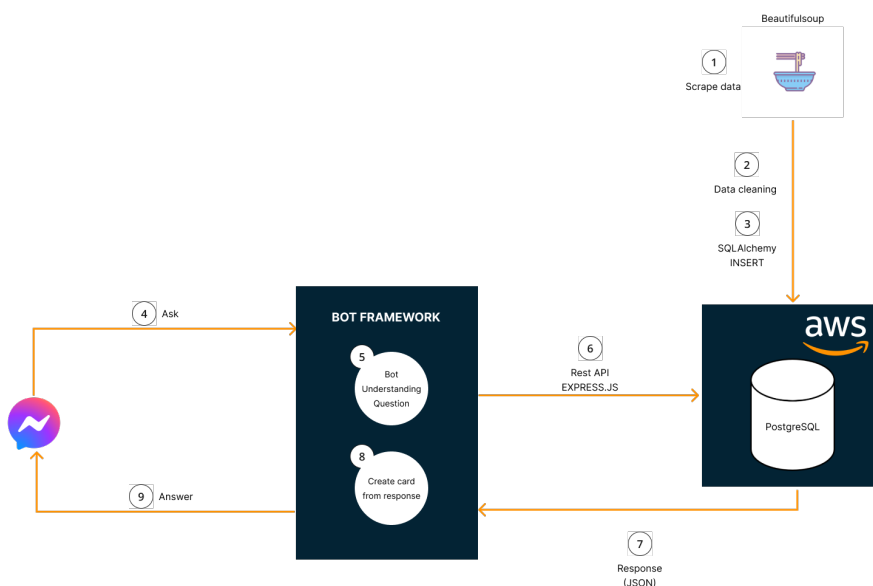
5. ХЭРЭГЖҮҮЛЭЛТ, ҮР ДҮН

5.1 Хөгжүүлсэн байдал

Чатбот системийн хөгжүүлэлтийг хийхдээ шаардлагууд дээр үндэслэн, үечилсэн төлөвлөгөө болон шаардлагатай хөгжүүлэлтийг дэс дараалалтайгаар хийж гүйцэтгэсэн.

- Өгөгдөл цуглуулах
- Өгөгдлийг нэгтгэх, цэвэрлэж өгөгдлийн санд хадгалах
- Системийн шаардлага, үйл ажиллагааг тодорхойлох
- Өгөгдөлд анализ хийх
- Чатбот хөгжүүлэх

гэсэн дарааллын дагуу хөгжүүлэлтийг хийсэн болно.



Зураг 5.1: Үндсэн процесс зураглал

Доорх зурагт чатбот системийн үндсэн процессийн зураглал харагдаж байна.

5.1.1 Өгөгдөл цуглуулах

Үндсэн ашиглагдах өгөгдөл болох ажил олгогчид, ажлын байрны өгөгдлийг **zangia.mn**-ээс BeautifulSoup ашиглан авсан. Эхлээд вебсайтынхаа HTML бүтцийг нь судалж, авах өгөгдлийнхөө класс утгуудыг (className) олж авах нь зөв юм. Вебсайтаас өгөгдөл цуглуулах 2 үндсэн арга байдгаас өгөгдлийг олж илрүүлж, хаягийг цуглуулах (data crawling) аргаар бүх ангиллуудын хаяг (url)-уудын түүж авна. Харин data scraping нь тэр хооронд олсон бүх хаягуудаараа явж хэрэгтэй агуулгыг цуглуулна.¹

```
1 categorySet = set()
2 adUrlDict = {}
3
4 soup = useScrape(initialUrl)
5 navigatorList = soup.find_all('div', class_='filter')
6 for navigator in navigatorList:
7     if navigator.find('h3').text.strip() != 'Salbar, mergejil':
8         continue
9     categoryList = navigator.find_all('div')
10
11 for categoryItem in categoryList:
12     categories = categoryItem.find('a')
13     url = initialUrl + categories['href']
14     tempCategory = Category(splitUrl(url, 'b.'), url, categories.text)
15     print('CATEGORY LINK SCRAPED! ', url, tempCategory.id)
16     soup = useScrape(url)
17     subCategory = soup.find('div', class_='pros')
18     subCategoryList = subCategory.find_all('a')
19     for subCategoryItem in subCategoryList:
```

¹Кодын жишээг оруулахад utf-8 формат танихгүй байсан тул монголоос галиглаж бичсэн болно.

```

20     subCategoryId = initialUrl + subCategoryId['href']
21     tempSubCategory = Category(splitUrl(subCategoryId, 'r.'),
22                               subCategoryId, subCategoryId.text
23                               , tempCategory)
24     categorySet.add(tempSubCategory)
25
26     categorySet.add(tempCategory)
27
28     for categoryItem in categorySet:
29         upsertCategory(categoryItem)
30         if categoryItem.parentCategory == None:
31             continue
32
33     soup = useScrape(categoryItem.url)
34     hasPagination = soup.find('div', class_='page-link')

```

Код 5.1: Data Link crawling

Дээрх код нь эхлээд вебсайтруу орж “filter” класс доторх “Салбар, мэргэжил” гэсэн хэсгээс бүх эцэг категориудыг data crawling хийж авч байна. Үүний дараа хүүхэд категориудыг олж categorySet дотор бүх хаягуудыг хийж хадгалж байна.² Энд categorySet set-ийн элемент нь category төрлийн объект бөгөөд өгөгдлийн сангийн диаграм дээр тодорхойлж өгсөн байгаа. Ингэснээр data crawling-ийг зогсоож, цуглуулсан хаягаасаа өгөгдлөө цуглуулъя.

```

1     if hasPagination != None:
2         pagesUrl = createLinkList(hasPagination, categoryItem.url)
3     else:
4         pagesUrl.append(categoryItem.url)
5     for pageUrl in pagesUrl:
6         soup = useScrape(pageUrl)
7         ads = soup.find_all('div', class_='ad')

```

²Python хэлний set өгөгдлийн төрөл нь давхацахгүй утгуудын хүснэгт гэж хэлж болно.

```

8         for ad in ads:
9             adUrl = initialUrl+ad.find('a', class_=None)['href']
10            adUrlDict[adUrl] = categoryItem
11        print(pagesUrl)
12        pagesUrl.clear()

```

Код 5.2: Өгөгдөл цуглуулах

Дээрх кодоод бүх хүүхэд категориудын дотор агуулагдаж буй зарын мэдээллийг цуглуулж байна. Ингэхдээ эхлээд категори доторх өгөгдлүүд нь хуудаслагдсан (pagination) байх боломжтой бөгөөд хэрэв олон хуудастай байвал хаягуудыг нь угсарч тэдгээрээс ч мөн өгөгдлийг нь цуглуулах ёстой юм. Энд dictionary үүсгэхдээ хаягийг нь түлхүүр (key) болгож категори объектыг нь утга(value) болгож хадгалсан. Мэдээж хэрэг dictionary нь түлхүүр давхцахаас сэргийлдэг тул бид ямар нэгэн байдлаар нэг зарын өгөгдлийг 2 удаа цуглуулах эрсдэлгүй болж байна.³

Харин одоо үүсгэсэн dictionary-оо ашиглан өгөгдлөө SQLAlchemy ашиглан PostgreSQL өгөгдлийн санруугаа хуулна. Ингэхдээ **upsert** функцийг хүснэгт бүрийн хувьд бичиж тухайн функцийн тусламжтайгаар update эсвэл insert хийх юм. Ангиллын хүснэгтийн хэрхэн upsert хийж байгааг доорх кодын жишээгээр харуулав.

```

1 def upsertCategory(category: Category):
2     if(category.parentCategory != None):
3         upsertCategory(category.parentCategory)
4     row = session.query(PCategory).filter(PCategory._id == category.id)
5     if row.first() == None:
6         if category.parentCategory != None:
7             insertToCategory(category, category.parentCategory.id)
8         else:
9             insertToCategory(category, None)

```

³Нэг зарын өгөгдлийг цуглуулахад интернетийн хурдаас хамааран 0.2-оос 0.5 секунд хугацаа зарцуулдаг

```

10     else:
11         dict = PCategory(category).__dict__
12         del dict['_sa_instance_state']
13         row.update(dict, synchronize_session=False)
14     session.commit()
15     print(category.id, 'CATEGORY UPSERT DONE')

```

Код 5.3: Upsert to PostgreSQL

Дээрх код нь энгийн python програм файльтай харьцаж өөрт цуглуулсан өгөгдлөө хадгалж байна. Нийт өгөгдлийн хүснэгтийг excel файл болгон энд ⁴ оруулав.

5.1.2 Ажлын байрны өгөгдөл

Zangia.mn ажлын байрны зарын вебсайтаас бакалаврын судалгааны ажлын үечилсэн төлөвлөгөөний дагуу 3 сарын 10-аас эхлэн өгөгдөл цуглуулсан билээ. Цуглуулж, өгөгдлийн сангийн зохиомжийн дагуу дараах утгуудын мэдээллийг CSV файлуудад хадгалж авсан бөгөөд үүнээс цааш бүх өгөгдлийг AWS-EC2 дээр байрших өгөгдлийн санд хадгалж байна. Доорх зурагт хамгийн сүүлд буюу 4 сарын 21нд өгөгдлийн цуглуулга хийж 8900 цуглуулж баазруу хийснийг dataGrip-ийн зургаас харж болж байна. Өгөгдөл цуглуулах нь цаг их шаардлагатай, тогтмол ажиллуулж байхад машиныг асаалттай ашиглаж байх шаардлага тулгарах бөгөөд энэхүү асуудлыг виртуал машин болох amazon-ec2-ээ ашиглаж linux машин дээр *crontab* ажиллуулж 7 хоног тутамд тогтмол өгөгдлийг цуглуулж өөр дээрх баазруу цуглуулах юм. Ингэхдээ эх кодоо github-аараа дамжуулан виртуал машин дээрээ ажиллуулж байгаа болно.

⁴<https://docs.google.com/spreadsheets/d/1rtATUKhUlleIKaWgFGvqiUWMipsrv-aCWZk-tYmzezU/edit?usp=sharing>

id	url	company	title	city	district	exact
1	https://...	БСБ Сервис ХХК	Аялалын менежер	Улаанбаатар хот	Хан-Уул дүүрэг	Монгол
2	https://...	Вуууу технологи ХХК	ERP системийн Нягтлан бодох санхүү бүртгэл...	Улаанбаатар хот	Сүхбаатар дүүрэг	Монгол
3	https://...	MRT Менежмент ХХК	Автын засварчин	Улаанбаатар хот	<null>	Монгол
4	https://...	Юнител Групп	Маркетингийн мэргэжилтэн	Улаанбаатар хот	<null>	Монгол
5	https://...	Сулд Юнайтед ХХК	Борлуулалтын менежер	Улаанбаатар хот	<null>	Монгол
6	https://...	Мөнххада ХХК	Үйлчилгээний зөвлөх	Улаанбаатар хот	Хан-Уул дүүрэг	Монгол
7	https://...	Новел Лизинг ХХК	Зэт энд Эй - Гагнуурчин	Улаанбаатар хот	Сонгинохайрхан дүүрэг	Улаанбаатар
8	https://...	Новел Лизинг ХХК	Зэт энд Эй -Тогоч	Улаанбаатар хот	Сонгинохайрхан дүүрэг	Улаанбаатар
9	https://...	Тера Экспресс ХХК	Засварчин	Өмнөговь аймаг	<null>	Монгол
10	https://...	Ти Ти Си энд Ти ХХК	Гагнуурчин	Өмнөговь аймаг	<null>	Монгол
11	https://...	Эрдэнэт Орчлон ХХК	Нягтлан бодогч ажилд авна	Улаанбаатар хот	<null>	Монгол
12	https://...	Таван Богд Билдинг Салла...	Хангамж үйлчилгээний ажилтан	Улаанбаатар хот	Хан-Уул дүүрэг	Монгол
13	https://...	Гранд Сутай ХХК / 288557...	Сэлбэгийн Нярав	Улаанбаатар хот	<null>	Монгол
14	https://...	Лимитед Моторс	Үйлчлэгч ажилд авна	Улаанбаатар хот	Хан-Уул дүүрэг	Монгол
15	https://...	Монгол Хяндай Автомотив...	Авто засварын төвийн гадаад харилцаа	Улаанбаатар хот	Чингэлтэй дүүрэг	Монгол
16	https://...	Моннис Групп	Үйлчилгээний зөвлөх	Улаанбаатар хот	Сүхбаатар дүүрэг	Монгол
17	https://...	Талх Чихэр ХК	АВТЫН ЗАСВАРЧИН	Улаанбаатар хот	Сонгинохайрхан дүүрэг	<null>
18	https://...	Гэр Контент Групп	Жолооч	Улаанбаатар хот	Баянзүрх дүүрэг	Монгол
19	https://...	СтарЧейз Монголиа	НЯГТЛАН БОДОГЧ	Улаанбаатар хот	Хан-Уул дүүрэг	Монгол
20	https://...	Наран Трейд ХХК	УГТАГЧ + ОВУТАН БИШ	Улаанбаатар хот	Баянзүрх дүүрэг	Монгол
21	https://...	МСМ Групп ХХК	ТУСЛАХ МЕХАНИК-Автомотив салбар	Улаанбаатар хот	Хан-Уул дүүрэг	Монгол
22	https://...	Говь Инфрастратчер Парти...	ТОНОГ ТӨХӨВРӨРЧИЙН ЗАСВАРЧИН	Улаанбаатар хот	Хан-Уул дүүрэг	Монгол
23	https://...	Сутайн буант ХХК	Механик инженер	Улаанбаатар хот	Сонгинохайрхан дүүрэг	Монгол
24	https://...	Новел Лизинг ХХК	Зэт энд Эй - Механик инженер	Улаанбаатар хот	<null>	Улаанбаатар
25	https://...	Сод-Зра ХХК	Тооцооны нягтлан бодогч	Улаанбаатар хот	Баянзүрх дүүрэг	Монгол
26	https://...	Топ моторс ХХК	Засварын механикч	Улаанбаатар хот	Баянгол дүүрэг	Монгол
27	https://...	Орика Монголиа ХХК	Ил уурхайн механик	Өмнөговь аймаг	<null>	Монгол

Зураг 5.2: Data set

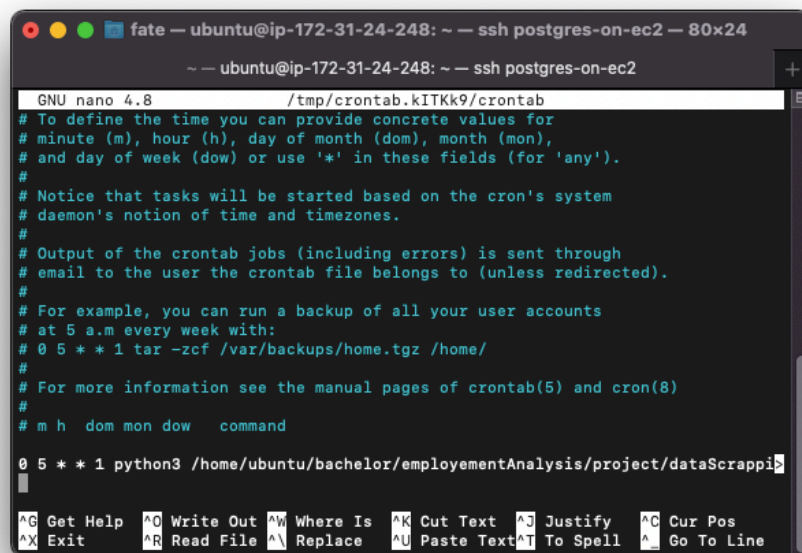
5.1.3 Өгөгдлийг нэгтгэх, цэвэрлэх

Ихэнх цуглуулсан өгөгдөл нь өгөгдлийн сангийн диаграммын дагуу амжилттай цуглуулсан бөгөөд дүн шинжилгээ хийх боломжтой өгөгдлүүдийг тусад нь хадгалж ашигласан болно. Data scrape хийх явцад бүх өгөгдлийг *string* хэлбэрээр цуглуулсан бол өгөгдөлд дүн шинжилгээ хийх явцад энэ нь тохиромжгүй тул цалинг *float* тохиролцох эсэхийг *boolean* болгож өөрчлөв.

```

1 def cleanSalary(salary) -> float:
2     if(isinstance(salary, str)):
3         return float(salary.replace(',',' '))
4     return None
5
6
7 def cleanDealable(deal) -> bool:
8     if(deal == '    '):
9         return True
10    return False
11

```

Зураг 5.3: Виртуал машин дээрх crontab

```
12
13 def cleanLocation(location) -> str:
14     if isinstance(location, str) and location != 'None':
15         return location
```

Код 5.4: Өгөгдөл цэвэрлэх функц

5.1.4 API ажиллуулах

Чатбот системд шаардлагатай RestAPI-ийг express.js ашиглан хөгжүүлсэний дараа backend-ийг виртуал машин дээрээ ажиллуулсан. Ингэхдээ node.js -ийн сан болох *pm2*-ийг ашиглав.

```
zolboo — ubuntu@ip-172-31-24-248: ~ — ssh postgres-on-ec2 — 78x24
...2-31-24-248: ~ — zsh ...  ...sh postgres-on-ec2  ...postgres-on-ec2  +

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

  https://ubuntu.com/aws/pro

36 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

*** System restart required ***
Last login: Sun May 15 22:48:55 2022 from 202.21.108.29
[ubuntu@ip-172-31-24-248:~]$ sudo pm2 list


```

id	name	mode	u	status	cpu	memory
0	index	fork	2	online	0%	60.7mb

```
ubuntu@ip-172-31-24-248:~$
```

Зураг 5.4: Виртуал машин дээрх pm2

5.1.5 Өгөгдлийн статистик

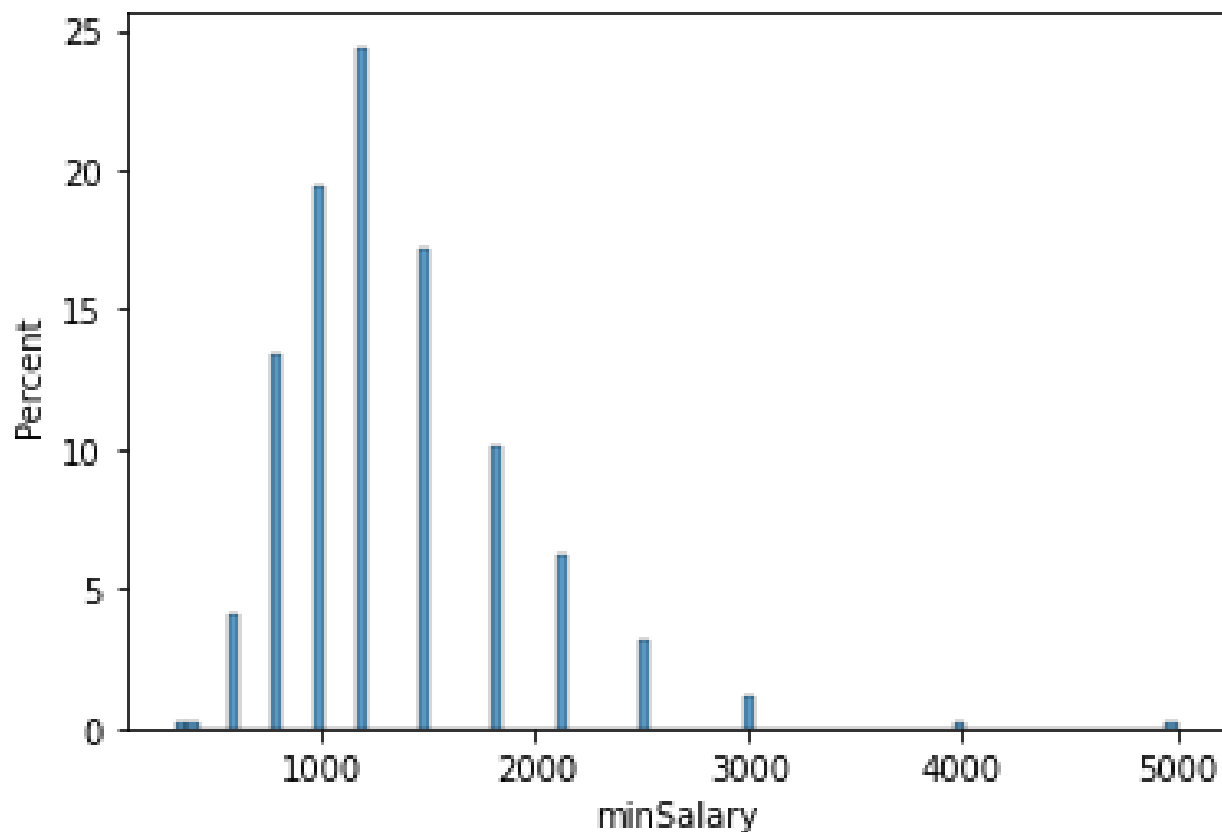
Өөрчлөлт хийсэн өгөгдөл буюу нийт 8202 зарын өгөгдөл дээрээ дүн шинжилгээ хийж, цаашид үүнийгээ чатбот хөгжүүлэлт, асуулт хариултын загварт тусгахыг зорив.

Статистик - 1

Энэхүү графикт нийт цалингийн давхардсан утгууд нийт өгөгдлийн хэдэн хувийг эзэлж байгааг харуулж байна. Үүнээс харвал дундаж цалин 1 сая төгрөгөөс 1.5 сая төгрөгийн хооронд байгааг харж болж байна. Энэ нь нийт цалингийн хувийн 50-аас 55-ийг эзэлж байна.

Статистик - 2

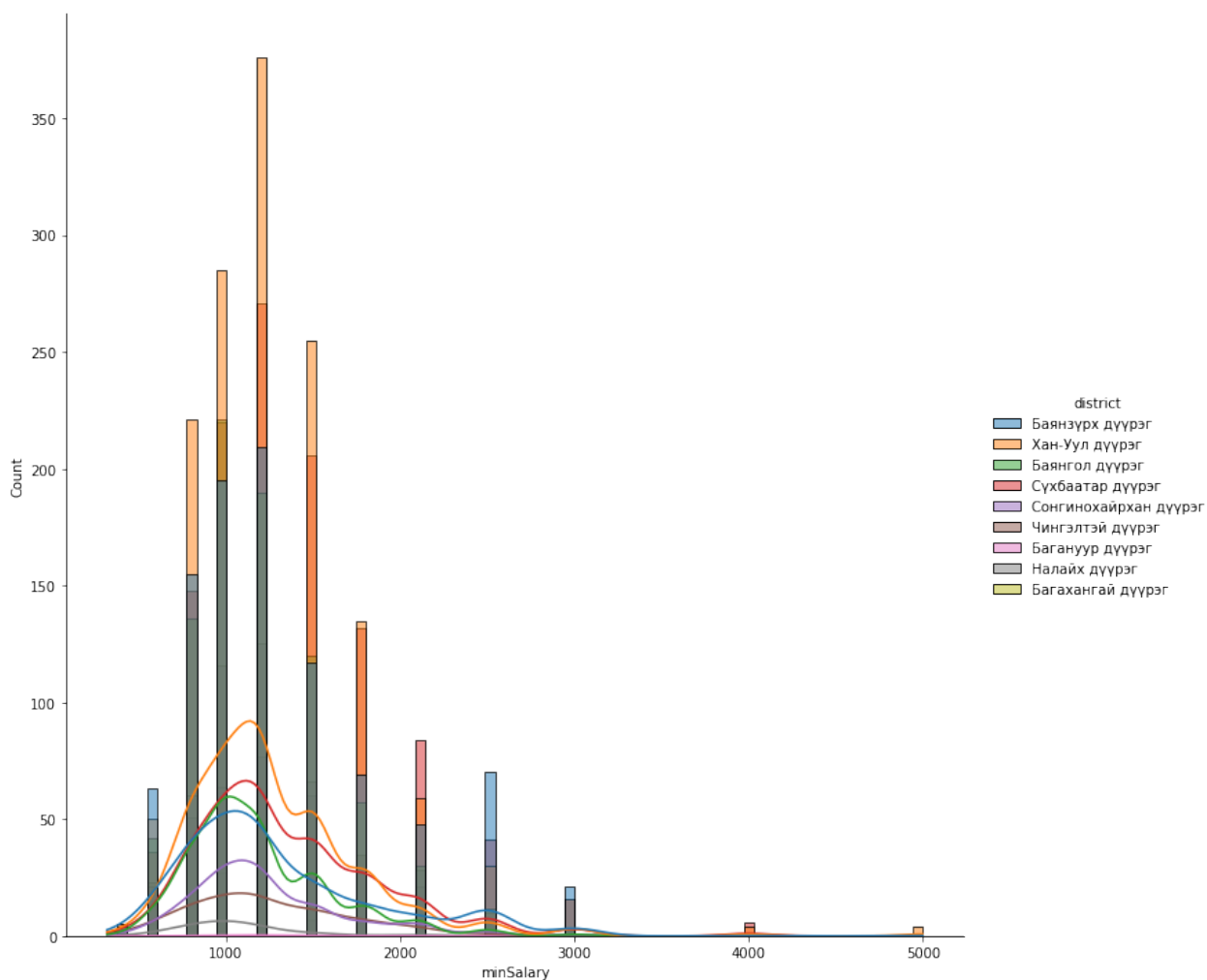
Доорх графикт нийт цалингийн давхардсан утгуудыг тоолж аль дүүрэгт хамгийн их байгааг өнгөөр нь дүрслэн харуулсан байна. Энэхүү графикаас харвал дундаж цалин буюу 1 сая төгрөгөөс 1.5 сая төгрөгөөр цалинжуулах олон ажлын байр санал болгож байгаа дүүрэг нь Хан-Уул болон Сүхбаатар дүүрэг байна. Ажлын байрны төвлөрөл болон их хотын бүтээгдэхүүн үйлдвэрлэлийн цэгийг Хан-Уул, Сүхбаатар дүүрэг гэж дүгнэж болохоор байна.



Зураг 5.5: Өгөгдлийн статистик-1

Статистик - 3

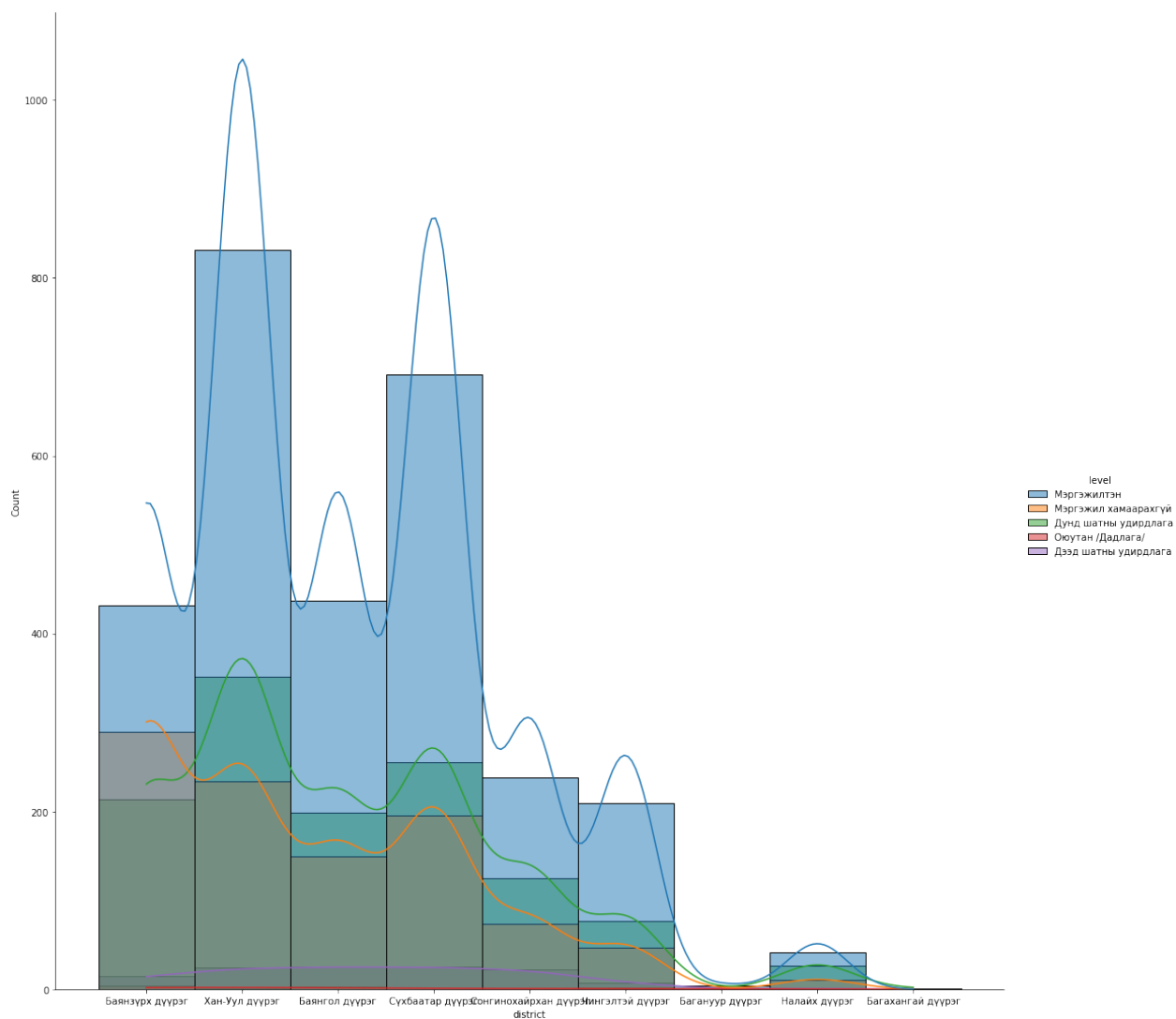
Өмнөх графикийн дүгнэлтийн адилаар хамгийн их ажлын санал болгож буй дүүрэг нь Хан-Уул, Сүхбаатар дүүрэг байх бөгөөд ажлын байрны шаардах түвшинг хамтад нь харуулсан байна. Үүнээс үзвэл, мэргэжилтэн болон дунд шатны удирдлагын орон тоо эрэлттэй байна гэж үзэж болно. Харин дадлагажигч болон дээд шатны удирдлагын эрэлт харьцангуй бага байгааг харж болж байна.



Зураг 5.6: Өгөгдлийн статистик-2

Статистик - 4

Энэхүү графикт дүүргүүд харгалзан ямар төрлийн цагийн хуваарьтай ажил санал болгож байгаа болон тэдгээрийн тоотой нь харьцуулан дүрсэлжээ. Эндээс бүтэн цагийн ажилтан болон ээлжийн төрлийн ажлын байр ихэнх хувийг эзэлж байгааг харлаа.

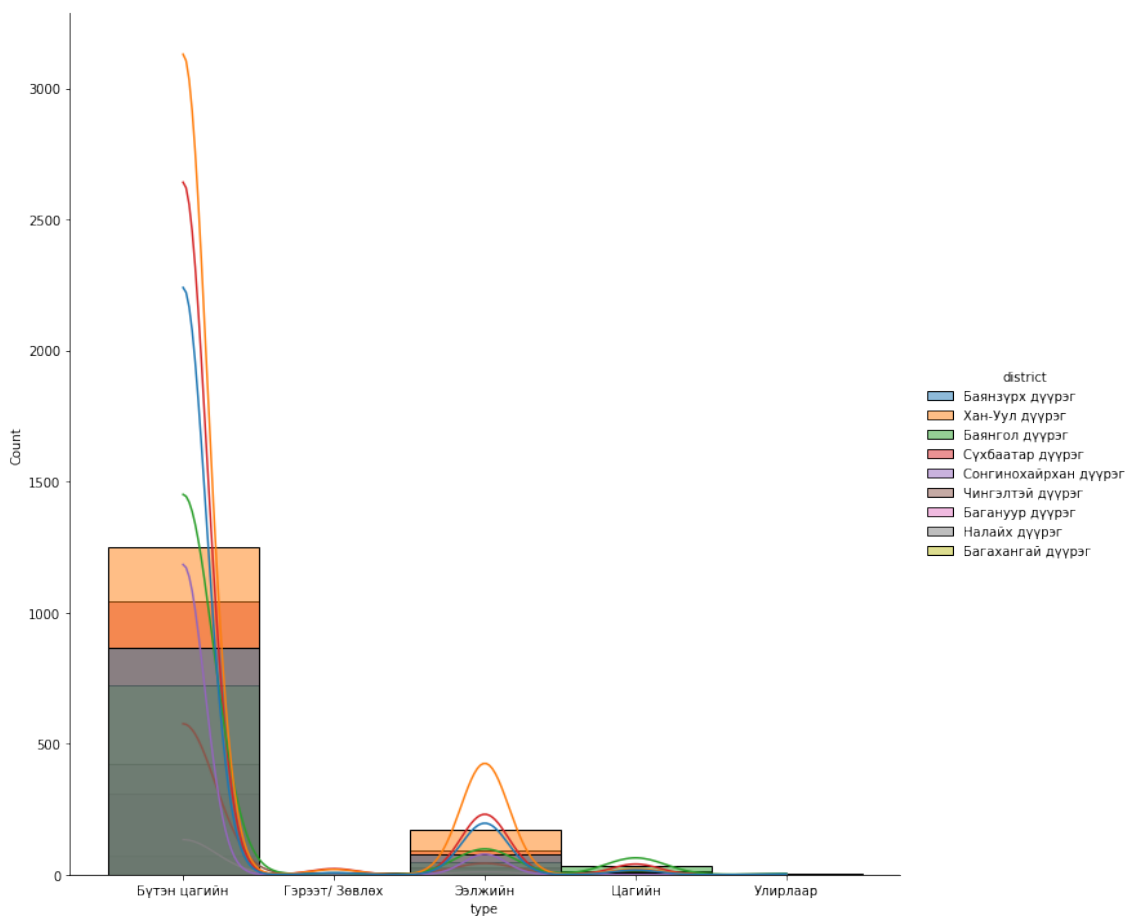


Зураг 5.7: Өгөгдлийн статистик-3

Статистик - 5

Доорх графикаас ажил олгогчид ямар төрлийн цагийн хуваарьтай, хаана ажил санал болгож байгааг 21 аймгаар бүсчлэн өнгөөр илэрхийлсэн байна. Үүнээс дүгнэвэл, Улаанбаатар хотод нягтаршил маш өндөр байгаа бөгөөд ажлын байрны эрэлт аймгуудтай харьцуулахад маш өндөр байна.

Дээрх статисткууд дээр үндэслэн, хэрэглэгчдэд хүртээмжтэй, нийтлэг асуултуудыг дараах байдлаар зохиомжлов.

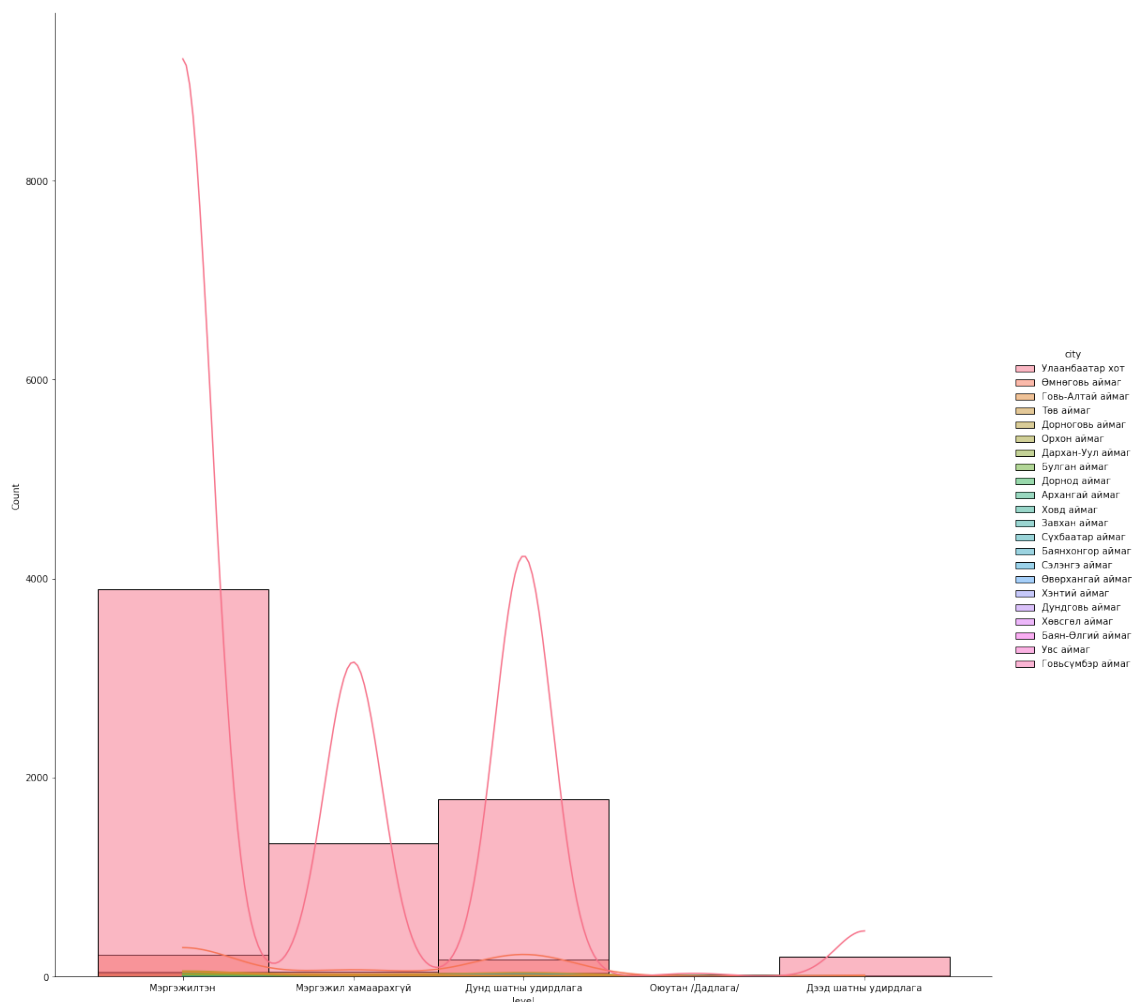


Зураг 5.8: Өгөгдлийн статистик-4

5.1.6 Чатбот хөгжүүлэлт

Чатбот нь хэрэглэгчийн асуултаас онцлох түлхүүр үгшийг шүүж түүнд тохирох API-ын дагуу AWS дээр байрших өгөгдлийн санруу хүсэлт явуулна. API нь RESTful API бөгөөд express.js ажиллаж үр дүнд JSON өгөгдөл авна. Bot Framework нь тэрхүү өгөгдлийг *Adaptive-Card*-ын тусламжтайгаар хэрэглэгчдэд ойлгогдохуйц болгон харуулна. Үүний дараа хэрэглэгчид хариултыг буцаах зарчмаар чатбот систем нь ажиллах юм. Энэхүү процессыг алхам алхамаар хэрэгжүүлж тайлбарлая. Хэрэглэгч чатботтой холбогдох үед чатботын хариулж чадах асуултыг харуулна. Хэрэглэгч өөрийн шаардлагад нийцүүлэн асуултаас сонгож чатботоос асууна.

Эхлээд ашиглагдах REST API-г бэлдэж хэрэглэгчийн асуултад цаг алдалгүй хариулдаг



Зураг 5.9: Өгөгдлийн статистик-5

байх шаардлагатай. Иймд express.js ашиглан PostgreSQL-ээс өгөгдлийг JSON-оор авах API бичиж өгсөн. ApiHeppler код дээр дурдсанчлан виртуал машин дээр ажиллах бөгөөд нь дараах байдалтай байна.

```

1 const categoryRoutes = require('./category/index')
2 const adRoutes = require('./advertisement/index')
3 app.use(express.json())
4
5 app.use('/api/v1/category', categoryRoutes)
6 app.use('/api/v1/ad', adRoutes)

```

Асуултын жагсаалт	
Чатбот нь доорх асуултуудад хариулж чадна.	
№	Асуулт
1	Байгууллагын ажлын байрны дэлгэрэнгүй мэдээлэл. Жишээлбэл: Голомт банк-д дизайнер ажлын байрны мэдээлэл
2	Байгууллагын нээлттэй ажлын байруудын мэдээлэл. Жишээлбэл: Голомт банк-д нээлттэй ажлын байрууд
3	Бүх төрлийн цагийн хуваарьтай ажлын байрны дэлгэрэнгүй мэдээлэл Жишээлбэл: Бүтэн цагийн менежер ажлын байрууд
4	Цалингийн нөхцөлтэй ажлын байрны дэлгэрэнгүй мэдээлэл Жишээлбэл: 3000000 цалинтай хөгжүүлэгч ажлын байр

Зураг 5.10: Чатботын хариулж чадах асуултууд

```

7
8 app.listen(4000, () => {
9   console.log('Server is listening in port 4000')
10 })

```

Код 5.5: REST API controller

Үүний дараа хэрэглэгчийн асуусан асуултад хариулахад бэлэн болох бөгөөд дараах код нь хэрэглэгчийн асуултыг таньж query үүсгэхэд туслана. Асуултын төрлийг таньсаны дараа асуултаас шаардлагатай query-дэх түлхүүр үгийг олж авна.

```

1  getTypеOfQuestion(text) {
2    if (this.contextText.search('          ') != -1) return 1
3    else if (this.contextText.search('      ') != -1) return 2
4    else if (this.contextText.search('    ') != -1) return 3
5    else if (this.contextText.search('  ') != -1) return 4
6    else return 404

```



```
7 }
```

Код 5.6: Question-understand классын кодын жишээ 1

```
1  
2 findKeyWord() {  
3     switch (this.getTypeOfQuestion()) {  
4         case 1:  
5             this.queryNumber = 1  
6             var keyword = this.contextText.substring(0, this.contextText.  
7                 search(' ')).trim().split(' -')  
8             return keyword  
9             break
```

Код 5.7: Question-understand классын кодын жишээ 2

Ингэж query параметрийг түлхүүр үгийн тусламжтайгаар тодорхойлсны дараа API-аар хандаж өгөгдлийг авахад бэлэн болно. Үүний үр дүнд өгөгдлийн санруу явах query бэлэн болно.

```
1 getQuery() {  
2     switch (this.questionNumber) {  
3         case 1:  
4             return util.format('/ad/company=%s&title=%s', this.keyword[0].  
5                 trim().replace(' ', '%20'), this.keyword[1].trim().replace(' ', '%20'))  
6             break
```

Код 5.8: query бэлтгэх кодын жишнээ

Дараах кодын хэсэг нь дотоод орчинд ажиллаж буй REST API controller-ыг ашиглан хүсэлт явуулж байна. Үр дүнд нь асуултын хариулт болох JSON объектыг авч байна.

```

1  async responseBack() {
2      var query = this.getQuery()
3      console.log('query:', query)
4      if (query === 404) {
5          return undefined
6      }
7      var query = this.getQuery()
8      const host = 'http://3.228.127.116'
9      const port = ':4000'
10     const path = '/api/v1'
11     const url = host + port + path + query
12     try {
13         var res = await fetch(url)
14     } catch (error) {
15         console.error(error)
16     }
17     if (!res.ok) {
18         throw res
19     }
20     var responseBody = await res.json()
21     return responseBody

```

Код 5.9: JSON хүлээж авах кодын жишээ.

Тухайн үр дүнгээ хэрэглэгчид ойлгогдохоор харуулах шаардлагатай. Үүний тулд Microsoft Bot Framework-ийн картын төрлүүдийг ашигласан. Карт нь олон төрлийн хувилбаруудтай бөгөөд тэр дундаас ”Adaptive Cards”-ыг сонгосон.

```

1  const d = new Date()
2  d.toLocaleDateString

```

```

3   for (let index = 0; index < this.body.length; index++) {
4       var cardData = {
5           "type": "AdaptiveCard",
6           "$schema": "http://adaptivecards.io/schemas/adaptive-card.json"
7           ,
8           "version": "1.2",
9           "body": [],
10          "actions": []
11      }
12      const element = this.body[index]
13      cardData["body"].push({
14          "type": "TextBlock",
15          "size": "Medium",
16          "weight": "Bold",
17          "text": element['title'],
18          "wrap": true,
19          "style": "heading"
20      }, {
21          "type": "ColumnSet",
22          "columns": [{
23              "type": "Column",
24              "items": [
25                  {
26                      "type": "Image",
27                      "style": "Person",
28                      "url": "https://cdn-icons-png.flaticon.com/512/622/622848.png",
29                      "altText": element['company'],

```

```

29         "size": "Small"
30     }
31 ],
32     "width": "auto"
33 }, , {
34     "type": "Column",
35     "items": [{
36         "type": "TextBlock",
37         "weight": "Bolder",
38         "text": element['company'],
39         "wrap": true
40     }, {
41         "type": "TextBlock",
42         "spacing": "None",
43         "text": "          " + new Date(element['publishedDate']).
44             toLocaleDateString('zh-Hans-CN'),
45         "isSubtle": true,
46         "wrap": true
47     }],
48     "width": "stretch"
49 }],
50     "type": "TextBlock",
51     "text": element['roles'],
52     "maxLines": 3,
53     "wrap": true
54 }, {
55     "type": "FactSet",

```

```

56     "facts": [{
57         "title": "    :",
58         "value": (element['city']) ? element['city'] : ' ' + ' ' + (
                    element['district']) ? element['distirct'] : ' '
59     }, {
60         "title": "    :",
61         "value": (element['maxSalary'] || element['minSalary']) ?
                    element['minSalary'] + ' - ' + element['maxSalary'] : '
                    '
62     }, {
63         "title": "    :",
64         "value": element['types']
65     }, {
66         "title": "    :",
67         "value": element['phoneNumber']
68     }]
69 }
70 )
71 cardData['actions'].push({
72     "type": "Action.OpenUrl",
73     "title": "    ",
74     "url": element['url']
75 })
76 ret.push(CardFactory.adaptiveCard(cardData))
77 }
78 return ret
79 };

```

Код 5.10: Хэрэглэгчийн UI дүрслэх кодын жишээ.

Хэрэглэгчид харагдах байдлыг өөрийн хүссэнээр уян хатан зохион байгуулах боломжийг ”Adaptive Cards” олгодог бөгөөд эмх цэгцтэй мэдээллийг хүргэхэд туслах юм. AdaptiveCard нь JSON хэлбэртэй өгөгдлийн хадгалдаг бөгөөд үндсэн 5 төрлийн өгөгдөлтэй байна. Үүнд:

- type: Microsoft Botframework нь өөр олон төрлийн картуудтай бөгөөд энд тэдгээрийг зааж өгнө.
- schema: Энд картын эх буюу схемийн хаягийг тодорхойлно.
- version: Сонгосон төрлийн картын ямар хувилбарыг ашиглах болохыг зааж өгнө.
- body: Энд карт доторх бүх контент буюу мэдээллүүд байна.
- actions: Хэрэв карт нь ямар нэг төрлийн товчтой байвал тэдгээр нь юу хийх талаар програмчлана. (OnClick() г.м)

5.2 Үр дүн

Энэхүү хэсэгт асуулт бүрийн хэрэгжүүлэлтийн хэрэглэгчдэд харагдах байдлыг дүрсэлсэн болно.

5.2.1 Угтах текст

Угтах текстийг шинэ хэрэглэгч болон чатлаагүй удсан хэрэглэгчдэд харуулна. Энэ нь хэрэглэгчийн session дээр суурилсан байна.

5.2.2 Байгууллагын ажлын байрны дэлгэрэнгүй мэдээлэл

Ажиллахыг хүссэн байгууллын нэрийг ажлын байрны нэрийн хамтаар асуух нь хамгийн нийтлэг байх бөгөөд тухайн байгууллагын ажлын байрны нэрээр хайлт хийж хэрэглэгчдэд дэлгэрэнгүй мэдээллийг харуулна.

- Голомт Банк-д дизайнер ажлын байрны дэлгэрэнгүй мэдээлэл асуултын хариулт хэрэглэгчид харагдах байдал:

Сайн байна уу! Та ажлын байрны туслах чатботтой холбогдлоо. 😊 Хүссэн ажлын байрны мэдээллийг цаг алдалгүй аваарай

Асуултын жагсаалт


Чатбот нь доорх асуултуудад хариулж чадна.

№	Асуулт
1	Байгууллагын ажлын байрны дэлгэрэнгүй мэдээлэл. Жишээлбэл: Голомт банк-д дизайнер ажлын байрны мэдээлэл
2	Байгууллагын нээлттэй ажлын байруудын мэдээлэл. Жишээлбэл: Голомт банк-д нээлттэй ажлын байрууд
3	Бүх төрлийн цагийн хуваарьтай ажлын байрны дэлгэрэнгүй мэдээлэл Жишээлбэл: Бүтэн цагийн менежер ажлын байрууд
4	Цалингийн нөхцөлтэй ажлын байрны дэлгэрэнгүй мэдээлэл Жишээлбэл: 3000000 цалинтай хөгжүүлэгч ажлын байр

Зураг 5.11: Угтах текст

Хэрэв тухайн байгууллагад хайсан ажлын байрны нээлттэй зарын мэдээлэл байхгүй тохиолдолд байгууллагын нээлттэй ажлын байруудын мэдээллийг дүрслэнэ.

ВИДЕО ГРАФИК ДИЗАЙНЕР


 Голомт Банк
Нийтлэгдсэн 2022/3/28

Зорилтод зах зээлд чиглэсэн PR, идэвхжүүлэлт, сурталчилгааны үйл ажиллагааг хэрэгжүүлэхийн хүрээнд бүх видео сурталчилгааны материал, шийдлийг гарган бэлтгэх.

Хот, Дүүрэг: Улаанбаатар хот
Цалин: 1500000 - 1800000
Төрөл: Бүтэн цагийн
Утас: 75751111, 70111646-1705

[Дэлгэрэнгүй харах](#)

ВИДЕО ГРАФИК ДИЗАЙНЕР

 Голомт Банк
Нийтлэгдсэн 2022/5/2

Зорилтод зах зээлд чиглэсэн PR, идэвхжүүлэлт, сурталчилгааны үйл ажиллагааг хэрэгжүүлэхийн хүрээнд бүх видео сурталчилгааны материал, шийдлийг гарган бэлтгэх.

Хот, Дүүрэг: Улаанбаатар хот
Цалин: 1800000 - 2100000
Төрөл: Бүтэн цагийн
Утас: 75751111, 70111646-1705

[Дэлгэрэнгүй харах](#)

Just now

Зураг 5.12: <Байгууллага>-д <ажил> ажлын байрны мэдээлэл?

5.2.3 Байгууллагын нээлттэй ажлын байруудын мэдээлэл

Ажиллахыг хүссэн байгууллын нэрийн дагуу нээлттэй ажлын байруудыг харах боломжтой.







- Голомт Банк-д нээлттэй ажлын байруудын мэдээлэл асуултын хариулт хэрэглэгчдэд харагдах байдал:

Уучлаарай, Голомт банк байгууллагад хамгаалагч ажлын байр олдсонгүй.

Голомт банк-д нээлттэй ажлын байрууд:

Нийт: 23 ажлын байр байна

activity








	Голомт Банк Зээлийн бодлогын мэргэжилтэн 1200000 - 1500000
	Голомт Банк ВИДЕО ГРАФИК ДИЗАЙНЕР 1500000 - 1800000
	Голомт Банк Архивын эрхлэгч 1000000 - 1200000
	Голомт Банк БОРЛУУЛАЛТЫН МЕНЕЖЕР 1200000 - 1500000
	Голомт Банк Туслах ажилтан 1200000 - 1500000
	Голомт Банк Хүний нөөцийн менежер 1200000 - 1500000

Зураг 5.13: Бусад ажлын байрууд

5.2.4 Бүх төрлийн цагийн хуваарьтай ажлын байрны дэлгэрэнгүй мэдээлэл

Ажиллахыг хүссэн ажлын байрны дагуу ажиллах цагийн хувиариар шүүж харах боломжтой байна. Бүх боломжит ажлын байруудын дэлгэрэнгүй мэдээллийг харуулна.

- Бүтэн цагийн тогооч ажлын байр асуултын хариулт хэрэглэгчдэд харагдах байдал:

Голомт банк-д нээлттэй ажлын байрууд:	
Нийт: 23 ажлын байр байна	
	Голомт Банк Зээлийн бодлогын мэргэжилтэн 1200000 - 1500000
	Голомт Банк ВИДЕО ГРАФИК ДИЗАЙНЕР 1500000 - 1800000
	Голомт Банк Архивын эрхлэгч 1000000 - 1200000
	Голомт Банк БОРЛУУЛАЛТЫН МЕНЕЖЕР 1200000 - 1500000
	Голомт Банк Туслах ажилтан 1200000 - 1500000
	Голомт Банк Хүний нөөцийн менежер 1200000 - 1500000
	Голомт Банк Слесарь 600000 - 800000

Зураг 5.14: <Байгууллага>-д нээлттэй ажлын байрууд?

5.2.5 Цалингийн нөхцөлтэй ажлын байрны дэлгэрэнгүй мэдээлэл

Ажиллахыг хүссэн ажлын байрны дагуу цалингийн хэмжээгээр шүүж харах боломжтой байна. Бүх боломжит ажлын байрыг жагсаалт хэлбэрээр харуулна.

- Бүтэн цагийн тогооч ажлын байр асуултын хариулт хэрэглэгчдэд харагдах байдал:

Бүтэн цагийн тогооч ажлын байр

3 minutes ago

БАЙРЛАЖ АЖИЛЛАХ ТУСЛАХ ТОГООЧ



Гэрэлт ассемоур трейдинг ХХК
Нийтлэгдсэн 2022/4/21

Ажлын байрны тодорхойлолтонд заасны дагуух ажил үүрэг, гүйцэтгэнэ.

Хот, Дүүрэг:

Цалин: 1000000 - 1200000

Төрөл: Бүтэн цагийн

Утас: 80948030, 77778030

[Дэлгэрэнгүй харах](#)

Ажилчдын хоолны туслах тогооч



Ситимент ХХК
Нийтлэгдсэн 2022/4/4

Ажлын байрны тодорхойлолтын дагуу

Хот, Дүүрэг: Улаанбаатар хот

Цалин: 600000 - 800000

Төрөл: Бүтэн цагийн

Утас: 70187591, 91995196, 80128851

[Дэлгэрэнгүй харах](#)

Зураг 5.15: <Ажиллах цаг> цагийн <ажил> ажлын байр?

3000000 цалинтай хөгжүүлэгч ажлын байр

Just now

3000000 -аас өндөр цалинтай нээлттэй ажлын байрууд:

Нийт: 18 ажлын байр байна



Таван Богд Финанс ХХК
UX/UI хөгжүүлэгч
3000000 - 4000000



Миний Дэлгүүр ХХК
АХЛАХ ПРОГРАММ ХӨГЖҮҮЛЭГЧ
3000000 - 4000000



Монос Групп
Ахлах Мобайл хөгжүүлэгч
3000000 - 4000000



Инновэйшн Инвестмент ХХК
Ахлах программ хөгжүүлэгч
3000000 - 4000000



"Стор Пэй"ХХК
FRONTEND ХӨГЖҮҮЛЭГЧ
3000000 - 4000000



Гүнд инвестмент холдинг групп
Ахлах програм хөгжүүлэгч - BrightOn LLC /SAM App/
3000000 - 4000000



Миний Дэлгүүр ХХК
АХЛАХ ПРОГРАММ ХӨГЖҮҮЛЭГЧ
3000000 - 4000000

Зураг 5.16: <Цалин> цалинтай <ажил> ажлын байр?

Дүгнэлт

Бакалаврын судалгааны ажлын хүрээнд ажил хайгчдын хэрэгцээнд нийцсэн, хялбар байдлаар ажил хайх боломжтой “Ажил олгогчдын өгөгдөлд суурилсан чатбот” -ыг хөгжүүлэхээр зорилго тавин зорилтуудын дагуу ажлуудыг хийж гүйцэтгэлээ.

Системийг хөгжүүлэхдээ уламжлалт холбоост өгөгдлийн сан(relational database) санг ашиглаж нийт өгөгдлийг zangia.mn-ээс цуглуулан ашиглав. Системийг Microsoft Azure платформын Bot Framework-д ашиглан хөгжүүлсэн бөгөөд хэрэглэгчдэд хүргэхэд олон төрлийн сувгийг ашиглах боломжийг олгон хөгжүүлэв.

Энэхүү програм нь хэрэглэгч ашиглаж эхэлснээс дуустал бүрэн хүний оролцоогүй бөгөөд өгөгдлүүдийг олборлох процесийг мөн автоматжуулав.

Цаашид хэрэглэхэд хялбар байдалд анхаарч, асуултад эх хэлний боловсруулалтын тусламжтайгаар хариулах, түгээмэл болон, шаардлагатай асуултуудыг хэрэглэгчийн туршлагад үндэслэн сайжруулалт, хөгжүүлэлт хийхэд анхаарал хандуулсаар байх болно.

Bibliography

- [1] Чатбот системийн тухай
<https://www.engati.com/blog/types-of-chatbots-and-their-applications>
- [2] Өгүүлбэр хувиргалтын арга зүй
<https://www.sbert.net/docs/quickstart.html>
- [3] Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks
<https://arxiv.org/abs/1908.10084>
- [4] Use case diagram
https://app.diagrams.net/#G1jhom3sc_holt-X9XLALtQja_Gl_Eykhj
- [5] Business Process Model Notation 2.0 диаграмм
<https://cawemo.com/diagrams/ea037ec0-c1c5-4ab6-8262-521657472803--bpmn-2-0?v=960,418,1>
- [6] Өгөгдлийн сангийн диаграмм
<https://dbdiagram.io/d/6249fb7cd043196e39e87451>

А. ҮЕЧИЛСЭН ТӨЛӨВЛӨГӨӨ

Батлаа.

МКУТ-ийн эрхлэгч:...../док. проф. Н.Оюун-Эрдэнэ/

2022 оны 02 сарын 11

Монгол нэр

Ажил олгогчдын өгөгдлийн анализ систем дээр суурилсан чат бот

Англи нэр

Chat bot based on system analysis of employers' data

Сэдэвт бакалаврын судалгааны ажлын 7 хоногийн үеичлэсэн төлөвлөгөө

Хугацаа: 2022.02.07-оос 2022.05.06 хүртэл 13 долоо хоног

№	Хийх ажил	Долоо хоног											14 Жинхэнэ хамгаалалт	Тайлбар	
	Онолын судалгаа	1	2	3	4 Явц I	5	6	7	8 Явц II	9	10	11	12 Урьдчилсан хамгаалалт	13	
1	Scrapper tool Bot tool														
2	Цуглуулах код бичих Өгөгдлийг бааруу нуулуулах														
3	Системийн шаардлага тодорхойлох Хэрэглэгчийн шаардлага тодорхойлох														
5	Системийн зохиомж Өгөгдлийн сангийн зохиомж Чат бот хөгжүүлэлт														
6	Хэрэгжүүлэлт Өгөгдөлд анализ хийх сайжруулах														
7	Бичиг баримт Тайлан боловсруулах														

Тайлбар: Төслийг гэрээжүүлэх төлөвлөгөөг 7 хоногийн дотоод хамгаалалтаар хийж төв гарвар будамж илмдэгдэнэ. Хийх ажил дэд хэсгийн байдал үз ажилд зарцуулах хугацааг хувиар тэмдэглэж байна. Ажлын гэрээжүүлэхийг дутууж үзвэл хэсгээр хийх боломжтой байхад "7 хоног" багасныг үүсгэнэ.

Зөвшөөрсөн: Удирдагч багшБ. Хужаабаатар/

Боловсруулсан: Оюутин/Мэдээллийн технологи А. Сайнболбоо/

Оюутны ID: 1851num1762

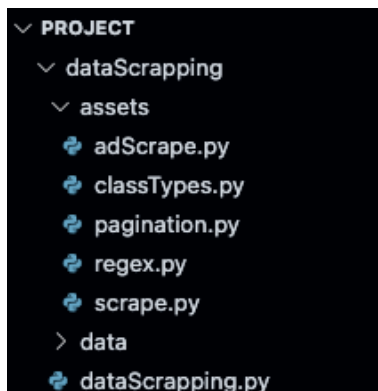
Холбогдох утас: 91990388

Зураг А.1: Бакалаврын судалгааны ажлын үеичлэсэн төлөвлөгөө

В. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

В.1 Өгөгдөл цуглуулалт

Өгөгдөл цуглуулах програм нь дараах бүтэцтэй байх бөгөөд assets доторх кодууд нь үндсэн кодыг ажлуулахад туслах функцууд байна.



Зураг В.1: Фолдерийн бүтэц

В.1.1 Үндсэн өгөгдлийг цуглуулах

```
1 from assets.classTypes import Advertisement, Category
2 from assets.scrape import UseBeautifulSoup as useScrape
3 from assets.adScrape import advertisementScrape as useAdScrape
4 from assets.splitter import createLinkList, splitUrl
5 from connection import Base, db, session
6 from insert import upsertAdvertisement, upsertCategory
7
8 initialUrl = 'https://www.zangia.mn/'
9 categorySet = set()
10 adUrlDict = {}
11
12 soup = useScrape(initialUrl)
13 navigatorList = soup.find_all('div', class_='filter')
14 for navigator in navigatorList:
15     if navigator.find('h3').text.strip() != 'Salbar, mergejil':
16         continue
17     categoryList = navigator.find_all('div')
18
19 for categoryItem in categoryList:
20     categories = categoryItem.find('a')
21     url = initialUrl + categories['href']
22     tempCategory = Category(splitUrl(url, 'b.'), url, categories.text)
23     print('CATEGORY LINK SCRAPED! ', url, tempCategory.id)
24     soup = useScrape(url)
25     subCategory = soup.find('div', class_='pros')
```



```

26     subCategoryList = subCategory.find_all('a')
27     for subCategoryItem in subCategoryList:
28         subCategoryUrl = initialUrl + subCategoryItem['href']
29         tempSubCategory = Category(splitUrl(subCategoryUrl, 'r.'),
30                                     subCategoryUrl, subCategoryItem.text
                                     , tempCategory)
31         categorySet.add(tempSubCategory)
32     categorySet.add(tempCategory)
33
34     for categoryItem in categorySet:
35         upsertCategory(categoryItem)
36         if categoryItem.parentCategory == None:
37             continue
38         soup = useScrape(categoryItem.url)
39         hasPagination = soup.find('div', class_='page-link')
40         pagesUrl = []
41         if hasPagination != None:
42             pagesUrl = createLinkList(hasPagination, categoryItem.url)
43         else:
44             pagesUrl.append(categoryItem.url)
45         for pageUrl in pagesUrl:
46             soup = useScrape(pageUrl)
47             ads = soup.find_all('div', class_='ad')
48             for ad in ads:
49                 adUrl = initialUrl+ad.find('a', class_=None)['href']
50                 adUrlDict[adUrl] = categoryItem
51     print(pagesUrl)
52     pagesUrl.clear()
53
54     for adUrl in adUrlDict:
55         tempAdItem = useAdScrape(adUrl)
56         tempAdItem.setCategory(adUrlDict[adUrl])
57         tempAdItem.setId(splitUrl(adUrl, 'ad'))
58         try:
59             upsertAdvertisement(tempAdItem, tempAdItem.category)
60         except:
61             print('Write to db error')
62     del tempAdItem

```

Код В.1: Бүх өгөгдлийг цуглуулах - dataScraping.py

В.1.2 Нэг зарын шаардлагатай бүх мэдээллийг цуглуулах

```

1 import re
2 from .classTypes import Advertisement
3 from .scrape import UseBeautifulSoup as useScrape
4 from .cleanData import cleanAdObject
5
6
7 def listScrapper(sections, key) -> str:
8     content = []
9     for section in sections:

```

```

10     subTitle = section.find('h2', class_=None).text
11     if key != subTitle:
12         continue
13     div = section.find('div', class_=None)
14     children = div.next_element
15
16     while(children != None):
17         try:
18             content.append(textStrip(children.text))
19             children = children.next_sibling
20             continue
21         except:
22             print('An error occurred')
23             children = children.next_sibling
24     content = [s for s in filter(listFunc, content)]
25     if not content:
26         return ''
27     return ' '.join(content)
28
29
30 def textStrip(text) -> str:
31     pattern = re.compile('[\r\n\xa0\t ]+', re.MULTILINE | re.IGNORECASE)
32     return pattern.sub(' ', text.strip())
33
34
35 def listFunc(e):
36     return len(e) != 0
37
38
39 def singleItemScraper(sections, key, subKey) -> str:
40     for section in sections:
41         subTitle = section.find('h2', class_=None).text
42         if key != subTitle:
43             continue
44         div = section.find_all('div', class_=None)
45         for item in div:
46             if item.next_element.text == subKey:
47                 return textStrip(item.find('span').text)
48     return 'None'
49
50
51 def salaryScraper(salary):
52     isDealable = ''
53     k = re.split(r'[^\d,]+', salary, 2, re.IGNORECASE)
54     if len(k) < 2:
55         [a] = k[0]
56         return a, a, isDealable
57     if k[1] == '':
58         a = k[0]
59         return a, a, isDealable
60     [a, b] = k[0:2]

```

```

61     if len(k) > 2:
62         isDealable = ' '
63     return a, b, isDealable
64
65
66 def locationScrapper(location):
67     city = ''
68     district = ''
69     k = location.split(',')
70     if len(k) < 2:
71         city = k[0]
72         return city, district
73     [city, district] = k[0:2]
74     return city, district
75
76
77 def advertisementScrape(url) -> Advertisement:
78     soup = useScrape(url)
79     advertisement = Advertisement(url, soup.find('h3').text.strip())
80     try:
81         companyTitle = soup.find('div', class_='nlp').find('td')
82         for item in companyTitle:
83             if item.name == None:
84                 advertisement.company = textStrip(item.text)
85     except:
86         print('Company name scrape error')
87
88     sections = soup.find_all('div', class_='section')
89     # all items
90     advertisement.level = singleItemScrapper(sections, ' ', ' ')
91     advertisement.type = singleItemScrapper(sections, ' ', ' ')
92     minSalary, maxSalary, isDealable = salaryScrapper(
93         singleItemScrapper(sections, ' ', ' '))
94     advertisement.setSalary(minSalary, maxSalary, isDealable)
95     city, district = locationScrapper(
96         singleItemScrapper(sections, ' ', ' '))
97     advertisement.location.city = city
98     advertisement.location.district = district
99     advertisement.location.exactAddress = singleItemScrapper(
100         sections, ' ', ' ')
101     advertisement.roles = listScrapper(
102         sections, ' ')
103     advertisement.requirements = listScrapper(
104         sections, ' ')
105     advertisement.additionalInfo = listScrapper(
106         sections, ' ')
107     advertisement.contact.phoneNumber = singleItemScrapper(
108         sections, ' ', ' ')
109     advertisement.contact.fax = singleItemScrapper(
110         sections, ' ', ' ')
111     advertisement.adAddedDate = singleItemScrapper(
112         sections, ' ', ' ')

```

```
113     return cleanAdObject(advertisement)
```

Код В.2: Нэг зарын өгөгдлийг цуглуулах - adScrape.py

В.1.3 Цуглуулах өгөгдлийн төрөл

```
1  class Category:
2      id = ''
3      url = ''
4      name = ''
5
6      def __init__(self, id, url, name, parentCategory=None) -> None:
7          self.id = id
8          self.url = url
9          self.name = name
10         self.parentCategory = parentCategory
11
12     def getUrl(self) -> str:
13         return self.url
14
15
16 class Location:
17     city = ''
18     district = ''
19     exactAddress = ''
20
21     def __init__(self, city=None, district=None, exactAddress=None) ->
22         None:
23         self.city = city
24         self.district = district
25         self.exactAddress = exactAddress
26
27 class Contact:
28     phoneNumber = ''
29     fax = ''
30
31     def __init__(self, phoneNumber=None, fax=None) -> None:
32         self.phoneNumber = phoneNumber
33         self.fax = fax
34
35
36 class Advertisement:
37     id = ''
38     category = Category
39     url = ''
40     company = ''
41     title = ''
42     level = ''
43     type = ''
44     minSalary = ''
45     maxSalary = ''
```

```

46     isDealable = ''
47     location = Location
48     roles = ''
49     requirements = ''
50     additionalInfo = ''
51     contact = Contact
52     adAddedDate = ''
53
54     def __init__(self, url, title) -> None:
55         self.url = url
56         self.title = title
57
58     def setId(self, id) -> None:
59         self.id = id
60
61     def setSalary(self, minSalary, maxSalary, isDealable) -> None:
62         self.minSalary = minSalary
63         self.maxSalary = maxSalary
64         self.isDealable = isDealable
65
66     def setCategory(self, category) -> None:
67         self.category = category
68
69     def setLocation(self, location) -> None:
70         self.location = location
71
72     def setContact(self, contact) -> None:
73         self.contact = contact

```

Код В.3: Өгөгдлийн төрөл - classTypes.py

B.1.4 BeautifulSoup scraper

```

1  from bs4 import BeautifulSoup
2  import requests
3  from urllib.error import HTTPError
4
5
6  def UseBeautifulSoup(url):
7      try:
8          response = requests.get(url)
9          response.raise_for_status()
10     except HTTPError as error:
11         print(error)
12     soup = BeautifulSoup(response.text, 'html.parser')
13     return soup

```

Код В.4: Scrape хийх функц - scrape.py

В.2 Өгөгдөл нэгтгэх, цэвэрлэх

```

1  import pandas as pd
2  from .classTypes import Advertisement

```

```

3 from datetime import datetime
4
5
6 def cleanSalary(salary) -> float:
7     if isinstance(salary, str) and salary != '' and salary != 'None':
8         return float(salary.replace(',', ''))
9     return None
10
11
12 def cleanDealable(deal) -> bool:
13     if deal == '':
14         return True
15     return False
16
17
18 def cleanNone(text) -> str:
19     if isinstance(text, str) and text != 'None' and text != '':
20         return text
21     return None
22
23 def cleanAdObject(advertisement: Advertisement) -> Advertisement:
24     advertisement.level = cleanNone(advertisement.level)
25     advertisement.type = cleanNone(advertisement.type)
26     advertisement.minSalary = cleanSalary(advertisement.minSalary)
27     advertisement.maxSalary = cleanSalary(advertisement.maxSalary)
28     advertisement.isDealable = cleanDealable(advertisement.isDealable)
29     advertisement.location.city = cleanNone(advertisement.location.city)
30     advertisement.location.district = cleanNone(
31         advertisement.location.district)
32     advertisement.location.exactAddress = cleanNone(
33         advertisement.location.exactAddress)
34     advertisement.roles = cleanNone(advertisement.roles)
35     advertisement.requirements = cleanNone(advertisement.requirements)
36     advertisement.additionalInfo = cleanNone(advertisement.
37         additionalInfo)
38     advertisement.contact.phoneNumber = cleanNone(
39         advertisement.contact.phoneNumber)
40     advertisement.contact.fax = cleanNone(advertisement.contact.fax)
41     advertisement.adAddedDate = cleanNone(advertisement.adAddedDate)
42     return advertisement

```

Код В.5: Өгөгдөл цэвэрлэх - dataClean.py

В.3 Өгөгдлийн сангийн холболт

В.3.1 Өгөгдлийн санг удирдах

ORM ашиглан виртуал машин дээрх өгөгдлийн сантай харьцна.

```

1 from sqlalchemy import create_engine
2 from sqlalchemy.orm import sessionmaker

```

```

3 from sqlalchemy_utils import database_exists, create_database
4 from sqlalchemy.ext.declarative import declarative_base
5 from local_settings import postgresql as settings
6 import logging
7
8 log = logging.getLogger(__name__)
9
10
11 def get_database():
12     try:
13         engine = get_engine_from_settings()
14         log.info("Connected to PostgreSQL database!")
15     except:
16         log.exception("Failed to get database connection!")
17         return None, 'fail'
18
19     return engine
20
21
22 def get_engine_from_settings():
23     keys = ['user', 'password', 'host', 'port', 'db']
24     if not all(key in settings for key in keys):
25         raise Exception('Bad config file')
26
27     return get_engine(settings['user'],
28                       settings['password'],
29                       settings['host'],
30                       settings['port'],
31                       settings['db'])
32
33
34 def get_engine(user, password, host, port, db):
35     url = f"postgresql://{user}:{password}@{host}:{port}/{db}"
36     if not database_exists(url):
37         create_database(url)
38     engine = create_engine(url, pool_size=50, echo=False)
39     return engine
40
41
42 def get_session():
43     engine = get_database()
44     session = sessionmaker(bind=engine)
45     return session()
46
47
48 db = get_database()
49 session = get_session()
50 Base = declarative_base()
51 print(db)

```

Код В.6: Elastic IP-пуу холбогдох - connection.py

В.3.2 Өгөгдлийн сантай харьцах

```
1 from re import L
2 from sqlalchemy import Boolean, Column, DateTime, Float, ForeignKey,
  String
3 from sqlalchemy.orm import relationship
4 from assets.classTypes import Advertisement, Category
5 from connection import Base, db, session
6
7
8 class PCategory(Base):
9     __tablename__ = 'category'
10
11     _id = Column(String, primary_key=True)
12     url = Column(String, nullable=False)
13     name = Column(String, nullable=False)
14     parent_id = Column(String,
15                         ForeignKey('category._id',
16                                   onupdate='CASCADE',
17                                   ondelete='CASCADE'),
18                         nullable=True)
19     category = relationship('PCategory')
20
21     def __init__(self, category: Category):
22         self._id = category.id
23         self.url = category.url
24         self.name = category.name
25         if category.parentCategory != None:
26             self.parent_id = category.parentCategory.id
27         else:
28             self.parent_id = None
29
30
31 class PAdvertisement(Base):
32     __tablename__ = 'advertisement'
33
34     _id = Column(String, primary_key=True)
35     category_id = Column(String,
36                          ForeignKey('category._id',
37                                    onupdate='CASCADE',
38                                    ondelete='CASCADE'),
39                          nullable=False)
40     url = Column(String, nullable=False)
41     company = Column(String)
42     title = Column(String)
43     roles = Column(String)
44     requirements = Column(String)
45     additionalInfo = Column(String)
46     city = Column(String)
47     district = Column(String)
48     exactAddress = Column(String)
```



```

49     level = Column(String)
50     types = Column(String)
51     minSalary = Column(Float)
52     maxSalary = Column(Float)
53     isDealable = Column(Boolean)
54     phoneNumber = Column(String)
55     fax = Column(String)
56     publishedDate = Column(DateTime)
57     category = relationship('PCategory')
58
59     def __init__(self, advertisement: Advertisement):
60         self._id = advertisement.id
61         self.category_id = advertisement.category.id
62         self.url = advertisement.url
63         self.company = advertisement.company
64         self.title = advertisement.title
65         self.roles = advertisement.roles
66         self.requirements = advertisement.requirements
67         self.additionalInfo = advertisement.additionalInfo
68         self.city = advertisement.location.city
69         self.district = advertisement.location.district
70         self.exactAddress = advertisement.location.exactAddress
71         self.level = advertisement.level
72         self.types = advertisement.type
73         self.minSalary = advertisement.minSalary
74         self.maxSalary = advertisement.maxSalary
75         self.isDealable = advertisement.isDealable
76         self.phoneNumber = advertisement.contact.phoneNumber
77         self.fax = advertisement.contact.fax
78         self.publishedDate = advertisement.adAddedDate
79
80
81     def createDB():
82         Base.metadata.create_all(db)
83
84
85     def insertToCategory(category: Category):
86         session.add(PCategory(category))
87
88
89     def insertToAdvertisement(advertisement: Advertisement):
90         session.add(PAdvertisement(advertisement))
91
92
93     def upsertCategory(category: Category):
94         if category.parentCategory != None:
95             upsertCategory(category.parentCategory)
96         row = session.query(PCategory).filter(PCategory._id == category.id)
97         if row.first() == None:
98             if category.parentCategory != None:
99                 insertToCategory(category, category.parentCategory.id)
100         else:

```

```

101         insertToCategory(category, None)
102     else:
103         dict = PCategory(category).__dict__
104         del dict['_sa_instance_state']
105         row.update(dict, synchronize_session=False)
106     session.commit()
107     print(category.id, 'CATEGORY UPSERT DONE')
108
109
110 def upsertAdvertisement(advertisement: Advertisement):
111     row = session.query(PAdvertisement).filter(
112         PAdvertisement._id == advertisement.id)
113     if row.first() == None:
114         insertToAdvertisement(advertisement)
115     else:
116         dict = PAdvertisement(advertisement).__dict__
117         del dict['_sa_instance_state']
118         row.update(dict, synchronize_session=False)
119     session.commit()
120     print(advertisement.id, 'ADVERTISEMENT UPSERT DONE')
121
122
123 createDB()

```

Код В.7: Өгөгдлийн сангийн зохиомж болон upsert функц - upsert.py

В.4 Чатбот хөгжүүлэлт

В.4.1 Үндсэн чатбот удирдлага

```

1  const { ActivityHandler, MessageFactory, ActivityTypes } = require('
    botbuilder');
2  const { QuestionUnderstand } = require('./src/assets/question-
    understand/index');
3  const { ApiHelper } = require('./src/assets/bot-api/index')
4  const { CardBuilder } = require('./src/card/index')
5  const welcome = require('./src/card/welcome')
6
7
8  class EchoBot extends ActivityHandler {
9      constructor() {
10         super();
11         const noResponse =
12             '
13         this.onMessage(async (context, next) => {
14             const question = new QuestionUnderstand(context.activity.
                text);
15             const api = new ApiHelper(question.findKeyWord(), question.
                getQueryNumber())
16             console.log('keyword: ' + api.keyword + ' quest: ' +
                question.getQueryNumber())
17             var responseBody = await api.responseBack()

```

```

18     console.log('response:' + responseBody)
19     if (typeof (responseBody) !== 'undefined') {
20         if ((responseBody.length !== 0)) {
21             const view = new CardBuilder(responseBody);
22             switch (question.getQueryNumber()) {
23                 case 1:
24                     var content = view.createAdvertisementCard
25                         ()
26                     for (let index = 0; index < content.length;
27                         index++) {
28                         await context.sendActivity({
29                             attachments: [content[index]]
30                         })
31                     }
32                     break
33                 case 2:
34                     await context.sendActivity({
35                         attachments: [view.createListCard(api.
36                             keyword)]
37                     })
38                     break
39                 case 3:
40                     var content = view.createAdvertisementCard
41                         ()
42                     for (let index = 0; index < content.length;
43                         index++) {
44                         await context.sendActivity({
45                             attachments: [content[index]]
46                         })
47                     }
48                     break
49                 case 4:
50                     await context.sendActivity({
51                         attachments: [view.createSalaryListCard
52                             (api.keyword[0])]
53                     })
54                     break
55                 case 404:
56                     await context.sendActivity(
57                         MessageFactory.text(noResponse,
58                             noResponse)
59                     )
60                     await context.sendActivity({
61                         attachments: [welcome.questionsCard()]
62                     });
63                     break
64                 default:
65                     await context.sendActivity(
66                         MessageFactory.text(noResponse,
67                             noResponse)
68                     )
69                     await context.sendActivity({

```

```

62         attachments: [welcome.questionsCard()]
63     });
64     break
65 }
66 }
67 else {
68     switch (question.getQueryNumber()) {
69         case 1:
70             const tempApi = new ApiHelper(api.keyword
71                 [0], 2)
72             var secondAnswer = await tempApi.
73                 responseBack()
74             const view = new CardBuilder(secondAnswer);
75             var responseText = '      , ' + api.keyword
76                 [0] + '      ' + api.keyword[1] + '
77                 .'
78             if (secondAnswer === undefined)
79                 break
80             await context.sendActivity(
81                 MessageFactory.text(responseText,
82                     responseText)
83             )
84             await context.sendActivity({
85                 attachments: [view.createListCard(api.
86                     keyword[0])]
87             })
88             break
89         default:
90             await context.sendActivity(
91                 MessageFactory.text('
92                     .')
93             )
94             await context.sendActivity({
95                 attachments: [welcome.questionsCard()]
96             })
97             break
98     }
99 }
100 }
101 else {
102     console.log('here')
103     await context.sendActivity(MessageFactory.text(
104         noResponse, noResponse));
105     await context.sendActivity({
106         attachments: [welcome.questionsCard()]
107     });
108 }
109 await next();
110 });
111
112 this.onMembersAdded(async (context, next) => {
113     const membersAdded = context.activity.membersAdded;

```

```

106         const welcomeText =
107             '                !                .
108             ';
109         for (let cnt = 0; cnt < membersAdded.length; ++cnt) {
110             if (membersAdded[cnt].id !== context.activity.recipient
111                 .id) {
112                 await context.sendActivity(MessageFactory.text(
113                     welcomeText, welcomeText));
114                 await context.sendActivity({
115                     attachments: [welcome.questionsCard()]
116                 });
117             }
118         }
119         await next();
120     });
121 }
122 module.exports.EchoBot = EchoBot;

```

Код В.8: чатбот класс - bot.js

B.4.2 API helper

```

1  const fetch = require('node-fetch');
2  const util = require('util');
3
4  class ApiHelper {
5      constructor(word, number) {
6          this.keyword = word
7          this.questionNumber = number
8      }
9
10     async responseBack() {
11         var query = this.getQuery()
12         console.log('query:', query)
13         if (query === 404) {
14             return undefined
15         }
16         var query = this.getQuery()
17         const host = 'http://3.228.127.116'
18         const port = ':4000'
19         const path = '/api/v1'
20         const url = host + port + path + query
21         try {
22             var res = await fetch(url)
23         } catch (error) {
24             console.error(error)
25         }
26         if (!res.ok) {
27             throw res
28         }

```

```

29     var responseBody = await res.json()
30     return responseBody
31 }
32
33 getQuery() {
34     switch (this.questionNumber) {
35         case 1:
36             return util.format('/ad/company=%s&title=%s', this.keyword[0].
                trim().replace(' ', '%20'), this.keyword[1].trim().replace('
                ', '%20'))
37             break
38         case 2:
39             return util.format('/ad/comp=%s', this.keyword.trim().replace('
                ', '%20'))
40             break
41         case 3:
42             return util.format('/ad/types=%s&title=%s', this.keyword[0].
                trim().replace(' ', '%20'), this.keyword[1].trim().replace('
                ', '%20'))
43             break
44         case 4:
45             return util.format('/ad/salary=%s&title=%s', this.keyword[0].
                trim().replace(' ', '%20'), this.keyword[1].trim().replace('
                ', '%20'))
46         case 404:
47             return 404
48             break
49     }
50 }
51 }
52 module.exports.ApiHelper = ApiHelper;

```

Код В.9: Өгөгдлийн сангаас өгөгдөл авах - apiHelper.js

B.4.3 Question Understand

```

1 class QuestionUnderstand {
2     constructor(contextText) {
3         this.contextText = contextText;
4         this.queryNumber = 0;
5     }
6
7     getTypeOfQuestion(text) {
8         if (this.contextText.search('') != -1) return 1
9         else if (this.contextText.search('') != -1) return 2
10        else if (this.contextText.search('') != -1) return 3
11        else if (this.contextText.search('') != -1) return 4
12        else return 404
13    }
14
15    findKeyWord() {
16        switch (this.getTypeOfQuestion()) {

```

```

17     case 1:
18         this.queryNumber = 1
19         var keyword = this.contextText.substring(0, this.contextText.
20             search('      ')).trim().split(' -')
21         return keyword
22         break
23     case 2:
24         this.queryNumber = 2
25         var keyword = this.contextText.substring(0, this.contextText.
26             search(' -')).trim()
27         return keyword
28         break
29     case 3:
30         this.queryNumber = 3
31         var keyword = this.contextText.substring(0, this.contextText.
32             search('      ')).trim().split(' ')
33         return keyword
34         break
35     case 4:
36         this.queryNumber = 4
37         var keyword = this.contextText.substring(0, this.contextText.
38             search('      ')).trim().split(' ')
39         return keyword
40         break
41     default:
42         this.queryNumber = 404
43         return null
44         break
45 }
46 }
47
48 getQueryNumber() {
49     return this.queryNumber;
50 }
51
52 }
53
54 module.exports.QuestionUnderstand = QuestionUnderstand;

```

Код В.10: Хэрэглэгчийн асуултыг ойлгох query үүсгэх - questionUdnerstand.js

B.4.4 Card Builder

```

1  const { CardFactory } = require('botbuilder');
2
3  class CardBuilder {
4      constructor(resultObject) {
5          this.body = resultObject;
6      }
7
8      createAdvertisementCard() {
9          var ret = []
10         const d = new Date()

```

```

11 d.toLocaleDateString
12 for (let index = 0; index < this.body.length; index++) {
13     var cardData = {
14         "type": "AdaptiveCard",
15         "$schema": "http://adaptivecards.io/schemas/adaptive-card.json"
16         ,
17         "version": "1.2",
18         "body": [],
19         "actions": []
20     }
21     const element = this.body[index]
22     cardData["body"].push({
23         "type": "TextBlock",
24         "size": "Medium",
25         "weight": "Bold",
26         "text": element['title'],
27         "wrap": true,
28         "style": "heading"
29     }, {
30         "type": "ColumnSet",
31         "columns": [{
32             "type": "Column",
33             "items": [
34                 {
35                     "type": "Image",
36                     "style": "Person",
37                     "url": "https://cdn-icons-png.flaticon.com/512/622/622848.png",
38                     "altText": element['company'],
39                     "size": "Small"
40                 }
41             ],
42             "width": "auto"
43         }, , {
44             "type": "Column",
45             "items": [{
46                 "type": "TextBlock",
47                 "weight": "Bolder",
48                 "text": element['company'],
49                 "wrap": true
50             }, {
51                 "type": "TextBlock",
52                 "spacing": "None",
53                 "text": " " + new Date(element['publishedDate']).toLocaleDateString('zh-Hans-CN'),
54                 "isSubtle": true,
55                 "wrap": true
56             }],
57             "width": "stretch"
58         }],
59         "type": "TextBlock",

```



```

60         "text": element['roles'],
61         "maxLines": 3,
62         "wrap": true
63     }, {
64         "type": "FactSet",
65         "facts": [{
66             "title": "    :",
67             "value": (element['city']) ? element['city'] : ' ' + ' ' + (
                element['district']) ? element['district'] : ' '
68         }, {
69             "title": "    :",
70             "value": (element['maxSalary'] || element['minSalary']) ?
                element['minSalary'] + ' - ' + element['maxSalary'] : '
                '
71         }, {
72             "title": "    :",
73             "value": element['types']
74         }, {
75             "title": "    :",
76             "value": element['phoneNumber']
77         }]
78     }
79 )
80 cardData['actions'].push({
81     "type": "Action.OpenUrl",
82     "title": "    ",
83     "url": element['url']
84 })
85 ret.push(CardFactory.adaptiveCard(cardData))
86 }
87 return ret
88 };
89 createListCard(title) {
90     var cardData = {
91         "type": "AdaptiveCard",
92         "$schema": "http://adaptivecards.io/schemas/adaptive-card.json",
93         "version": "1.2",
94         "type": "AdaptiveCard",
95         "body": [{
96             "type": "TextBlock",
97             "size": "Medium",
98             "weight": "Bold",
99             "text": title + " -    :",
100             "wrap": true,
101             "style": "heading"
102         }, {
103             "type": "TextBlock",
104             "text": "    : " + this.body.length + "    ",
105             "wrap": true
106         }],
107         "actions": []
108     }

```

```

109     for (let index = 0; index < this.body.length; index++) {
110         const element = this.body[index];
111         cardData['body'].push({
112             "type": "ColumnSet",
113             "spacing": "Medium",
114             "separator": true,
115             "columns": [
116                 {
117                     "type": "Column",
118                     "items": [
119                         {
120                             "type": "Image",
121                             "style": "Person",
122                             "url": "https://cdn-icons-png.flaticon.com/512/2103/2103862.png",
123                             "size": "Small"
124                         }
125                     ],
126                     "width": "auto"
127                 },
128                 {
129                     "type": "Column",
130                     "items": [
131                         {
132                             "type": "TextBlock",
133                             "spacing": "None",
134                             "text": element['company'],
135                             "wrap": true
136                         },
137                         {
138                             "type": "TextBlock",
139                             "weight": "Bold",
140                             "spacing": "None",
141                             "text": element['title'],
142                             "wrap": true
143                         },
144                         {
145                             "type": "TextBlock",
146                             "spacing": "None",
147                             "text": (element['maxSalary'] || element['minSalary'])
148                                 ? element['minSalary'] + ' - ' + element['maxSalary']
149                                 : ' ',
150                             "isSubtle": true,
151                             "wrap": true
152                         }
153                     ],
154                     "width": "stretch"
155                 }
156             ]
157         })
158     }
159     return CardFactory.adaptiveCard(cardData)

```

```

158 }
159 createSalaryListCard(title) {
160   var cardData = {
161     "type": "AdaptiveCard",
162     "$schema": "http://adaptivecards.io/schemas/adaptive-card.json",
163     "version": "1.2",
164     "type": "AdaptiveCard",
165     "body": [{
166       "type": "TextBlock",
167       "size": "Medium",
168       "weight": "Bold",
169       "text": title + " - ",
170       "wrap": true,
171       "style": "heading"
172     }, {
173       "type": "TextBlock",
174       "text": " : " + this.body.length + " ",
175       "wrap": true
176     }],
177     "actions": []
178   }
179   for (let index = 0; index < this.body.length; index++) {
180     const element = this.body[index];
181     cardData['body'].push({
182       "type": "ColumnSet",
183       "spacing": "Medium",
184       "separator": true,
185       "columns": [
186         {
187           "type": "Column",
188           "items": [
189             {
190               "type": "Image",
191               "style": "Person",
192               "url": "https://cdn-icons-png.flaticon.com/512/2103/2103862.png",
193               "size": "Small"
194             }
195           ],
196           "width": "auto"
197         },
198         {
199           "type": "Column",
200           "items": [
201             {
202               "type": "TextBlock",
203               "spacing": "None",
204               "text": element['company'],
205               "wrap": true
206             },
207             {
208               "type": "TextBlock",

```

```

209         "weight": "Bold",
210         "spacing": "None",
211         "text": element['title'],
212         "wrap": true
213     },
214     {
215         "type": "TextBlock",
216         "spacing": "None",
217         "text": (element['maxSalary'] || element['minSalary'])
218             ? element['minSalary'] + ' - ' + element['maxSalary']
219             : ' ',
220         "isSubtle": true,
221         "wrap": true
222     }
223 ],
224 "width": "stretch"
225 ]
226 })
227 }
228 return CardFactory.adaptiveCard(cardData)
229 }
230
231 module.exports.CardBuilder = CardBuilder;

```

Код В.11: Хэрэглэгчид харагдах байдлыг угсрах - cardBuilder.js