

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

Лувсандагвын Давааням

Өрсөлдөөнт түргэн бичилтийн вэб апп
(Competitive fast typing web app)

Програм Хангамж (D 061302)
Бакалаврын судалгааны ажил

Улаанбаатар

2023 он

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

Өрсөлдөөнт түргэн бичилтийн вэб апп
(Competitive fast typing web app)

Програм Хангамж (D 061302)
Бакалаврын судалгааны ажил

Удирдагч:

Мастер. Р.Жавхлан

Гүйцэтгэсэн:

Л. Давааням (20B1NUM0182)

Улаанбаатар

2023 он

Зохиогчийн баталгаа

Миний бие Лувсандагвын Давааням "Өрсөлдөөнт түргэн бичилтийн вэб апп" 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

ЗУРГИЙН ЖАГСААЛТ	iv
ХҮСНЭГТИЙН ЖАГСААЛТ	v
КОДЫН ЖАГСААЛТ	vi
УДИРТГАЛ	1
Зорилго	1
Зорилт	1
Сэдэв сонгох үндэслэл	2
Ач холбогдол	2
БҮЛГҮҮД	3
1. СЭДВИЙН ЕРӨНХИЙ СУДАЛГАА	3
1.1 Үндсэн ойлголтууд	3
1.2 Ижил төсөөтэй систем	4
1.3 Ашиглах технологи	6
2. СИСТЕМИЙН ШААРДЛАГА	11
2.1 Шаардлагын шинжилгээ	11
2.2 UX/UI шаардлага	12
3. СИСТЕМИЙН АРХИТЕКТУР, ЗОХИОМЖ	14
3.1 Системийн архитектур	14
3.2 Дарааллийн диаграм	17
3.3 Ажлын явцын диаграм	19
3.4 UX/UI дизайн	21
3.5 Өгөгдлийн сан	24
4. ХЭРЭГЖҮҮЛЭЛТ	31
4.1 Хөгжүүлэлтийн орчин	31
4.2 Код хөгжүүлэлт	33

5. ҮР ДҮН	44
6. ДҮГНЭЛТ	47
НОМ ЗҮЙ	48
ХАВСРАЛТ	49
A. ӨГӨГДӨЛ БОЛОВСРУУЛАХ ФУНКЦ	49
B. ӨГӨГДЛИЙГ АШИГЛАХ КОМПОНЕНТ	50
C. MUI КОМПОНЕНТ АШИГЛАХ	51
D. AUTH SESSION ЗОХИЦУУЛАХ	52
E. AUTH SESSION АШИГЛАХ	53
F. SOCKET ДЭЭР CORS ЗӨВШӨӨРӨХ	54
G. SOCKET SERVER	55

Зургийн жагсаалт

Зураг	Хуудас
1.1 typingtest.com/pro/ сайтын харагдах байдал	4
1.2 play.typeracer.com сайтын харагдах байдал	6
1.3 Figma ашиглаж хийсэн компонент	7
1.4 Firebase Authentication	9
1.5 Mui Components	10
3.1 Three-tier Архитектурын үлгэр загварын дүрслэл	15
3.2 Хэрэгжүүлэх гэж буй системийн архитектур	16
3.3 Дарааллийн диаграм	17
3.4 Ажлын явцын диаграм	19
3.5 Нүүр хуудас	22
3.6 Уралдах хэсэг	22
3.7 Тэргүүлэгчдийн хэсэг	23
3.8 Профайл хэсэг	23
3.9 Системийн баазын бүтэц	24
3.10 Хэрэглэгчийн үзүүлэлтийн жишээ дата	25
3.11 Хэрэглэгчийн үзүүлэлтийн жишээ дата	25
3.12 Уралдааны жишээ дата	26
3.13 Хэрэглэгчийн үзүүлэлтийн жишээ дата	26
4.1 Төслийн файлын бүтэц	32
5.1 Нүүр хуудас	44
5.2 Хэрэглэгч шивэх тест өгөх хуудас	44
5.3 Нийт тоглогчдыг тэргүүлэгчдийн хуудас	45
5.4 Хэрэглэгчийн статистикийн хуудас	45
5.5 Нэвтрэх хуудас	46

Хүснэгтийн жагсаалт

2.1	Функциональ шаардлага	11
2.2	Функциональ бус шаардлага	12
2.3	User Experience дизайны шаардлага	13
2.4	User Interface дизайны шаардлага	13
3.1	User хүснэгт	27
3.2	User Statistics Table	28
3.3	Race Table	29
3.4	Race Type Table	29
3.5	Race Text Table	30

Кодын жагсаалт

4.1	Хэрэглэгчийн мэдээллийг бодит хугацаанд харуулах компонент	33
4.2	Хэрэглэгч өрөөнд нэвтрэх client талын код	34
4.3	Текст fetch хийж буй async функц	35
4.4	Firebase config	37
4.5	Тоглоомын цагийг удирдах Firebase өгөгдлийг шинэчлэхэд зориулсан useEffect hook	38
4.6	Хэрэглэгчийн Authentication болон Session-г зохицуулах	40
4.7	Socket ашиглан хэрэглэгчдийг зохицуулах	41

УДИРТГАЛ

Орчин үеийн ертөнц дижитал харилцаа холбоог хөгжүүлж, шивэх хурд, нарийвчлал нь янз бүрийн мэргэжлээр ажилладаг хүмүүст зайлшгүй шаардлагатай ур чадвар болж байна. "Duck Racer"-нь шивэх чадвараа сайжруулах, сурах үйл явцыг тоглоом болгох зорилготой вебэд суурилсан программ юм. Энэхүү систем нь өрсөлдөөнт уралдааны механизмыг нэгтгэсэн бөгөөд оролцогчид өгөгдсөн догол мөрүүдийг бичиж бодит цагт тоглогчдын эсрэг уралдана. Хэрэглэгчдэд сонирхолтой интерфэйс, бодит цагийн гүйцэтгэлийн хэмжигдэхүүн, текстийн хэсгээс бүрдсэн олон төрлийн мэдээллийг цогцоор нь хослуулж, шивэх урлагийг эзэмших өвөрмөц аргыг хэрэглэгчдэд санал болгож байна.

Зорилго

Шивэх дасгалыг үр дүнтэй, тааламжтай болгох системийг тоглоомын аргаар сургалтын үйл явцад нэгтгэж шивэх дасгалыг сонирхолтой, хэрэглэгчдийг өрсөлдөх чадвартай болгохыг зорьж байна.

Зорилт

Уг веб аппыг хөгжүүлэхдээ дараах үе шатын дагуу ажиллана.

1. Техникийн болон хэрэглэгчийн шаардлагыг тодорхойлох;
2. Хэрэглэгчдийн анхаарлыг татахуйц хэрэглэгчдэд ээлтэй интерфэйсийг дизайн гаргах;
3. Ашиглах технологийг онол болон практик дээр суурилж судлах;
4. Системийн архитектурын бүтэц, дизайныг зохион байгуулж бэлдэх;
5. Гаргасан баримт бичгийн дагуу системийн хөгжүүлэлтээ хийх;
6. Бэлэн болсон системд домейн нэр авж, хост хийн байршуулах.

Сэдэв сонгох үндэслэл

Энэхүү дипломын ажил нь шивэх чадварыг сайжруулах өвөрмөц боловсролын систем бөгөөд. Энэ сэдвийг сонгосон нь хэд хэдэн чухал шалтгаанаас үүдэлтэй:

1. Монголын хүн амын гуравны нэгээс илүү хувь нь 24-өөс доош насныхан байдаг бөгөөд, тоглоомын платформоор залуучуудыг татан оролцуулах нь тэднийг дижитал эринд илүү сайн бэлтгэж чадна.
2. COVID-19 тахал гэх мэт нөхцөл байдлаас үүдэн хурдассан онлайн боловсрол руу дэлхий даяар шилжиж байгаа нь шивэх чадвар сайтай байх шаардлагатайг улам тодотгосон.¹
3. Орчин үеийн ажлын байрууд салбараас үл хамааран харилцаа холбоо, хамтын ажиллагаа, бичиг баримт бүрдүүлэхэд дижитал хэрэгслүүд ихээхэн ашигладаг. шивэх ур чадвар нь бүтээмжийг дээшлүүлж, алдааг багасгаж, ажлын урсгалыг илүү хялбар болгоход хувь нэмэр оруулна.²
4. Вэб дээр суурилсан платформ нь дэлхий даяарх хэрэглэгчдийг холбох боломжтой. Энэхүү дэлхийн холболт нь соёлын солилцоог дэмжиж, эрүүл өрсөлдөөнийг дэмжиж, суралцах нийгэмлэгийг бий болгож чадна.³

Ач холбогдол

Уг системийг бүтээснээр тоглоомын сорилтуудаар дамжуулан хэрэглэгчид шивэх хурд, нарийвчлалыг цаг хугацааны явцад сайжруулахад түлхэц болох. Сургууль, боловсролын байгууллагууд хичээлд нэмэлт хэрэгсэл болгон ашиглах. Оюутнууд дадлага хийж, ахиц дэвшилээ хянаж, ангийнхантайгаа хөгжилтэй, интерактив байдлаар өрсөлдөж, сургалтын үйл явцыг илүү сонирхолтой байх боломжуудыг бүрдүүлэх юм.

¹Боловсролын талаарх ЮНЕСКО-гийн тайлан: <https://www.unesco.org/en/education>

²LinkedIn ажлын зах зээлийн албан ёсны тайлан: <https://economicgraph.linkedin.com/resources>

³Pew судалгааны төв: <https://www.pewresearch.org/>

1. СЭДВИЙН ЕРӨНХИЙ СУДАЛГАА

Сэдвийн хүрээнд өмнө нь ажиллаж үзээгүй олон шинэ технологиудыг судалж хэд хэдэн шинэ технологи ашиглаж, тэдгээрийг үнэлж, харьцуулав. Энэ бүлэгт би судалгаанаасаа сонгосон технологиудыг танилцуулж, вэб платформыг бий болгох үндсэн ойлголтуудыг танилцуулж байна.

1.1 Үндсэн ойлголтууд

Энэхүү судалгааг амжилттай дуусгахын тулд зарим үндсэн ойлголтуудыг ойлгох нь зайлшгүй чухал юм.

1.1.1 *Web Sockets*

WebSocket нь хэрэглэгчийн хөтөч болон серверийн хооронд хоёр талын интерактив харилцааны session нээх боломжтой дэвшилтэт технологи юм. Энэхүү WebSocket - ийн тусламжтайгаар сервер рүү мэдээлэл илгээж, серверээс хариулт авах шаардлагагүйгээр үйл явдалд тулгуурласан хариултуудыг хүлээн авах боломжтой.

- Real-time Interaction: Хэрэглэгчдийг бодит цаг хугацаанд бие биетэйгээ уралдуулахыг хүсвэл WebSockets маш чухал. Тэд оролцогч бүрийн мэдээллийг бусад бүх оролцогчдын дэлгэцэн дээр нэн даруй шинэчлэх боломжийг олгоно.
- Dynamic Updates: Хэрэв тоглоомын орчин, дүрэм, зард ямар нэгэн өөрчлөлт орсон бол WebSockets нь бүх идэвхтэй хэрэглэгчдэд шууд мэдэгдэх боломжтой.
- Multiplayer Racing: Олон оролцогчтой уралдааны хувьд тоглоомын төлөвийг удирдаж, хэрэглэгчдэд синхрончлолыг хангах нь WebSockets-ийн тусламжтай илүү удирдах боломжтой болно.

WebSockets нь бодит цагийн интерактив платформын салшгүй технологийн нэг бөгөөд шуурхай шинэчлэлт, харилцан үйлчлэлд шаардагдах хурд, үр ашгийг санал болгодогоороо

1.2. ИЖИЛ ТӨСӨӨТЭЙ СИСТЕМ БҮЛЭГ 1. СЭДВИЙН ЕРӨНХИЙ СУДАЛГАА

давуу талтай.

1.1.2 Web-based Educational Platforms

Вэб дээр суурилсан платформууд нь хэрэглэгчид хүссэн үедээ, хаанаас ч, ихэвчлэн өөрийн хүссэн хэмжээгээр систем руу хандах боломжтойгоороо давуу талтай.

Өрсөлдөх чадвартай эсвэл хамтран ажиллах боломжуудыг агуулж, суралцахыг нийгмийн туршлагыг бий болгож хэрэглэгчид дэлхийн хэмжээнд үе тэнгийнхэнтэйгээ өрсөлдөх эсвэл тэднээс суралцах боломжтой.

1.2 Ижил төсөөтэй систем

1.2.1 TypingTest.com

”TypingTest Pro” нь хэрэглэгчид шивэх хурд, нарийвчлалыг хэмжихэд зориулагдсан онлайн платформ юм. Энэ платформ нь хэрэглэгчдэд бодит цагийн горимд шивэх янз бүрийн текстүүдийг өгч, гүйцэтгэлийн талаар шууд санал хүсэлтийг санал болгодог.



Зураг 1.1: typingtest.com/pro/ сайтын харагдах байдал

1.2. ИЖИЛ ТӨСӨӨТЭЙ СИСТЕМ БҮЛЭГ 1. СЭДВИЙН ЕРӨНХИЙ СУДАЛГАА

Манай бүтээх гэж байгаа вебээс ялгаатай тал нь

- 1-ээс 10 минутын хугацаатай туршилтуудыг санал болгож, хэрэглэгчдэд сорилтын түвшингээ сонгох боломжийг олгодог.
- Шууд guest хэлбэрээр орон тест өгөх боломжгүй.

Frontend: HTML5, CSS3 болон динамик хэрэглэгчийн интерфэйсүүдэд зориулсан React.js-тэй JavaScript. Backend: Express.js хүрээтэй Node.js нь өргөтгөх боломжтой, үр ашигтай сервер талын програмыг хангадаг. Өгөгдлийн сан: Хэрэглэгчийн профайл, тестийн үр дүн, текстийн хэсгүүдийг хадгалах MongoDB. WebSockets: Бодит цагийн өрсөлдөөн, шууд санал хүсэлтийн функцэд зориулагдсан. Third-party Integrations: Дэлхий даяар тэргүүлэгчдийн самбар, олон нийтийн мэдээллийн хэрэгслээр хуваалцах боломжууд.

Онлайнаар шивэх тестийн олон платформууд байдаг ч "TypingTest Pro" нь энгийн хэрэглэгчид болон шивэх чадвараа сайжруулахад нухацтай ханддаг хүмүүст зориулсан иж бүрэн функцээрээ бусдаас ялгардаг.

1.2.2 *play.typeracer.com*

TypeRacer бол олон тоглогчийн онлайн хөтөч дээр суурилсан шивэх тоглоом юм. Тоглогчид богино хэсгийг аль болох хурдан шивэх замаар өрсөлддөг бөгөөд тоглоом нь хурдыг минут тутамд үгээр (WPM) болон нарийвчлалыг хэмждэг. Үүсгэн байгуулагдсан цагаасаа эхлэн энэ нь өрсөлдөөнтэй, хөгжилтэй орчинд шивэх чадвараа сайжруулахыг хүсч буй хүмүүсийн дуртай хэрэгсэл болсон.

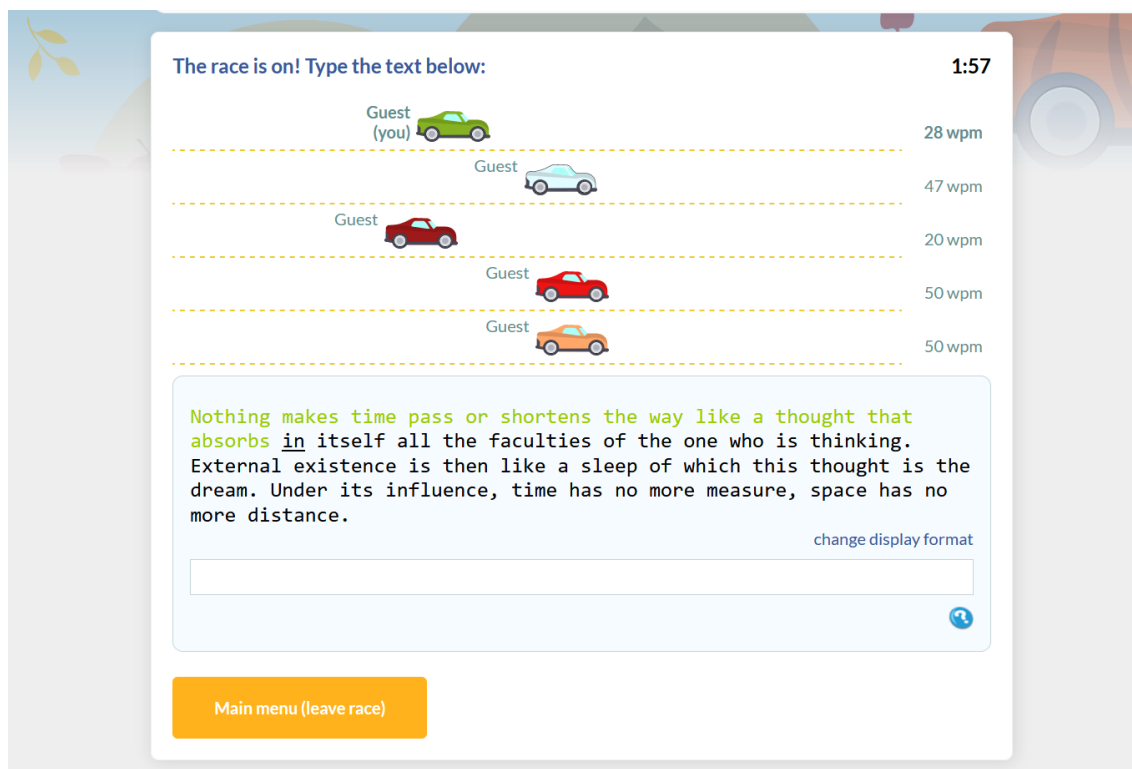
Гол онцлог:

Энэ бол вэб дээрх анхны олон тоглогчийн шивэх тоглоом юм.

2008 оны 3-р сард нээлтээ хийснээс хойш дэлхийн өнцөг булан бүрээс сая сая хүмүүс typeracer.com сайт дээр хэдэн зуун сая уралдаанд оролцож, шивэх хурдаа минут тутамд 50 үгээр сайжруулсан.

TypeRacer нь 50 өөр хэл дээр байдаг.

2010 онд гарсан TypeRacer School Edition нь дэлхийн хамгийн хөгжилтэй боловсролын бүтээгдэхүүн болох зорилготой юм. K-12 сургуулиудад зориулагдсан бөгөөд шивэ-



Зураг 1.2: play.typeracer.com сайтын харагдах байдал

хийг спорт болгон хувиргаж, суралцахыг хөгжилтэй болгодог TypeRacer тоглоомын үзэл баримтлалыг ашигладаг.

1.3 Ашиглах технологи

1.3.1 Figma - интерфэйс дизайн, Prototype хувилбар гаргах багаж

Figma нь хэрэглэгчдэд нэг платформ дээр дизайн хийх, загвар гаргах, санал хүсэлтийг цуглуулах боломжийг олгодог дизайны хэрэгсэл юм. Дизайнерууд, хөгжүүлэгчид, бүтээгдэхүүний менежерүүд болон оролцогч талууд бодит цаг хугацаанд хамтран ажиллах боломжтой бөгөөд энэ нь дижитал бүтээгдэхүүн дээр ажилладаг багуудад хүчирхэг хэрэгсэл болгодог.

Figma ашиглахын давуу талууд

- Вэб дээр суурилсан бөгөөд энэ нь том хэмжээний програм хангамжийн багц татаж авах шаардлагагүй. Windows, MacOS, Linux зэрэг өөр өөр үйлдлийн системүүд дээр

саадгүй ажилладаг.

- Хийсэн өөрчлөлтүүдийг автоматаар хадгалж, шаардлагатай бол хуучин өөрчлөлт рүү буцах боломжийг олгодог.
- Figma нь оюутнуудад нэмэлт зардал гаргахгүйгээр иж бүрэн төсөл хэрэгжүүлэх боломжийг үнэ төлбөргүй санал болгодог.



Зураг 1.3: Figma ашиглаж хийсэн компонент

Зурсан интерфэйсүүдээ хооронд нь холбож хийсвэрээр аппаа ажиллуулан хэрэглэгчийн туршилт хийх хэсгийг Prototype гэдэг бөгөөд заавал кодын хэрэгжүүлэлт хийж цаг хугацаа болон мөнгөн зардал гаргалгүйгээр хийж буй аппаа хэрэглэгчээр туршуулах, үр дүнгээ гарган авч түүнийг сайжруулах нөхцөлийг уг веб аппликейшн маань гаргаж өгсөн нь UX/UI дизайнеруудын ашиглах болсон хамгийн том шалтгаануудын нэг юм.

1.3.2 Next.js - React дээр суурилсан фрэймворк

Сонгосон шалтгаан

Энгийнээр хэлэхэд Next.js нь Javascript програмуудыг хөгжүүлэхэд зориулагдсан React framework юм. Вэб платформыг өндөр гүйцэтгэлтэй, өргөтгөх боломжтой, зөвхөн код дээрээ анхаарал хандуулах боломжийг олгож хурдан ажилладаг.

Next.js давуу талуудаас дурьдвал:

- Вэб програмуудыг бүтээхэд хялбар бүтэцтэй.
- Automatic code splitting JS болон CSS шаардлагагүй бол заавал татаж авдаггүй.

- Zero config буюу нэг ч тохиргоо хийлгүйгээр төслөө эхлүүлэх боломж.
- Server Side Render хийх (SSR).
- Typescript болон Fast Refresh дэмждэг.
- HRM болон хөгжүүлэгчдэд ээлтэй хэрэгслүүдтэй.
- API Routes буюу өөр дээрээ nodejs сервер ашиглаж API endpoint гаргах боломжтой.
Ингэснээр тусдаа сервер ашиглах шаардлага үүсэхгүй.
- SEO буюу хайлтын системийн оновчлолыг SSR ашиглаж тохируулж өгөх.

Хөгжүүлэлтийг хялбарчлаж хөгжүүлэгчдэд ээлтэй орчинг бүрдүүлэх чадвартай тул Next.js-ийг сонгов.

Технологийн талаар

Next.js¹ нь уян хатан React дээр суурилсан фрэймворк бөгөөд хурдан вэб програмуудыг чанартай үүсгэх боломжийг өгдөг билээ.

1.3.3 Firebase - Өгөгдлийн сан

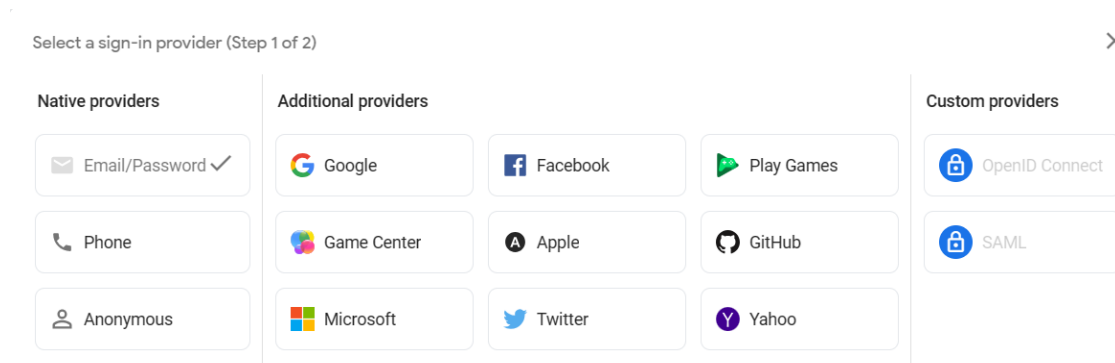
Firebase бол Google-ээс гаргасан гар утас болон вэб програм хөгжүүлэх цогц платформ юм. Энэ нь хөгжүүлэгчдэд програмуудыг хурдан бөгөөд үр дүнтэй бүтээх, сайжруулах, масштаблахад туслах өргөн хүрээний хэрэгсэл, үйлчилгээг санал болгодог. Firebase нь янз бүрийн функц, үйлчилгээг багтаасан бөгөөд үүнийг програм хөгжүүлэхэд түгээмэл сонголт болгодог. Firebase-ийн зарим үндсэн бүрэлдэхүүн хэсэг, боломжуудыг энд харуулав.

Firebase давуу талуудаас дурьдвал:

- Real-time Database энэ функц нь холбогдсон бүх үйлчлүүлэгчид тоглоомын өгөгдлийг шуурхай шинэчлэх боломжийг олгож, олон тоглогчийн туршлагыг тасралтгүй бий болгодог.

¹Next.js official site <https://nextjs.org>

- Authentication бүртгэл, нэвтрэх функцийг хэрэгжүүлэхэд хялбар болгодог
- Serverless Functions эдгээр функцийг тоглоомын session удирдлага, онооны тооцоо зэрэг сервер талын логикийг хэрэгжүүлэхэд ашиглаж болно.
- Cost efficiency боломжийн үнэ бүхий, төлбөргүй олон боломжуудыг санал болгодог.



Зураг 1.4: Firebase Authentication

Firebase-ийг бодит цагийн харилцан үйлчлэл, authentication зэргийг төслийн гол бүрэлдэхүүн хэсэг болгон хэрэглэх болно.

1.3.4 *Socket.io - Сан*

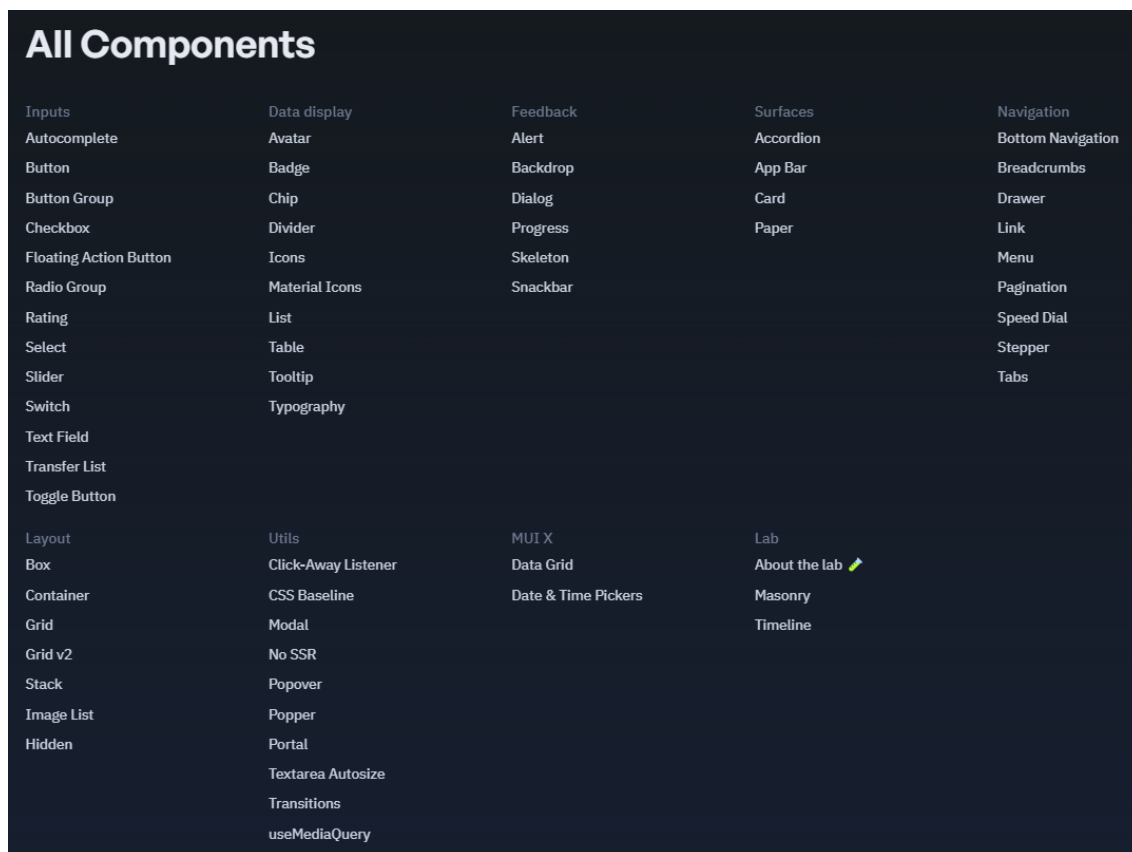
Socket.IO бол бодит цагийн вэб програмуудад зориулсан үйл явдалд суурилсан сан юм. Энэ нь вэб үйлчлүүлэгч болон серверүүдийн хооронд бодит цагийн, хоёр чиглэлтэй харилцаа холбоог идэвхжүүлдэг. Энэ нь үйлчлүүлэгч, сервер гэсэн хоёр бүрэлдэхүүн хэсгээс бүрдэнэ.

1.3.5 *ExpressJS - React дээр компонентуудын сан*

Express нь Node.js-д зориулсан хамгийн бага бөгөөд уян хатан вэб програмын фрэймворк бөгөөд JavaScript кодын сервер талд ажиллах ажиллах орчин юм. Socket.IO-г Express.js-тэй нэгтгэх үед HTTP хүсэлт болон WebSocket холболттой бодит цагийн вэб програм үүсгэх боломжтой болж байна.

1.3.6 MUI - Node.js дээр суурилсан фрэймворк

MUI нь иж бүрэн үнэгүй UI хэрэгслүүдийг санал болгодог. Интерфейс хэрэгжүүлдэг frontend инженерүүдийн хувьд цагийг хэмнэсэн түгээмэл багаж бөгөөд React-ийн Server Side Rendering-ийг дэмждэг.



Зураг 1.5: Mui Components

2. СИСТЕМИЙН ШААРДЛАГА

Уг бүлэг нь системийн хэрэглэгчийн зүгээс тавигдах шаардлагыг тодорхойлж, хэрэглэгч талаас ээлтэй хэрэглэхэд амархан байх тал дээр UX болон UI дизайны шаардлагуудыг гаргасан.

2.1 Шаардлагын шинжилгээ

2.1.1 Хэрэглэгчид

Интернэт сүлжээ ашиглан хурдан шивэх хүсэлтэй бүх төрлийн хэрэглэгчид хэрэглэх боломжтой.

2.1.2 Функционал шаардлагууд

Хүснэгт 2.1: Функциональ шаардлага

ФШ 101	Бодит цагт мэдээллүүдийг харуулах (хурд, нарийвчлал, одоогийн байрлал).
ФШ 102	Алдсан, буруу бичсэн үг эсвэл тэмдэгтүүдийг тодруулах.
ФШ 103	Хэрэглэгчид өгөгдсөн текстнээс хамаарч дуусгахад шаардагдах хугацааг өөрчлөх.
ФШ 104	Хэрэглэгчид unique холбоос ашиглан тодорхой найзуудтайгаа өрсөлдөх боломжтой байх.
ФШ 105	Уралдааны үеэр бүх оролцогчдын бодит цагийн явцыг харуулах.
ФШ 106	шивэх хурд, нарийвчлал дээр үндэслэн тэргүүлэгчдийн самбарыг харуулах.
ФШ 107	Веб нь хэрэглэгч бүртгэх боломжтой байх.

2.1.3 Функционал бус шаардлагууд

Хүснэгт 2.2: Функциональ бус шаардлага

ФБШ 101	Уралдааны үеэр бодит цагийн хариу үйлдэл үзүүлэг байх.
ФБШ 103	Систем нь дор хаяж 2 хэрэглэгчид хоорондоо өрсөлддөг байх.
ФБШ 104	SQL injection, CSRF халдлагууд болон бусад нийтлэг вэб эмзэг байдлаас хамгаалттай байх.
ФБШ 105	Хэрэглэгчийн өгөгдөл болон тоглоомын статистикийг тогтмол нөөцлөдөг байх.
ФБШ 102	Интерфейс, UI нь шинэ хэрэглэгчдэд зориулсан ойлгомжтой байх.
ФБШ 106	Төрөл бүрийн хөтчүүдэд нийцтэй байх.

2.2 UX/UI шаардлага

UI/UX (User Interface/User Experience) шаардлагууд нь програм хангамж, вэбсайт эсвэл аливаа дижитал бүтээгдэхүүн боловсруулахад хэд хэдэн шалтгааны улмаас зайлшгүй шаардлагатай байдаг.

1. Хэрэглэгч төвтэй дизайн -> Хэрэглэгчийн хэрэгцээ, сонголтыг ойлгож, баримтжуулснаар хэрэглэгчийн эерэг туршлагыг бий болгох боломжтой.
2. Эрсдэлийг бууруулана -> Шаардлагууд нь хөгжлийн дараагийн үе шатанд хамрах хүрээ болон өөрчлөлтөөс урьдчилан сэргийлэхэд тусална.
3. Зохион байгуулалт, өнгө, хэв маяг, загвар зэрэг дизайны элементүүдийн дагнасан байдал нь хэрэглэгчдэд системийг удирдах, ойлгоход хялбар болгодог.

Дизайны тодорхой шаардлагыг эхнээс нь хэрэгжүүлэх нь хэрэглэгчийн туршлагад сөргөөр нөлөөлөхөөс өмнө ашиглалтын асуудлыг шийдвэрлэхэд тусалдаг давуу талтай.

2.2.1 User Experience шаардлага

UX шаардлагууд нь хэрэглэгчийн хэрэгцээ, сонголт, хүлээлтийг ойлгох, хангахад зайлшгүй шаардлагатай.

Хүснэгт 2.3: User Experience дизайны шаардлага

ИДШ 101	Зорилтот хэрэглэгчдийн зорилгыг ойлгохын тулд хэрэглэгчийн судалгаа хийх.
ИДШ 102	Цэсийг хөнгөвчлөхийн тулд агуулга, онцлогуудыг логик, уялдаатай байдлаар зохион байгуулах.
ИДШ 103	Янз бүрийн төхөөрөмж болон дэлгэцийн хэмжээтэй сайн ажилладаг интерфэйсийг зохион бүтээх.
ИДШ 104	Брэнд, хэрэглэгчийн хүлээлттэй нийцсэн сэтгэл татам, үзэмжтэй интерфэйсийг зохион бүтээх.

2.2.2 User Interface дизайны шаардлага

UI шаардлагууд нь үр дүнтэй, харагдахуйц хэрэглэгчийн интерфэйсийг бий болгоход зайлшгүй шаардлагатай.

Хүснэгт 2.4: User Interface дизайны шаардлага

ИДШ 105	Хэрэглэгчдийн анхаарлыг чухал элементүүд болон контентод чиглүүлэхийн тулд тодорхой харааны шатлалыг бий болгох.
ИДШ 106	Гол элементүүд дээр "3A3845"hex кодтой өнгийг ашиглах.
ИДШ 107	Текст элементүүдийн үсгийн хэлбэр, үсгийн хэмжээ, мөр хоорондын зай, шатлал зэргийг зааж өгөх.
ИДШ 108	Цэс, навигацийн мөр, site map зэрэг навигацийн бүтцийг тодорхойлох.
ИДШ 109	Contrast буюу өнгөний ялгарлыг бага байлгах.
ИДШ 110	Элемент хооронд white-space буюу сул зайг сайн гаргаж өгөх.

3. СИСТЕМИЙН АРХИТЕКТУР, ЗОХИОМЖ

Уг бүлэгт системийн ерөнхий архитектур, зохиомжийг хамгийн ашигтай, хурдан ажиллах хэрэгтэй тул үүн дээрээ үндэслэн хэрэгжүүлэлт хийж үр дүнг харуулав.

3.1 Системийн архитектур

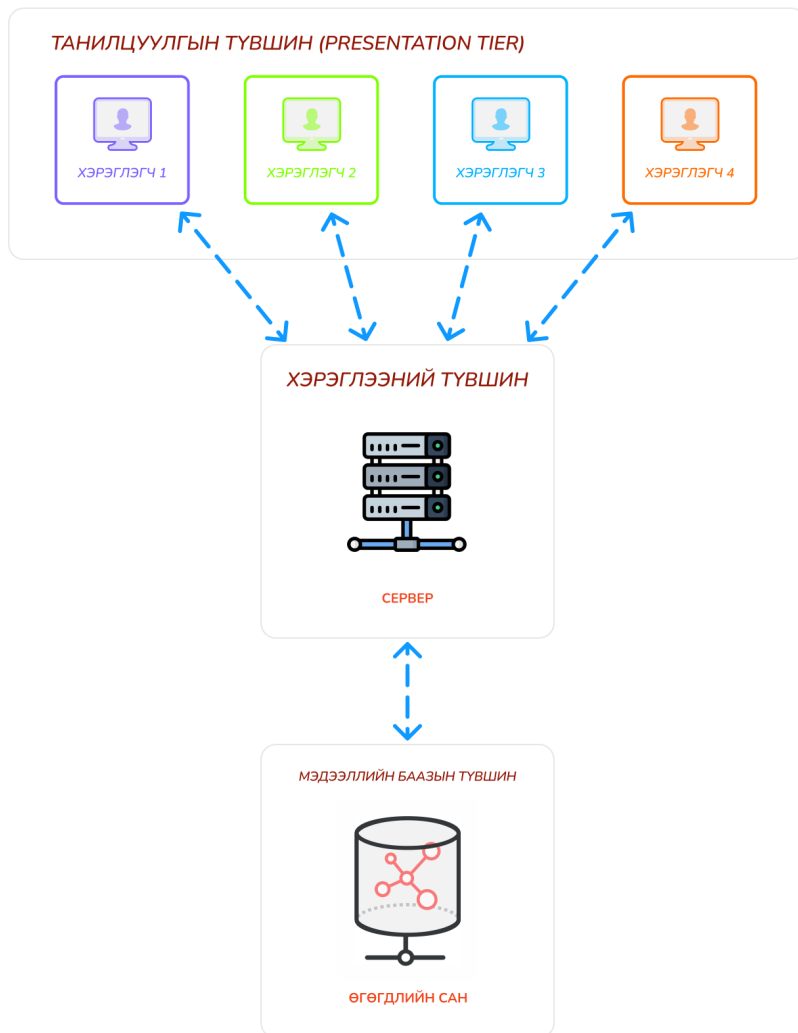
Системийн архитектур нь дараах хүчин зүйлсээс шалтгаална.

1. Хэрэв олон тооны хэрэглэгчдийг хүлээж байгаа бол томрох боломжтой архитектурыг сонгох хэрэгтэй бөгөөд ингэснээр илүү их урсгалыг зохицуулахын тулд илүү олон сервер нэмж болно.
2. Систем үргэлж бэлэн байх өндөр хүртээмжтэй архитектурыг сонгох хэрэгтэй бөгөөд энэ нь зарим бүрэлдэхүүн хэсэг нь бүтэлгүйтсэн ч үргэлжлүүлэн ажиллах боломжтой.
3. Хэрэглэгчийн мэдрэмтгий өгөгдлийг хадгалдаг бол аюулгүй, халдлагыг тэсвэрлэх чадвартай архитектурыг сонгох хэрэгтэй.

Дараах хүчин зүйлсүүдийг хамааран (Three-tier) үлгэр загварыг сонголоо. Вэб системийн нийтлэг архитектур бөгөөд танилцуулгын түвшин, хэрэглээний түвшин, мэдээллийн баазын түвшин гэсэн гурван давхаргаас бүрдэнэ.

3.1. СИСТЕМИЙН АРХИТЕКТУРА

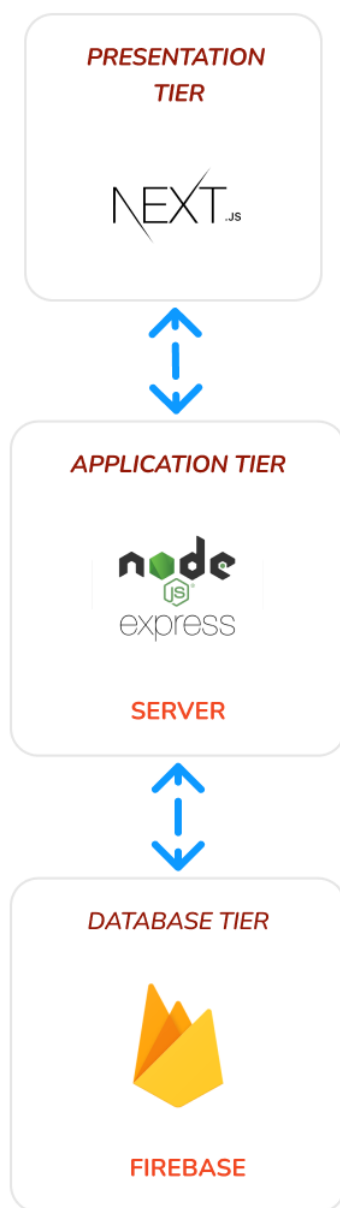
Үзүүлэнгийн түвшин нь хэрэглэгчийн интерфэйсийг зохицуулж, үр дүнг хэрэглэгчдэд харуулдаг. Хэрэглээний давхарга нь системийн логикийг зохицуулж, мэдээллийн сантай харьцдаг. Өгөгдлийн сангийн давхарга нь системийн өгөгдлийг хадгалдаг.



Зураг 3.1: Three-tier Архитектурын үлгэр загварын дүрслэл

3.1. СИСТЕМИЙН АРХИТЕКТУРА ~~ХҮҮРГ~~ 3. СИСТЕМИЙН АРХИТЕКТУР, ЗОХИОМЖ

Front-end хэсэг Next.js дээр хөгжүүлэлт хийгдэх бол. Харин Back-end хэсэг Express.js болон Firebase дээр хөгжүүлэлт хийгдэнэ.

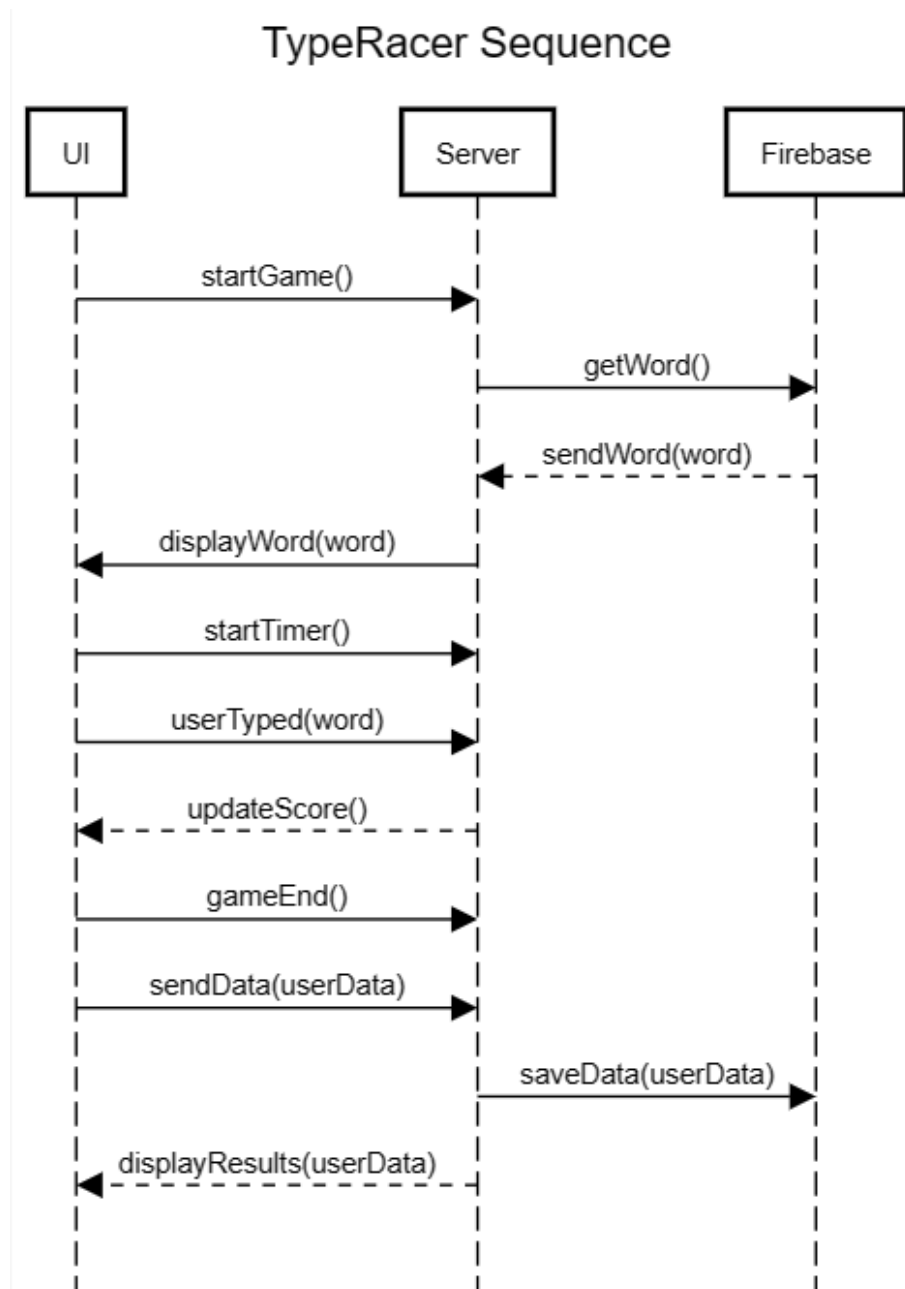


Зураг 3.2: Хэрэгжүүлэх гэж буй системийн архитектур

3.2 Дарааллийн диаграм

Энэхүү дарааллын диаграмм нь тест эхлэх үеийн UI, Server, Firebase хоорондын үйлчлэл, мессежийн урсгалын дарааллийг харуулна.

3.2.1 Дарааллийн диаграм



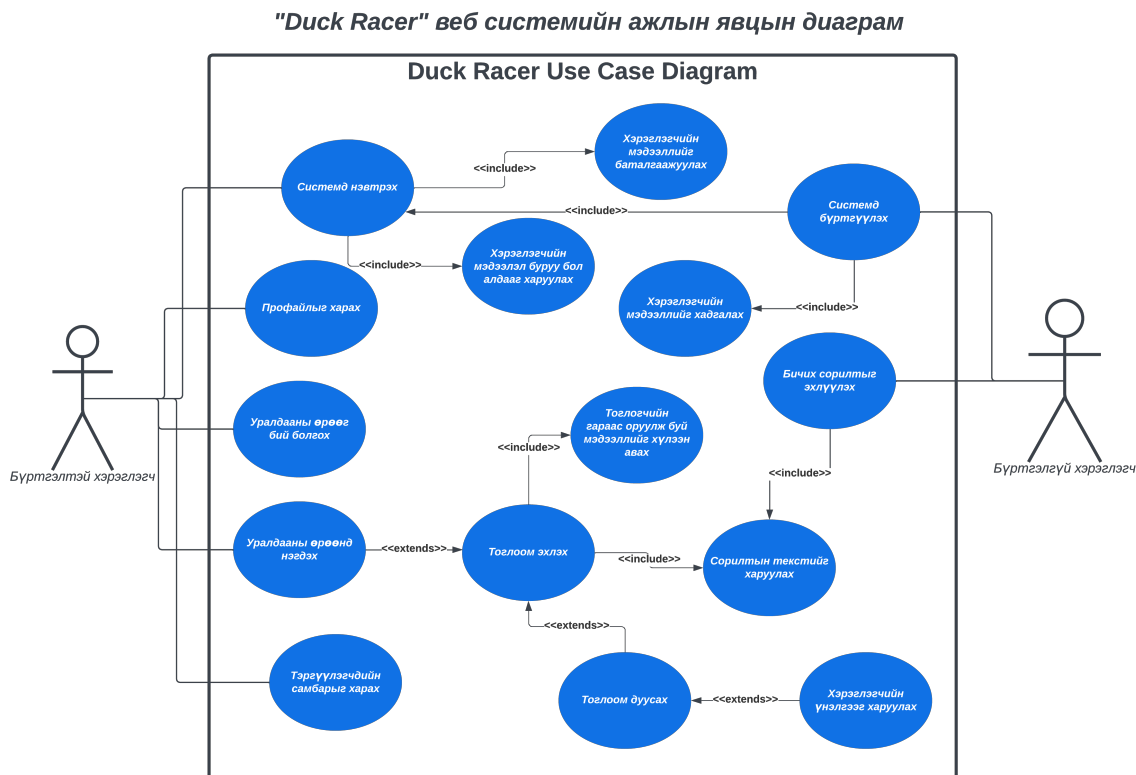
Зураг 3.3: Дарааллийн диаграм

Дарааллийн диаграмын тайлбар

- **startGame()** - Хэрэглэгч текстийг эхлүүлнэ.
- **getWord(word)** - Сервер firebase - с шинээр өгүүлбэр авахын тулд хүсэлт явуулна.
- **sendWord(word)** - Хүсэлт амжилттай тохиолдолд тухайн өгүүлбэрийг буцаан явуулна.
- **displayWord(word)** - Ирсэн өгүүлбэрийг хэрэглэгчийн дэлгэцэд харуулна.
- **startTimer()** - Өгүүлбэр амжилттай ирсэн тохиолдолд цагийг эхлүүлнэ.
- **userTyped(word)** - Хэрэглэгчийн бичсэн үсгийг серверлүү явуулна.
- **updateScore()** - Серверээс хэрэглэгчийн мэдээллийг шинэчлэнэ.
- **gameEnd()** - Цаг дууссан тохиолдолд текстийг дуусгана.
- **sendData(userData)** - Хэрэглэгчийн мэдээллийг (wpm, time, accuracy) серверлүү явуулна.
- **saveData(userData)** - Хэрэглэгчийн мэдээллийг firebase дээр хадгална.
- **displayResults(userData)** - Хэрэглэгчийн мэдээллийг модаль загвараар харуулна.

3.3 Ажлын явцын диаграм

3.3.1 Ажлын явцын диаграм



Зураг 3.4: Ажлын явцын диаграм

Ажлын явцын диаграммын тайлбар

- **Нэвтрэх** - Бүртгүүлсэн хэрэглэгч системд хэрэглэгчийн нэр, нууц үгээ оруулан нэвтрэх.
- **Бүртгүүлэх** - Хэрэглэгч бүртгэлгүй хэдий ч манай платформыг бүрэн ашиглах боломжтой. Хэрэв өөрөө веб холбоос оруулахыг хүсвэл хувийн мэдээллээ бөглөж бүртгүүлэх.
- **Хэрэглэгчийн мэдээллийг баталгаажуулах** - Бүртгэгдсэн хэрэглэгч шивэх сорилтынхоо үр дүнг харж, ахиц дэвшлийг нь харж, сайжруулах шаардлагатай хэсгийг тодорхойлох.

3.3. АЖЛЫН ЯВЦЫН ДИАГРАММ 3. СИСТЕМИЙН АРХИТЕКТУР, ЗОХИОМЖ

- **Хэрэглэгчийн мэдээлэл буруу бол алдааг харуулах** - Бүртгэгдсэн хэрэглэгч шивэх сорилтынхоо үр дүнг харж, ахиц дэвшлийг нь харж, сайжруулах шаардлагатай хэсгийг тодорхойлох.
- **Профайлыг харах** - Бүртгэгдсэн хэрэглэгч өөрийн профайлыг харж шивэх хурд, нарийвчлал болон бусад статистикийг харах.
- **Шивэх сорилтыг эхлүүлэх** - Бүх хэрэглэгч шивэх чадвараа сайжруулахын тулд шивэх сорилтыг эхлүүлэх.
- **Тэргүүлэгчдийн самбарыг харах** - Бүртгүүлсэн хэрэглэгч тэргүүлэгчдийн самбарыг харж бусад хэрэглэгчдийнхээ эсрэг хэрхэн эрэмбэлэгдсэнийг харах.
- **Уралдааны өрөөг бий болгох** - Бүртгүүлсэн хэрэглэгч бусад хэрэглэгчдийг шивэх уралдаанд уралдуулахын тулд уралдааны өрөө үүсгэх.
- **Уралдааны өрөөнд нэгдэх** - Бүртгүүлсэн хэрэглэгч өөр хэрэглэгчийн үүсгэсэн уралдааны өрөөнд нэгдэх.
- **Тоглоом эхлэх** - Өрөөнд нэгдсэний дараагаар уралдааныг эхлүүлэх.
- **Тоглогчийн гараас оруулж буй мэдээллийг хүлээн авах** - Хэрэглэгчийн мэдээллийг серверлүү илгээх.
- **Сорилтын текстийг харуулах** - Баазаас текстийг татан авч харуулах.
- **Тоглоом дуусах** - Тодорхой хугацааны дараа уралдааныг дуусгах.
- **Хэрэглэгчийн үнэлгээг харуулах** - Хэрэглэгчийн мэдээллийг боловсруулж мэдээллийг харуулах.
- **Хэрэглэгчийн мэдээллийг хадгалах** - Тоглоом дууссаны дараагаар хэрэглэгчийн мэдээллийг боловсруулж мэдээллийг хадгалах.

3.4 UX/UI дизайн

Өмнөх бүлэгт хийсэн судалгааны зорилго бол хэрэглэгчдэд ээлтэй, харагдахуйц харагдахуйц интерфэйсийг бий болгох явдал байсан.

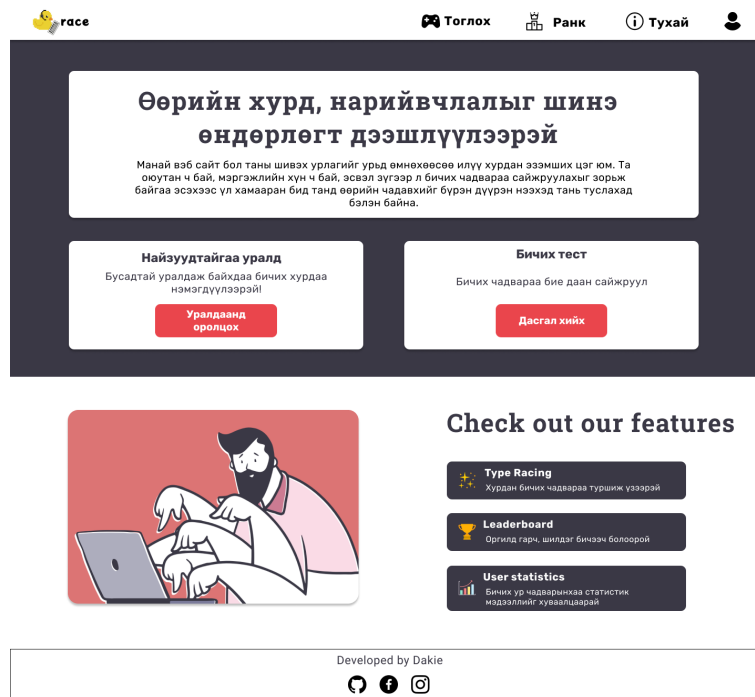
- хэрэглэгчдийн хэрэгцээ, хүсэл сонирхолд нарийвчлан дүн шинжилгээ хийж, бидний дизайн, хөгжүүлэлт зорилтот хэрэглэгчдэдээ хамгийн сайн туршлагыг бий болгоход чиглэсэн.
- Шинэлэг, мэдрэмжтэй хэрэглэгчийн интерфэйс (UI) загварыг бий болгоход хүчин чармайлтаа зориулсан.
- Дизайнаас хөгжүүлэлт рүү жигд шилжихийг хөнгөвчлөхийн тулд UI дизайнтай нягт уялдуулах замаар урд талын хөгжүүлэлтийн процессыг хурдасгасан.

3.4.1 Эхний загвар

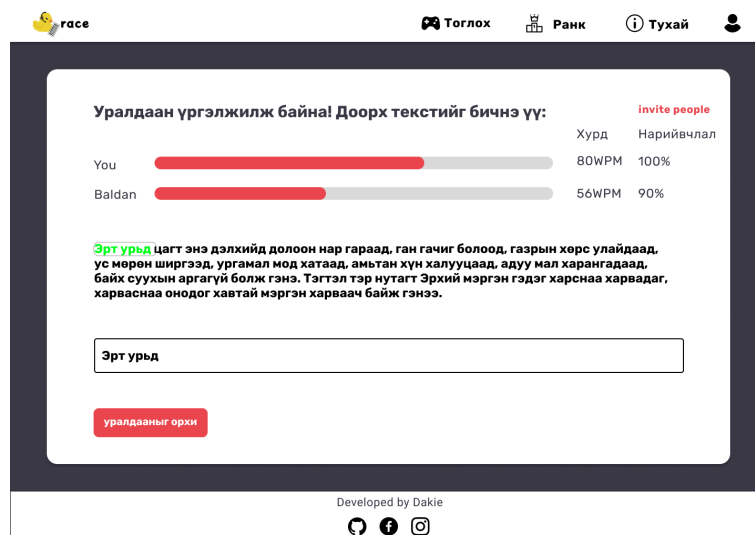
Эхний Wireframe загварыг гаргаж Prototype хувилбар бүтээсэн нь

Вебийн гол процессийг илэрхийлэх 4 зургийг оруулаа. Бусад хэсгийг хавсралтаас ¹ үзэх боломжтой.

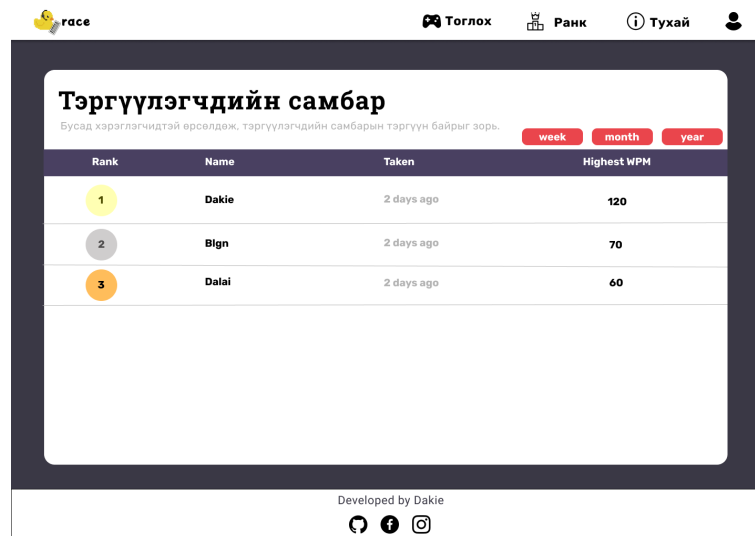
¹Зурсан интерфэйс дизайн <https://shorturl.at/djwAT>



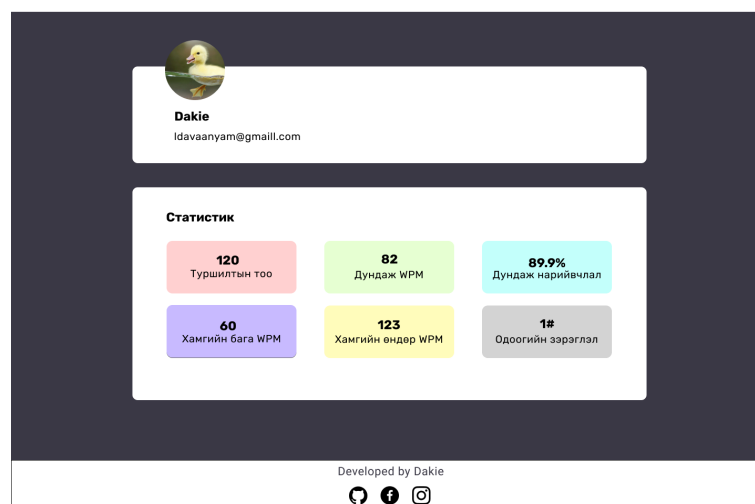
Зураг 3.5: Нүүр хуудас



Зураг 3.6: Уралдах хэсэг



Зураг 3.7: Тэргүүлэгчдийн хэсэг

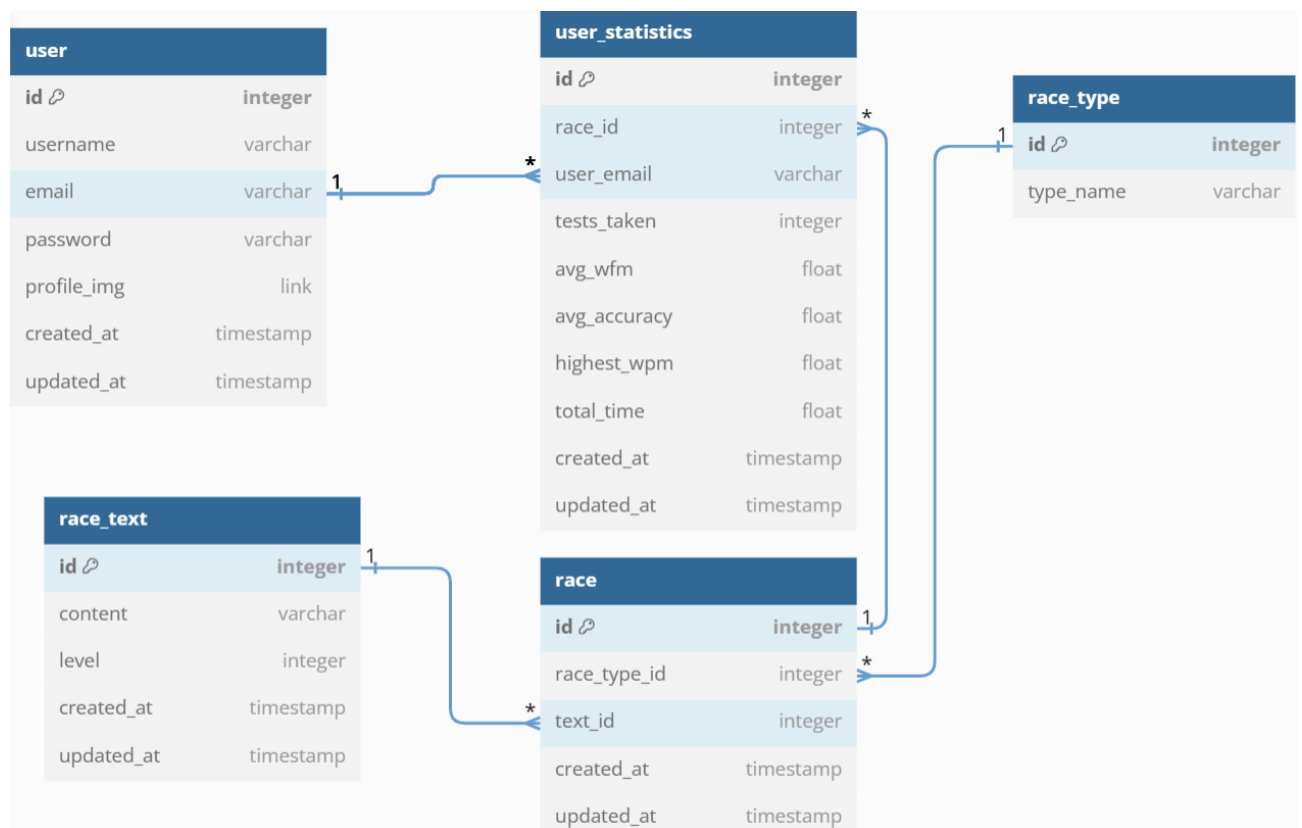


Зураг 3.8: Профайл хэсэг

3.5 Өгөгдлийн сан

Өгөгдлийн сангийн хувьд манай веб апп нь Firebase realtime өгөгдлийн бааз ашиглаж байгаа учир NoSQL төрлийн өгөгдлийн баазад өгөгдлөө "JSON" хэлбэрээр "Firebase realtime" өгөгдлийн баазд хадгалах юм.

3.5.1 Өгөгдлийн сангийн диаграм



Зураг 3.9: Системийн баазын бүтэц

Уралдааны төрөл хадгалах өгөгдлийн баазын JSON

```
"race_type": [  
  {  
    "type_name": "multiplayer"  
  },  
  {  
    "type_name": "singleplayer"  
  }  
],
```

Зураг 3.10: Хэрэглэгчийн үзүүлэлтийн жишээ дата

Уралдааны текст хадгалах өгөгдлийн баазын JSON

```
"race_text": [  
  {  
    "content":  
    "Эрт цагт нэгэн тосгонд тариачин хүү аавын хамт амьдран суудаг байжээ. Хүүгийн Аав ма  
ш хүнд өвчтэй байв. Нэгэн өдөр тэрээр хүүгээ дуудаж хүү минь миний бие өдөр ирэх туса  
м муудаж байна. Тиймээс би чамд үүнийг өгөх цаг болжээ гэж хэлээд нэгэн эвэр түүнд бэ  
лэглэв."  
    ,  
    "created_at": "Mon Nov 27 2023 09:50:18 GMT+0800 (Ulaanbaatar Standard Time)",  
    "level": 1,  
    "updated_at": "Mon Nov 27 2023 09:50:18 GMT+0800 (Ulaanbaatar Standard Time)"  
  }  
],
```

Зураг 3.11: Хэрэглэгчийн үзүүлэлтийн жишээ дата

Уралдаан хадгалах өгөгдлийн баазын JSON

```
{
  "race": [
    {
      "created_at": "Mon Nov 27 2023 09:50:18 GMT+0800 (Ulaanbaatar Standard Time)",
      "race_type_id": 1,
      "text_id": 1,
      "updated_at": "Mon Nov 27 2023 09:50:18 GMT+0800 (Ulaanbaatar Standard Time)"
    }
  ]
}
```

Зураг 3.12: Уралдааны жишээ дата

Хэрэглэгчийн үзүүлэлтийг хадгалах өгөгдлийн баазын JSON

```
"user_statistics": [
  {
    "avg_accuracy": "72.5",
    "avg_wfm": "54",
    "created_at": "Mon Nov 27 2023 09:50:18 GMT+0800 (Ulaanbaatar Standard Time)",
    "highest_wpm": 76,
    "race_id": 0,
    "total_time": 90,
    "updated_at": "Mon Nov 27 2023 09:50:18 GMT+0800 (Ulaanbaatar Standard Time)",
    "user_email": "ldavaanyam@gmail.com"
  }
]
```

Зураг 3.13: Хэрэглэгчийн үзүүлэлтийн жишээ дата

Өгөгдлийн сангийн хүснэгтүүдийн тайлбар

Хүснэгт 3.1: User хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	int	Хэрэглэгчийн дахин давтагдашгүй ID-г хадгална - Auto Incremented
2	username	varchar	Хэрэглэгчийн гараас оруулж өгсөн нэрийг хадгална. Зөвхөн латин тэмдэгтүүд ашиглах хэрэгтэй
3	email	varchar	Хэрэглэгчийн цахим шуудан
4	password	varchar	Хэрэглэгчийн гараас оруулж өгсөн нууц үгийг encrypt-лэж /hash/ уг хэсэгт хадгална
5	profile_img	link	Хэрэглэгчийн оруулж өгсөн зургийг сервер дээр хадгалж, замыг нь энэ хэсэгт хадгална
6	created_at	timestamp	Хэрэглэгчийн хаяг үүссэн хугацааг серверээс авч хадгална
7	updated_at	timestamp	Хэрэглэгчийн хаяг өөрчлөгдсөн хугацааг серверээс авч хадгална

Хүснэгт 3.2: User Statistics Table

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	int	Хэрэглэгчийн статистикийн дахин давтагдашгүй ID-г хадгална - Auto Incremented
2	user_race_id	int	Хэрэглэгчийн статистикийг уралдааны хүснэгттэй Foreign Key-р хадгална
3	user_email	int	Хэрэглэгчийн статистикийг хэрэглэгчийн хүснэгттэй Foreign Key-р хадгална
4	tests_taken	int	Хэрэглэгчийн авсан тестийн нийт тоог хадгална
5	avg_wpm	float	Хэрэглэгчийн шивэх хурдны минутын дундаж үгийг хадгална.
6	avg_accuracy	float	Хэрэглэгчийн шивэх дундаж нарийвчлалын хувийг хадгална
7	high_wpm	float	Хэрэглэгчийн шивэх хурдад хүрсэн минутанд хамгийн их үгийг хадгална
8	total_time	float	Хэрэглэгчийн уралдаанд зохиулсан хугацааг хадгална
9	created_at	timestamp	Хэрэглэгчийн хаяг үүссэн хугацааг серверээс авч хадгална
10	updated_at	timestamp	Хэрэглэгчийн хаяг өөрчлөгдсөн хугацааг серверээс авч хадгална

Хүснэгт 3.3: Race Table

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	int	Уралдааны дахин давтагдашгүй ID-г хадгална - Auto Incremented
2	race_type_id	int	Тэмцээний төрлийн хүснэгттэй Foreign Key-р хадгална
3	text_id	int	Тэмцээнд ашигласан текстийг зааж өгсөн race_text хүснэгттэй Foreign Key-р хадгална
4	created_at	timestamp	Хэрэглэгчийн хаяг үүссэн хугацааг серверээс авч хадгална
5	updated_at	timestamp	Хэрэглэгчийн хаяг өөрчлөгдсөн хугацааг серверээс авч хадгална

Хүснэгт 3.4: Race Type Table

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	int	Уралдааны төрлийн дахин давтагдашгүй ID-г хадгална - Auto Incremented
2	type_name	varchar	Тэмцээний төрлийн нэрийг хадгалах хувьсагчийн тэмдэгтийн багана

Хүснэгт 3.5: Race Text Table

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	int	Уралдааны өгүүлбэрийн дахин давтагдашгүй ID-г хадгална - Auto Incremented
2	контент	varchar	Тэмцээний үеэр хэрэглэгчдийн шивэх бодит текстийг хадгалах хувьсах тэмдэгтийн багана.
3	level	int	Текстийн хүндрэлийн түвшинг харуулах бүхэл багана
4	created_at	timestamp	Хэрэглэгчийн хаяг үүссэн хугацааг серверээс авч хадгална
5	updated_at	timestamp	Хэрэглэгчийн хаяг өөрчлөгдсөн хугацааг серверээс авч хадгална

4. ХЭРЭГЖҮҮЛЭЛТ

Өмнөх бүлгүүдэд гаргасан "UI" болон "UX" ийн дизайниуд, системийн архитектур болон диаграммууд, шаардлагуудыг хэрэгжүүлж бүтээгдэхүүн болгон гаргасан үе шатуудын талаар энэхүү бүлэгтээ дурдана. Хэрэгжүүлэлт хийх үе шатаа ерөнхийд нь задалвал:

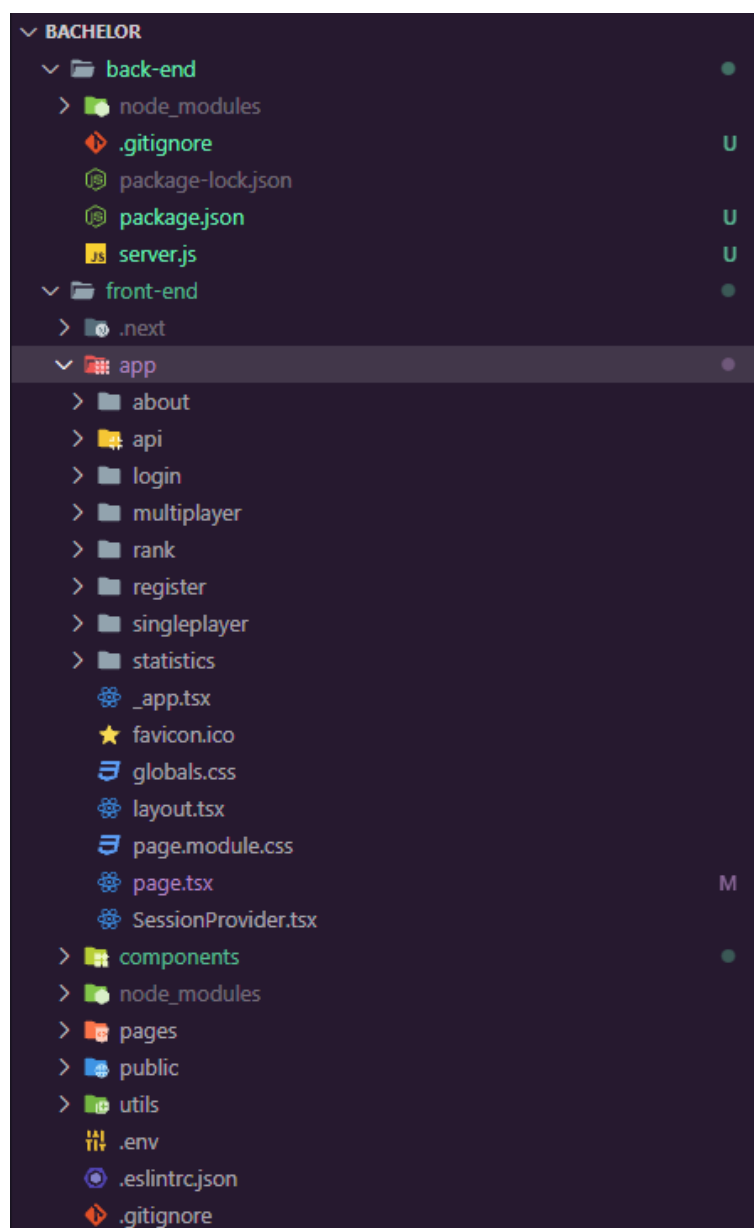
- Хөгжүүлэлтийн орчноо бэлдэх.
- Front-end хөгжүүлэлт.
- Back-end хөгжүүлэлт.
- Серверт байршуулах.

гэсэн алхмуудад хувааж гүйцэтгэсэн ба доор ажлуудаасаа гол гэсэн зүйлсээ нэгтгэн оруулав.

4.1 Хөгжүүлэлтийн орчин

Хэрэглэгч талд хэрэглэгдэх буюу "Front-end" талын хэрэгжүүлэлтээ "Next.js" технологи ашигласан ба харин "Back-end" талын хөгжүүлэлтийг NodeJS, Socket.io " технологи, өгөгдлөө "Firebase" платформ дээр "NoSQL" төрлөөр хадгалсан. Иймээс хөгжүүлэлтийн орчны хувьд "Next.js" фрэймворкоо суулгаж, шаардлагатай тохиргоонуудыг хийж мөн адил NodeJS болон Socket.io"-ийг суулгаж, хэрэгтэй ашиглах сангуудаа татаж суулгаад, өгөгдлийн баазаа "Firebase" дээр үүсгэж серверийг асаасан.

Хөгжүүлэлтийн явцын кодыг "Github" болон "Version Control" буюу "Git" гэсэн технологиудыг сонгож хадгалсан ба хадгалах "Repository" - ийг "private" горимтой үүсгэсэн.



Зураг 4.1: Төслийн файлын бүтэц

Next.js дээр файлын бүтцээ Зураг 4.1 дээр харагдаж байгаачлан бүтэцлэж хөгжүүлэлтээ эхлэхэд бэлэн болголоо. Төслийн хувь "Back-end" талын хөгжүүлэлт "NodeJS" хэл дээр бичигдсэн учир төслийн хавтас маань хоёр салж "front-end" "back-end" гэж тусгаарлаж өгсөн. Файлын болон хавтсуудын бүтцийг тайлбарлавал

- **back-end/server.js** төслийн "NodeJS" дээр бичигдсэн "back-end" хэрэгжүүлэлтийн код байршина.
- **public** хавтаст "front-end" төслийн хүрээнд ашиглах статик файлууд /зураг, icon/

байна.

- **component** хавтаст "front-end" төслийн бүх компонент кодууд байршина.
- **app** хавтаст "front-end" төслийн бүх хуудсууд, гол эх кодууд байршина.
- **pages/api/auth** Төсөлд хэрэглэгчийн хаяг, "session", "authentication" зэргийг зохицуулах код байршина.

4.2 Код хөгжүүлэлт

Хөгжүүлэлтийн хувьд маш олон компонент, хуудсууд, тэдгээрийн код хийгдсэн тул хэрэглэгч шивэх үйл явцыг тайлбарлав.

4.2.1 Гаргасан интерфэйс дээр тулгуурлаж гаргасан шивэх компонент хэсгийн хэрэгжүүлэлт

Хамгийн эхлээд бэлдсэн хөгжүүлэлтийн орчин дээрээ хэрэглэгчийн шивэх үндсэн модал болон дотор нь ашиглаж буй компонентуудыг статик байдлаар өрөв.

```

1  const PlayContent = (props: any) => {
2    const { name, progress, wpm, completion, style } = props;
3    return (
4      <div className={` ${style.playStatus} ${rubik.className}`} >
5        <Typography
6          variant="body1"
7          fontSize={18}
8          className={style.playStatusName}
9        >
10         {name}
11       </Typography>
12       <div className={style.progress_bar}>
13         <div
14           className={style.progress}
15           style={{ width: `${progress}%` }}
16         ></div>

```

```

17     </div>
18     <div className={style.playContentEnd}>
19         <Typography
20             variant="body1"
21             fontSize={18}
22             className={style.playContentBodyText}
23         >
24             {wpm}WPM
25         </Typography>
26         <Typography
27             variant="body1"
28             fontSize={18}
29             className={style.playContentBodyText}
30         >
31             {completion}%
32         </Typography>
33     </div>
34 </div>
35 );
36 };

```

Код 4.1: Хэрэглэгчийн мэдээллийг бодит хугацаанд харуулах
КОМПОНЕНТ

Мэдээлэлд тоглогчийн нэр (name), хувь (progress), минут тутамд үг (wpm), гүйцэтгэлийн хувь (accuracy) зэрэг орно.

4.2.2 Client хэсгийн Socket.io өрөө үүсгэх, орж буй хэрэгжүүлэлт

Уралдааны хуудас рэндэр хийгдэх үед Socket ашиглан unique өрөөг үүсэх. Хэрвээ үүссэн өрөөнд орох бол URL аас param ийг шалган өрөөнд нэвтрүүлнэ.

```

1  useEffect(() => {
2      const socket = io("https://typeracer-ytd7.onrender.com");
3      setSocket(socket);
4      const setupSocket = () => {

```

```

5      if (searchParams!.get("roomId")) {
6          const roomId = searchParams!.get("roomId");
7          socket!.emit("join-room", roomId);
8
9          socket.on("room-joined", (userCount, roomTextId) => {
10              // setTextId(roomTextId);
11          });
12      } else {
13          socket!.emit("create-room", textId);
14      }
15
16      return () => {
17          socket!.disconnect();
18      };
19  };
20
21  setupSocket();
22  }, [textId]);

```

Код 4.2: Хэрэглэгч өрөөнд нэвтрэх client талын код

4.2.3 Өгөгдлийн сангаас текст харуулах функц

Өрөөнд амжилттай орсон үед баазаас `race_text` гэх json оос санамсаргүй байдлаар сонгон авч `useEffect` ашиглан текстийн талбарт оноож өгнө. `useEffect` нь empty massive хамаарлын (`[]`) заасны дагуу бүрэлдэхүүнийг холбох үед нэг удаа ажиллаж өгөгдлийг дуудаж, боловсруулдаг.

```

1  useEffect(() => {
2      const fetchData = async () => {
3          try {
4              const race_text = await fetch(
5                  "https://typeracer-1be53-default-rtdb.asia-southeast1.
6                      firebase.database.app/race_text.json"
7              );

```

```
7
8     const race = await fetch(
9         "https://typeracer-1be53-default-rtdb.asia-southeast1.
10         firebase.database.app/race.json"
11     );
12
13     const user_statistics = await fetch(
14         "https://typeracer-1be53-default-rtdb.asia-southeast1.
15         firebase.database.app/user_statistics.json"
16     );
17
18     if (race_text.ok) {
19         const data = await race_text.json();
20         const id = Math.floor(Math.random() * data.length);
21         const quote = data[id];
22
23         setTextId(id);
24         setWords(quote.content);
25     } else {
26         console.error(
27             `Error: Unable to fetch data. Status Code: ${race_text.status}`
28         );
29     }
30
31     if (race.ok) {
32         const race_data = await race.json();
33         const raceId = race_data.length;
34         setRaceId(raceId);
35     }
36
37     if (user_statistics.ok) {
38         const user_data = await user_statistics.json();
39         const userId = user_data.length;
40         setUserId(userId);
41     }
42 } catch (error) {
```

```
39     console.error("Error fetching data:", error);
40   }
41 };
42 fetchData();
43 }, []);
```

Код 4.3: Текст fetch хийж буй async функц

4.2.4 *Firebase тохиргоо*

Firebase Config нь хэрэглэгчээс програмын шинэчлэлтийг татаж авах шаардлагагүйгээр өөрийн апп-ын төлөв байдал, харагдах байдлыг өөрчлөх боломжийг олгодог. Энэхүү код нь Firebase програм, бодит цагийн өгөгдлийн санг (db) болон баталгаажуулалтын жишээг (auth) програмын бусад хэсэгт ашиглахаар экспортолно.

```
1  import { getApp, getApps, initializeApp } from "firebase/app";
2  import { getDatabase } from "firebase/database";
3  import { getAuth } from "firebase/auth";
4
5  const firebaseConfig = {
6    apiKey: process.env.NEXT_PUBLIC_FIREBASE_API_KEY,
7    authDomain: process.env.NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN,
8    databaseURL: process.env.NEXT_PUBLIC_FIREBASE_DATABASE_URL,
9    projectId: process.env.NEXT_PUBLIC_FIREBASE_PROJECT_ID,
10   storageBucket: process.env.NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET,
11   messagingSenderId: process.env.NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID
12   ,
13   appId: process.env.NEXT_PUBLIC_FIREBASE_APP_ID,
14 };
15
16 const firebaseApp = getApps().length ? getApp() : initializeApp(
17   firebaseConfig);
18 const db = getDatabase(firebaseApp);
19 const auth = getAuth();
```

```
19 export { firebaseApp, db, auth };
```

Код 4.4: Firebase config

4.2.5 Тоглоомын удирдах hook

Энэхүү useEffect hook нь статус дээр тулгуурлан янз бүрийн зан үйлийг удирддаг. Статус "playing" үед setInterval-ийг ашиглан цагийн хязгаарыг тохируулж, TIME_LIMIT хүрэх хүртэл цагийн төлөвийг секунд тутамд нэмэгдүүлнэ. Хэрэв статус "completed" эсвэл "finished" бол энэ нь интервалыг арилгаж, төлөвийг шинэчилж, горимыг харуулж, тохируулсан функцийг ашиглан Firebase бодит цагийн мэдээллийн сан дахь өгөгдлийг шинэчилдэг.

```
1  useEffect(() => {
2    let timeLimit: any;
3    if (status === "playing") {
4      if (inputRef.current) {
5        inputRef.current.focus();
6      }
7      timeLimit = setInterval(() => {
8        setTime((prev) => {
9          if (prev >= TIME_LIMIT) {
10             clearInterval(timeLimit);
11             setStatus("finished");
12             return TIME_LIMIT;
13           } else {
14             return prev + 1;
15           }
16         });
17     }, 1000);
18   } else if (status === "completed") {
19     clearInterval(timeLimit);
20     setFailed(false);
21     setShowModal(true);
22   }
```

```
23     set(ref(db, "race/" + raceId), {
24         creator_id: session?.data?.user?.email,
25         race_type_id: 1,
26         text_id: textId,
27         created_at: Date.now(),
28         updated_at: Date.now(),
29     });
30
31     set(ref(db, "user_statistics/" + userId), {
32         user_id: session?.data?.user?.email,
33         avg_wfm: calculateWPM(time, wordIndex).currentWPM,
34         avg_accuracy: calculateAccuracy(correct, wrong),
35         highest_wpm: calculateWPM(time, wordIndex).highestWPM,
36         lowest_wpm: calculateWPM(time, wordIndex).lowestWPM,
37         created_at: Date.now(),
38         updated_at: Date.now(),
39     });
40 } else if (status === "finished") {
41     clearInterval(timeLimit);
42     setFailed(true);
43     setShowModal(true);
44
45     set(ref(db, "race/" + raceId), {
46         creator_id: session?.data?.user?.email,
47         race_type_id: 1,
48         text_id: textId,
49         created_at: Date.now(),
50         updated_at: Date.now(),
51     });
52
53     set(ref(db, "user_statistics/" + userId), {
54         user_id: session?.data?.user?.email,
55         avg_wfm: calculateWPM(time, wordIndex).currentWPM,
56         avg_accuracy: calculateAccuracy(correct, wrong),
```

```

57     highest_wpm: calculateWPM(time, wordIndex).highestWPM,
58     lowest_wpm: calculateWPM(time, wordIndex).lowestWPM,
59     created_at: Date.now(),
60     updated_at: Date.now(),
61   });
62 }
63
64   return () => clearInterval(timeLimit);
65 }, [status]);

```

Код 4.5: Тоглоомын цагийг удирдах Firebase өгөгдлийг шинэчлэхэд зориулсан useEffect hook

4.2.6 Next-Auth ашиглаж хэрэглэгчийн Authentication, Session зохицуулах

Цахим шуудан болон нууц үг ашиглан баталгаажуулалтын процессыг хариуцдаг "Credentials" нэрт тусгай баталгаажуулалтын үйлчилгээг зааж өгч, NextAuth болон Firebase ашиглаж хэрэглэгчийн Authenticationийг зохицуулсан.

```

1   export const authOptions = {
2     pages: {
3       signIn: "/login",
4     },
5
6     providers: [
7       CredentialsProvider({
8         name: "Credentials",
9         credentials: {},
10        async authorize(credentials): Promise<any> {
11          return await signInWithEmailAndPassword(
12            auth,
13            (credentials as any).email || "",
14            (credentials as any).password || ""
15          )
16          .then((userCredential) => {

```



```

17         if (userCredential.user) {
18             return userCredential.user;
19         }
20         return null;
21     })
22     .catch((error) => console.log(error))
23     .catch((error) => {
24         const errorCode = error.code;
25         const errorMessage = error.message;
26         console.log(error);
27     });
28 },
29 ),
30 ],
31 };
32
33 export default NextAuth(authOptions);

```

Код 4.6: Хэрэглэгчийн Authentication болон Session-г зохицуулах

4.2.7 Socket.io болон ExpressJS ашигласан back-end талын хөгжүүлэлт

Энэхүү Node.js серверийн код нь хэрэглэгч хоорондох бодит цагийн харилцааг удирдахын тулд Socket.IO санг ашигласан. Энэ нь хэрэглэгчдэд өрөө бүрийг санамсаргүй ID-аар тодорхойлсон өрөө үүсгэх, нэгдэх боломжийг олгоно. Сервер нь өрөө тус бүрд холбогдсон хэрэглэгчдийг хянаж, өрөө үүсгэх, нэгдэх үйл явдлуудыг зохицуулж, хэрэглэгчид гарах үед өрөөг устгадаг.

```

1
2 io.on("connection", (socket) => {
3     console.log("A user connected: " + socket.id);
4
5     socket.on("create-room", (textId) => {
6         const roomId = Math.random().toString(36).substring(2, 7);
7

```

```
8     socket.join(roomId);
9
10    roomUsers[roomId] = {
11      textId: textId,
12      users: [],
13    };
14    roomUsers[roomId].users.push(socket.id);
15    console.log(`User created and joined room ${roomId}`);
16
17    io.to(roomId).emit("room-created", roomId);
18    console.log(roomUsers);
19  });
20
21  socket.on("join-room", (roomId) => {
22    socket.join(roomId);
23    console.log(roomUsers);
24
25    if (!roomUsers[roomId]) {
26      roomUsers[roomId] = { textId: 0, users: [] };
27    }
28
29    roomUsers[roomId].users.push(socket.id);
30
31    io.to(roomId).emit(
32      "room-joined",
33      roomUsers[roomId].users.length,
34      roomUsers[roomId].textId
35    );
36
37    console.log(`${roomId} өрөөнийхөн ${roomUsers[roomId].users}`);
38  });
39
40  socket.on("disconnect", () => {
41    console.log("User disconnected");
```

```
42   Object.keys(roomUsers).forEach((room) => {
43     roomUsers[room].users = roomUsers[room].users.filter(
44       (user) => user !== socket.id
45     );
46     console.log(
47       `Users in room ${room} after disconnect:`,
48       roomUsers[room].users
49     );
50
51     if (roomUsers[room].users.length === 0) {
52       delete roomUsers[room];
53     }
54   });
55 });
56 });
```

Код 4.7: Socket ашиглан хэрэглэгчдийг зохицуулах

4.2.8 Серверт байршуулах

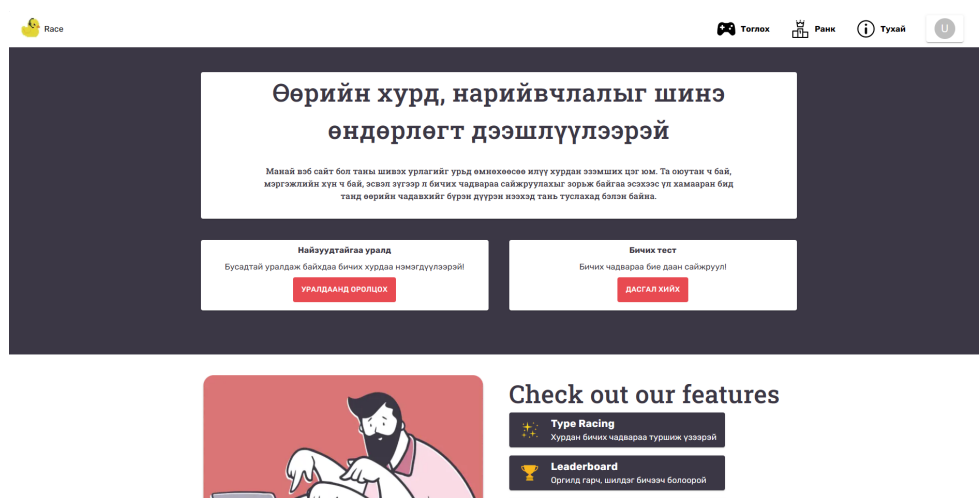
Хийсэн төслийнхөө эцсийн хувилбарыг онлайн хэлбэрээр серверт байршуулж хэрэглэгчид шууд ашиглах боломжтой байдаар серверт байршуулсан ба төслийнхөө "Front-end" хэсгийг "Vercel.com" дээр байршуулсан бөгөөд хэрэглэгч хүсэлт илгээхэд "Back-end" серверээс зохицуулалт хийж хүсэлтэд хариулт илгээж байх ёстой учир тусдаа сервер дээр "Backend" хэсгийн кодоо render.com дээр асаав.

- **Website link:** <https://duckrace.vercel.app/>

5. ҮР ДҮН

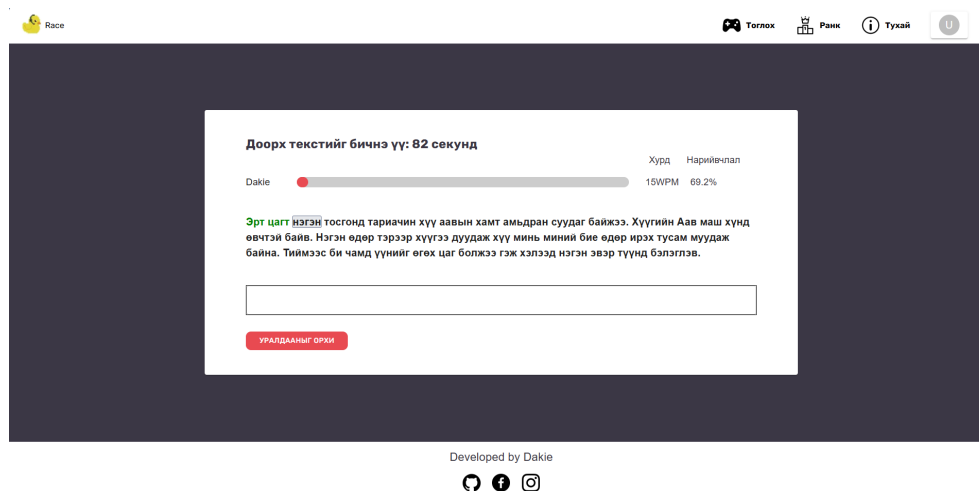
Дээрх хэрэгжүүлэлтийн үр дүнд дараах зургуудад харагдаж байгаачлан манай веб апп маань ашиглахад бэлэн болсон. Доор гол гэсэн хуудсуудын ажиллаж буй процессыг илэрхийлэх хуудсуудын зургуудуудыг оруулав.

Хэрэглэгч рендер хийхэд харагдах хамгийн эхний нүүр хуудас



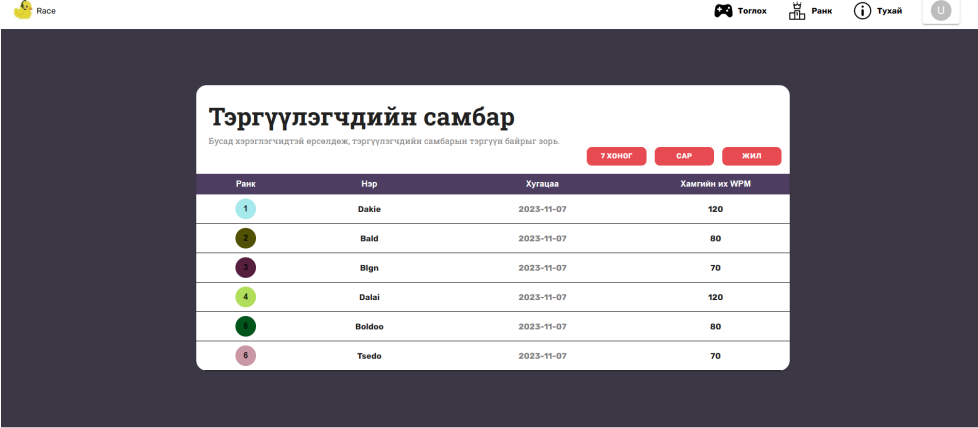
Зураг 5.1: Нүүр хуудас

Singleplayer буюу зөвхөн өөрөө тест өгөх хуудас



Зураг 5.2: Хэрэглэгч шивэх тест өгөх хуудас

Нийт хэрэглэгчдийг үзүүлэлтээр нь эрэмбэлж харуулах хуудас



Тэргүүлэгчдийн самбар

Бусад хэрэглэгчдтэй өрсөлдөж, тэргүүлэгчдийн самбарын тэргүүн байрыг зорь.


7 хоног сар жиж

Ранк	Нэр	Хугацаа	Хамгийн их WPM
1	Dakie	2023-11-07	120
2	Bald	2023-11-07	80
3	Blign	2023-11-07	70
4	Dalai	2023-11-07	120
5	Boldoo	2023-11-07	80
6	Tsedo	2023-11-07	70

Developed by Dakie

Зураг 5.3: Нийт тоглогчдыг тэргүүлэгчдийн хуудас

Хэрэглэгч өөрийн үзүүлэлтүүдийг харах хуудас



Dakie

ldavaanyam@gmail.com

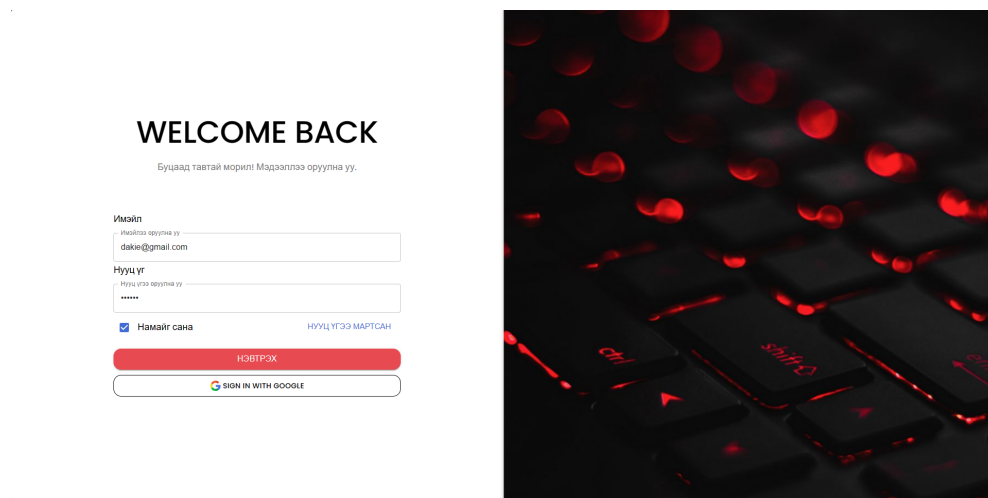
Статистик

120 Туршилтын тоо	120 Дундаж WPM	120 Дундаж нарийвчлал
120 Хамгийн бага WPM	120 Хамгийн өндөр WPM	1 Одоогийн зэрэглэл

Developed by Dakie

Зураг 5.4: Хэрэглэгчийн статистикийн хуудас

Хэрэв хэрэглэгч нэвтрээгүй тохиолдолд рендер хийгдэх хуудас



Зураг 5.5: Нэвтрэх хуудас

6. ДҮГНЭЛТ

Энэхүү судалгааны ажлын хүрээнд программ хангамжийн бүтээгдэхүүн боловсруулах арга, технологиос суралцах, хэрэглэгчдийн судалгаанд үндэслэн хэрэглэгчийн шаардлагыг тодорхойлох, хэрэглэгчийн интерфейс, хэрэглэгчийн туршлагын дизайны шаардлагыг хангаж, өрсөлдөөнт түргэн бичилтийн вэб аппыг миний бие дуусгаж чадсан нь өөрт маань асар их хэмжээний мэдлэг, туршлага болж чадсан.

DuckRacer нь шивэх чадварыг сайжруулах, олон нийттэй харилцах, нөхөрсөг өрсөлдөөнийг бий болгоход суралцах туршлагыг тоглоом болгож, сайжруулж байгаагаараа онцлог юм. Энэ нь практик шивэх чадварыг хөгжүүлэхийг эрэлхийлж буй хувь хүмүүс, сургууль, ажлын байруудад хүртээмжтэй, зугаатай хэрэгсэл болно гэдэгт итгэлтэй байна.

Мөн цаашлаад уг төсөл маань олон хүнд хүрсэн, хурдан шивэх зорилготой хүмүүст нас харгалзахгүй өргөн хэмжээний боломжыг олгож чадсан олон хэл дээр хөгжүүлж хүссэн хэлийнхээ шивэх дасгалыг хийх хоорондоо уралдах систем болгон хөгжүүлэх бүрэн боломжтой гэж үзэж байна.

Ашигласан материал

- [1] Next.js. *Next.js Documentation*. Retrieved from <https://nextjs.org/docs>
- [2] Maximilian Schwarzmüller. *Next.js: The Complete Introduction* (Udemy course). Retrieved from <https://www.udemy.com/course/nextjs-react-the-complete-guide/>
- [3] Lee Robinson. *Learning Next.js* (Free online book). Retrieved from <https://www.learnnextjs.com/>
- [4] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice* (Book).
- [5] Firebase. *Firebase Documentation*. Retrieved from <https://firebase.google.com/docs>

A. ӨГӨГДӨЛ БОЛОВСРУУЛАХ ФУНКЦ

```
1 let wpmValues: number[] = [];  
2  
3 export const calculateWPM = (  
4   time_taken: number,  
5   num_words: number  
6 ): { currentWPM: string | number; highestWPM: number; lowestWPM: number }  
   => {  
7   const wpm: string | number =  
8     time_taken > 0 ? ((num_words / time_taken) * 60).toFixed(0) : 0;  
9  
10  wpmValues.push(Number(wpm));  
11  
12  const highestWPM: number = Math.max(...wpmValues);  
13  const lowestWPM: number = Math.min(...wpmValues);  
14  
15  return {  
16    currentWPM: wpm,  
17    highestWPM,  
18    lowestWPM,  
19  };  
20 };  
21  
22 export const calculateAccuracy = (  
23   correct: number,  
24   wrong: number  
25 ): string | number => {  
26   return correct + wrong > 0  
27     ? ((correct / (correct + wrong)) * 100).toFixed(1)  
28     : 0;  
29 };  
30  
31 export const timeAgo = (curr_date: string): string => {  
32   const seconds = Math.floor(  
33     (new Date().getTime() - new Date(curr_date).getTime()) / 1000  
34   );  
35  
36   const time_tables = [  
37     { label: "year", div: 31536000 },  
38     { label: "month", div: 2592000 },  
39     { label: "day", div: 86400 },  
40     { label: "hour", div: 3600 },  
41     { label: "minute", div: 60 },  
42   ];  
43  
44   for (const table of time_tables) {  
45     const { label, div } = table;  
46     const interval = seconds / div;  
47     if (interval >= 1) {  
48       return `${Math.floor(interval).toString()} ${label}${  
49         interval > 1 ? "s" : ""  
50       } ago`;  
51     }  
52   }  
53   return "Just now";  
54 };
```

В. ӨГӨГДЛИЙГ АШИГЛАХ КОМПОНЕНТ

```
1   <ResultModal
2     open={showModal}
3     onClose={handleCloseModal}
4     handleRetry={handleRetry}
5     handleLeave={handleLeave}
6     wpm={calculateWPM(time, wordIndex).currentWPM}
7     time={time}
8     accuracy={calculateAccuracy(correct, wrong)}
9     failed={failed}
10  />
11
12  interface ResultModalProps {
13    open: boolean;
14    onClose: () => void;
15    handleRetry: () => void;
16    handleLeave: () => void;
17    wpm: any;
18    time: any;
19    accuracy: any;
20    failed: any;
21  }
22
23  function ResultModal({
24    open,
25    onClose,
26    handleRetry,
27    handleLeave,
28    wpm,
29    time,
30    accuracy,
31    failed,
32  }:)
```

C. MUI КОМПОНЕНТ АШИГЛАХ

```
1 import Typography from "@mui/material/Typography";
2 import Button from "@mui/material/Button";
3 import { Grid } from "@mui/material";
4
5 return (
6   <>
7     <Container>
8       <Card className={style.cardBody}>
9         <Typography
10           variant="h3"
11           fontWeight={"bold"}
12           className={roboto.className}
13           pt={4}
14           pl={3}
15           pb={1}
16         >Тэргүүлэгчдийнсамбар
17       </Typography>
18       <Typography
19         variant="body1"
20         className={roboto1.className}
21         pl={3}
22         color={"gray"}
23       >Бусадхэрэглэгчидтэйөрсөлдөж
24         , тэргүүлэгчдийнсамбарынтэргүүнбайрыгзорь
25         .
26       </Typography>
27       <div className={style.buttons}>
28         <Button
29           variant="text"
30           className={` ${rubik.className} ${style.filterBtn}`}
31         >
32           7 хоног
33         </Button>
34         <Button
35           variant="text"
36           className={` ${rubik.className} ${style.filterBtn}`}
37         >cap
38       </Button>
39       <Button
40         variant="text"
41         className={` ${rubik.className} ${style.filterBtn}`}
42       >жил
43     </Button>
44   </div>
45 </Container>
46 </>
47 );
```

D. AUTH SESSION ЗОХИЦУУЛАХ

```
1  export const authOptions = {
2  pages: {
3    signIn: "/login",
4  },
5
6  providers: [
7    CredentialsProvider({
8      name: "Credentials",
9      credentials: {},
10     async authorize(credentials): Promise<any> {
11       return await signInWithEmailAndPassword(
12         auth,
13         (credentials as any).email || "",
14         (credentials as any).password || ""
15       )
16       .then((userCredential) => {
17         if (userCredential.user) {
18           return userCredential.user;
19         }
20         return null;
21       })
22       .catch((error) => console.log(error))
23       .catch((error) => {
24         const errorCode = error.code;
25         const errorMessage = error.message;
26         console.log(error);
27       });
28     },
29   }),
30 ],
31 };
32
33 export default NextAuth(authOptions);
```

E. AUTH SESSION АШИГЛАХ

```
1  "use client";
2
3  import styles from "../page.module.css";
4  import Navbar from "@components/navbar/navbar";
5  import MainPage from "@components/mainpage/mainpage";
6  import Footer from "@components/footer/footer";
7  import { useSession } from "next-auth/react";
8  import { redirect } from "next/navigation";
9
10 export default function Home() {
11   const session = useSession({
12     required: true,
13     onUnauthenticated() {
14       redirect("/login");
15     },
16   });
17
18   return (
19     <div>
20       <Navbar></Navbar>
21       <MainPage></MainPage>
22       <Footer></Footer>
23     </div>
24   );
25 }
26
27 Home.requireAuth = true;
```

F. SOCKET ДЭЭР CORS ЗӨВШӨӨРӨХ

```
1  const app = express();
2  app.use(cors());
3
4  const server = createServer(app);
5  const io = new Server(server, {
6    cors: {
7      origin: "*",
8      methods: "*",
9    },
10  });
```

G. SOCKET SERVER

```
1
2 const roomUsers = {};
3
4 io.on("connection", (socket) => {
5   console.log("A user connected: " + socket.id);
6
7   socket.on("create-room", (textId) => {
8     const roomId = Math.random().toString(36).substring(2, 7);
9
10    socket.join(roomId);
11
12    roomUsers[roomId] = {
13      textId: textId,
14      users: [],
15    };
16    roomUsers[roomId].users.push(socket.id);
17    console.log(`User created and joined room ${roomId}`);
18
19    io.to(roomId).emit("room-created", roomId);
20    console.log(roomUsers);
21  });
22
23  socket.on("join-room", (roomId) => {
24    socket.join(roomId);
25    console.log(roomUsers);
26
27    if (!roomUsers[roomId]) {
28      roomUsers[roomId] = { textId: 0, users: [] };
29    }
30
31    roomUsers[roomId].users.push(socket.id);
32
33    io.to(roomId).emit(
34      "room-joined",
35      roomUsers[roomId].users.length,
36      roomUsers[roomId].textId
37    );
38
39    console.log(`${roomId} өрөөнийхөн ${roomUsers[roomId].users}`);
40  });
41
42  socket.on("disconnect", () => {
43    console.log("User disconnected");
44    Object.keys(roomUsers).forEach((room) => {
45      roomUsers[room].users = roomUsers[room].users.filter(
46        (user) => user !== socket.id
47      );
48      console.log(
49        `Users in room ${room} after disconnect:`,
50        roomUsers[room].users
51      );
52
53      if (roomUsers[room].users.length === 0) {
54        delete roomUsers[room];
55      }
56    });
57  });
58 });
59
```

```
60 | const PORT = 3030;  
61 | server.listen(PORT, () => {  
62 |   console.log(`Socket.IO server running on http://localhost:${PORT}`);  
63 | });
```