Pealkiri: Automaatne õpikaartide (*flashcard*) genereerimine ja nende abil õppimine

Rühma liikmed: Johan Kirikal, Daglas Aitsen

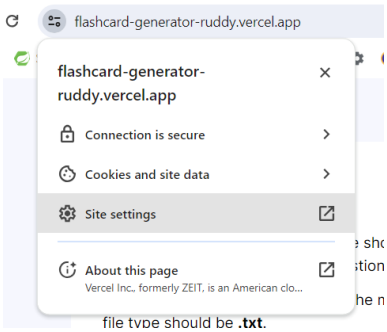
Valminud veebilehe link: https://flashcard-generator-ruddy.vercel.app/

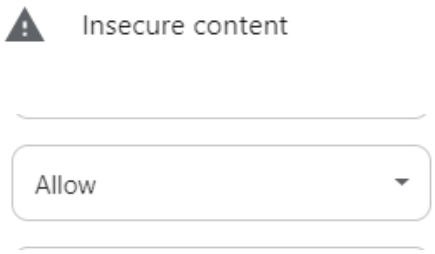
Projekti lähtekood: https://github.com/jkirikal/Flashcard_generator

NB! Selleks, et rakenduse õpikaartide genereerimine veebibrauseris töötaks, võib olla vaja muuta veebibrauseri seadeid ning lubada HTTPS veebilehelt HTTP päringute tegemine. Google Chrome'i puhul tuleb näiteks:

Üleval vasakul „*View site information*" -> „*Site Settings*" -> „*Insecure content*" -> „*Allow*"

Võib-olla on peale seadete muutmist tarvis veebilehele ka värskendus teha





Põhjus, miks selline brauseri seadete muutmine vajalik on, on kuna uuemad veebibrauserid ei luba HTTPS lehtedelt teha HTTP päringuid. Meie rakenduse backend on majutatud kasutatud AWS'i (*Amazon Web Services*) tasuta kättesaadavaid võimalusi, mis kahjuks ei võimalda HTTPS päringuid ilma lisakulutusi tegemata.

Töö eesmärk ja tehisintellekti kasutamise vajadus:

Töö eesmärgiks on luua ja majutada veebirakendus, mis pakub kasutajale võimalust genereerida eestikeelse õppematerjali ja kordamisküsimuste abil automaatselt õpikaarte koos küsimuste vastustega. Lisaks pakub programm võimalust küsimusi ja vastuseid kuvada ning nende vahel liikuda. Iga küsimuse ja vastuse juures on samuti nupp, mis võimaldab vastavat teksti helina kuulata.

Õpikaart kujutab endast justkui digitaalset paberit, kus ühel pool on kirjas küsimus ja teisel vastus. Õpikaarte on hea kasutada õppimiseks, lugedes kõigepealt küsimuse, siis proovides ise vastata ja lõpuks vaadata tegelikku vastust.

Tehisintellekti kasutamise vajadus on ilmne õpikaartide ja vastuste genereerimisel. Manuaalselt küsimustele vastamine võtab õpilase jaoks kauem aega ning on vaevarikkam. Kasutades küsimuste ja vastuste loomiseks tehisintellekti, saame seda protsessi kiirendada ning

mugavamaks muuta, genereerides küsimustele vastused ning pannes nad õigesse formaati tehisintellekti abil. Teine koht, kus meie programmis on tehisintellekti kasutamise vajadus, on genereeritud küsimuste ja vastuste helina ette mängimisel. Kuna küsimused ja vastused, mis tehisintellekti abil genereeritakse on üldjuhul erinevad, siis ei ole võimalik iga võimaliku küsimuse ja vastuse jaoks eraldi helifaili ette salvestada. Tehes õppimist kasutaja jaoks mugavamaks, genereerime vastava helifaili kasutades tehisintellekti. Iga küsimuse juures on nupp, mis võimaldab seda küsimust või siis vastavat vastust helina kuulata.

Kasutatud ideed (nii kursuse materjalidest kui mujalt) koos viidetega

Veebirakenduse arendamisel kasutasime ideid ja lähenemisi, mis on pärit meie kursuse materjalidest. Küsimuste ja vastuste kõnesünteesiks kasutame Neurokõne API-t (https://neurokone.ee/api) , mis oli kasutusel ka tehisintellekti aine 11. praktikumis. Google Gemini API kasutamise ja *prompt engineering*-u kohta õppisin leheküljelt https://ai.google.dev/gemini-api. Projekti valmimisele aitas ka kaasa keelemudel ChatGPT (https://chatgpt.com).

Kasutatud platvormid ja tehnoloogiad: Java, Spring Boot, Google Gemini, ChatGPT, AWS, Vercel, TypeScript, JavaScript, CSS

Iga autori enda panuse kirjeldus

1. **Johan Kirikal:**

Mina kirjutasin rakenduse backendi kasutades Java Spring Booti. See hõlmasi API ja muu loogika ehitamist. Seadsin üles võimalused Google Gemini API-ga suhtlemiseks ning moodustasin struktureeritud "prompt"-i õpikaartide genereerimiseks. Seejärel ühendasin Gemini API meie programmi backendiga. Lõpuks kasutasin platvorme nagu Vercel ja Amazon Web Services, et nii front- kui ka backendi serverites majutada.

2. **Daglas Aitsen:**

Mina keskendusin projekti juures kliendipoolse rakenduse loomisele. Veebilehe ehitasin kasutades React raamistikku, millega mul juba varasemalt oli kogemust. Huvitavamad asjad, mille kallal ma töötasin: kasutaja saab üles laadida ainult kindlat tüüpi faile, millega tehakse POST request backendi; backendist saadud küsimuste ja vastuste kuvamine; Neurokõne API-le kõnesünteesi päringute tegemine ja vastusena saadud helifailide mängimine veebibrauseris; veebilehe üleüldine kujundus. Kujundusega proovisin jääda minimalistlikuks, kuid siiski teha kasutajale lihtsasti arusaadavaks, mis funktsionaalsus on tema jaoks kättesaadav.

Programmi testimisvõimalused ja vähemalt mõned testimistulemused

Programmi on võimalik testida kui avada veebileht brauseris. Õpikaartide genereerimiseks peab brauseri seadeid natuke muutma, nagu üleval pool on kirjeldatud**. Kui brauseri seadeid muuta ei soovi**, siis võib õpikaartide kuvamiseks kasutada nuppu "Try Demo", mis kuvab juba varasemalt rakenduse poolt genereeritud küsimuste vastused. Niimoodi on võimalik katsetada, kuidas rakendus töötab. Demo puhul siiski töötab nii nagu peab küsimuste ja vastuste audiofailide genereerimine.

Kui brauseri seaded on muudetud, siis võib proovida lasta rakendusel genereerida õpikaarte. Selle jaoks oleme projekti GitHub-is pannud kausta nimega "test_materials" kaks faili. Üks on teema materjalide .pdf fail ja teine on kordamisküsimustega .txt fail. Küsimused on samad, mis demo puhul.

NB! Õpikaartide genereerimine võib võtta natuke aega. Kui õpikaardid on genereeritud, siis avaneb esimene õpikaart automaatselt.

Töö käigu kirjeldus:

Töö kulges suures pildis sujuvalt. Kui olime idee välja mõelnud, siis panime paika plaani: Johan keskendub backend koodi kirjutamisele ja Daglas tegeleb frontendi arendamisega. Kui olime individuaalselt saanud oma tööga valmis, siis ühendasime ja panime need suhtlema kasutades API endpointe.

Algselt oli plaanis programmile anda lisafunktsionaalsust. Näiteks soovisime, et kasutajal oleks võimalik genereeritud küsimusi ja vastuseid failina salvestada, et neid järgmisel korral uuesti kasutada. See oleks lahendanud probleemi, et kasutaja peab iga kord, kui ta soovib õpikaartidelt õppida, küsimustele vastused uuesti genereerima. Kahjuks see funktsionaalsus jäi programmi lisamata, sest kui olime projektiga alustanud, siis mõistsime, et töö läheks mahukamaks kui me esialgu plaanisime. Otsustasime keskenduda programmi põhifunktsionaalsusele ning üritasime planeerida arendust nii, et meie mõlema kohta oleks töömaht umbes 11 tundi.

Üks probleem, mis projekti käigus esile kerkis oli juba eelpool mainitud mure rakenduse majutamisega. Meie jaoks oli tähtis, et saaksime rakenduse veebi avalikustatud, sest niimoodi on seda kõige lihtsam teistel huvilistel proovida. Samuti tahtsime lehe majutada tasuta. Vercel pakkus meile tasuta HTTPS frontendi majutust ning AWS "free tier" pakkus tasuta backendi majutust. Murekohaks aga oli, et modernsed veebibrauserid ei luba HTTPS lehtedelt HTTP päringuid teha ilma kui kasutada seda brauseri seadetest manuaalselt ei muuda. See probleem on midagi, millega me algselt ei arvestanud, kuid ka mida me tulevikus tahaks kindlasti parandada. Probleemi lahendaks veebilehele domeeni ostmine.

Kindlasti on meil soov programmile funktsionaalsust tulevikus lisada ning arendust jätkata. Ideid, kuhu suunas saaks edasi minna, on palju. Lisasime järgmisele lehele ka ühe kuvatõmmise veebirakendusest.

Choose new files

Question 1:

Miks on rakujagunemine eluoluline?

1/34

Questions and Answers:

Hide

1. Miks on rakujagunemine eluoluline?	Rakkude jagunemine on organismile eluküsimus, kuna see tagab paljunemise, kasvu ja kudede uuenemise.
2. Mis võib juhtuda, kui raku jagunemise etapid ei kulge korrektselt?	Vigade korral jagunemisprotsessis tekivad vigased kromosoomid, mis võivad rakule surma või anomaaliaid põhjustada, näiteks kontrollimatu pooldumine vähkkasvaja puhul.
3. Mis üldiselt toimub interfaasis?	Interfaasis toimub kõigi rakukomponentide sünteesimine, et tagada tekkivale tütarrakule kõik vajalik uue rakutsükli alustamiseks.
4. Milleks on rakutsüklis G1 ja G2 faas olulised?	G1 ja G2 faas annavad rakule vajaliku aja kasvamiseks ja muude rakukomponentide kordistamiseks.