ENDLESS STORY

PROGRAMMATŪRAS IZSTRĀDE

Ievads

1.1 Mērķis

Projekta mērķis ir integrēt publiskos teksta ģenerēšanas modeļus un nodrošināt mijiedarbību ar lietotāju. Projekts piedāvā iespēju sazināties ar modeli dažādu sižetu ietvaros. Lietotājs var izvēlēties scenāriju un virzīties pa to, sarunājoties ar stāstītāju, kurš bezgalīgi ģenerē stāstu.

1.2 Darbības joma

Projekts nodrošina:

- Iespēju izvēlēties vienu no pieciem dažādiem stāstu scenārijiem.
- Saziņu ar stāstītāju, kurš ģenerē bezgalīgu un unikālu stāstu, pamatojoties uz lietotāja izvēlēm.
- Lietotāju kontu un kredītu sistēmu, kas ļauj sekot progresam un ierobežot mijiedarbību ar stāstītāju.

Projekts izmanto:

- Integrāciju ar bezmaksas teksta ģenerēšanas modeļiem no GitHub (OpenAI), lai nodrošinātu dinamisku un personalizētu stāstu veidošanu.

1.3 Mērķauditorija

Projekts ir paredzēts visu vecumu lietotājiem, sākot no 6 gadu vecuma, kuri interesējas par daiļliteratūru un mijiedarbību ar teksta ģenerēšanas modeļiem. Tas ir īpaši piemērots tiem, kurus aizrauj tādi žanri kā fantāzija, zinātniskā fantastika, postapokalipse un citi līdzīgi sižeti. Platforma piedāvā radošu un izglītojošu pieredzi, kas piemērota gan bērniem, gan pieaugušajiem.

Sistēmas pārskats

2.1 Arhitektūra

Projekts ir veidots pēc klienta-servera arhitektūras principa, nodrošinot lietotāja saskarni tīmekļa pārlūkprogrammā un servera pusi, kas apstrādā pieprasījumus.

- **Klienta puse**: HTML, CSS, JS. Lietotāji var reģistrēties, autentificēties un izvēlēties stāstu scenārijus.
- **Servera puse**: Python serveris, kas apstrādā HTTP pieprasījumus, autentificē lietotājus, vada sesijas un mijiedarbojas ar API.
- **Datu bāze**: MySQL (lietotājvārds, parole, kredīti).
- **Teksta ģenerēšana**: OpenAI API.

2.2 Izmantotās tehnoloģijas

- **Frontend**:
  - HTML, CSS
  - JavaScript
- **Backend**:
  - **Python** ar **Flask**
  - **PyMySQL**
  - **bcrypt** paroļu šifrēšanai
  - **OpenAI API** teksta ģenerēšanai

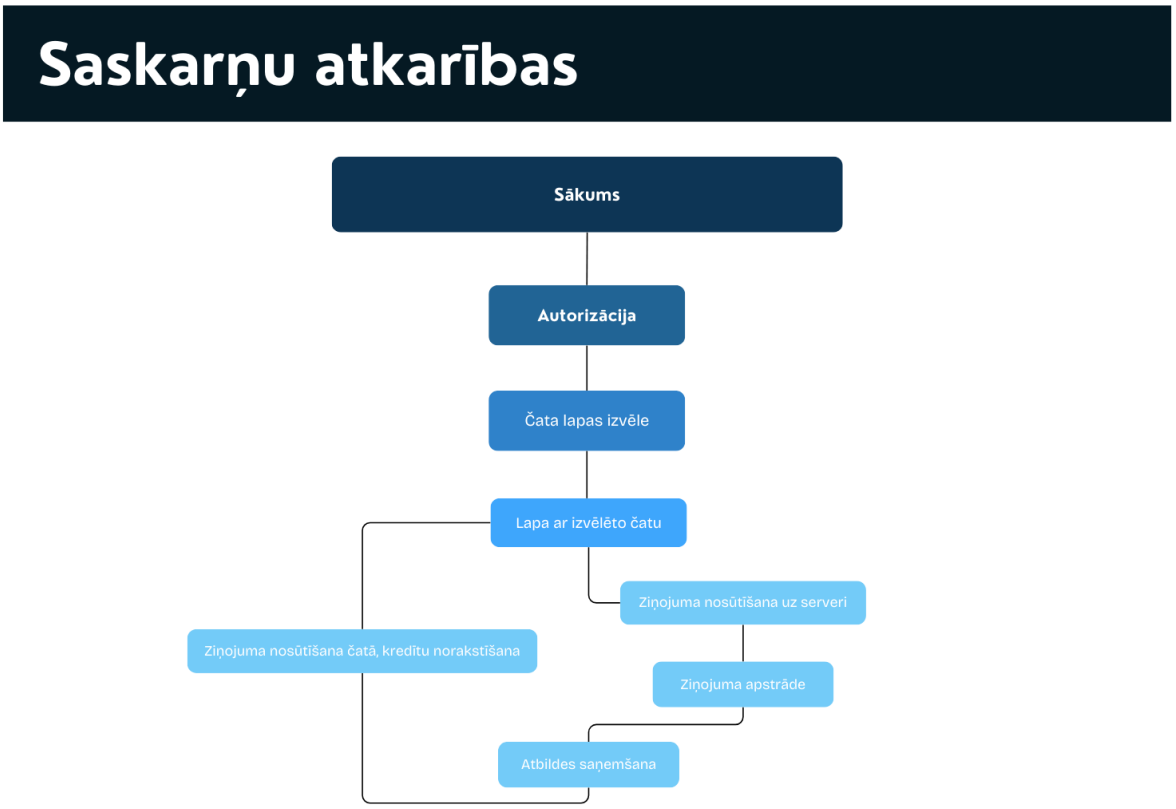
2.3 Atkarības

Lai nodrošinātu sistēmas pilnvērtīgu darbību, nepieciešamas šādas atkarības:

- **Python bibliotēkas**:
  - Flask
  - PyMySQL
  - bcrypt
  - openai
- **Servera prasības**:
  - Python
  - MySQL

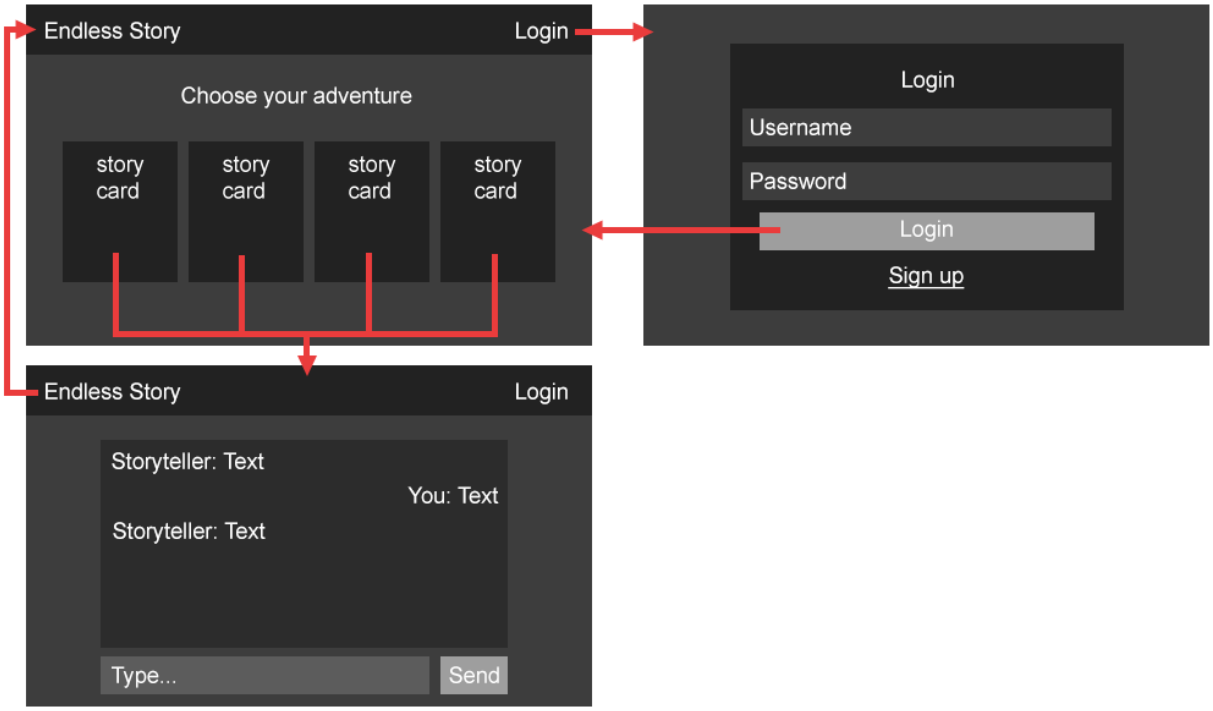
Projektēšana

3.1 Saskarņu atkarības

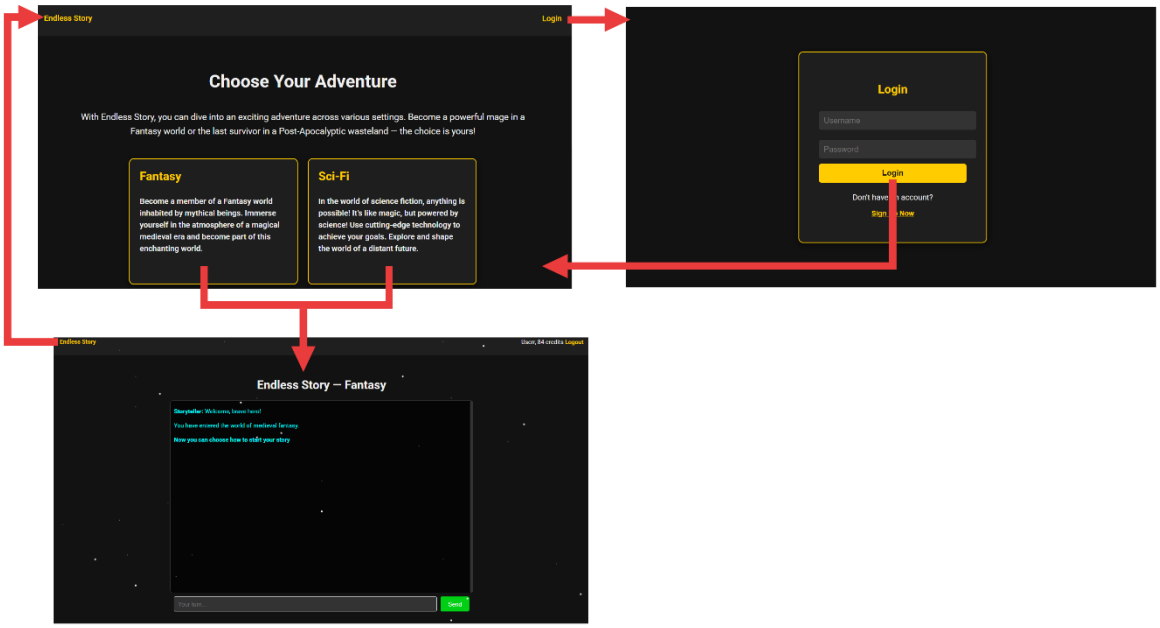


Dizaina izstrāde

4.1 Saskarnes karkasa diagramma



4.2 Makets



Testēšana

5.1 Testēšanas plans

Testēšanas galvenais uzdevums ir identificēt visas iespējamās programmas darbības kļūdas visos tās posmos, kā arī nodrošināt to turpmāku novēršanu.

Nepieciešams pārbaudīt:

- reģistrāciju,
- autorizāciju,
- pareizu savienojumu ar serveri,
- pareizu datu bāzes darbību,
- čata darbību,
- izmantoto API.

5.2 Testa gadījumi

1. ES_K1_T1:

Scenārijs: Klients atver reģistrācijas logu un to aizpilda.

Sasniedzamais rezultāts: Lietotāja dati tiek saglabāti datu bāzē, un lietotājs tiek novirzīts atpakaļ uz sākumlapu.

2. ES_K1_T2:

Scenārijs: Klients izvēlas sev nepieciešamo scenāriju.

Sasniedzamais rezultāts: Lietotājs tiek novirzīts uz viņa izvēlētā čata lapu.

3. ES_K1_T3:

Scenārijs: Lietotājs nosūta ziņojumu čatā.

Sasniedzamais rezultāts: Tiek pārbaudīts lietotāja kredītu atlikums, pēc tam pieprasījums tiek apstrādāts caur API, un lietotājam tiek atgriezta atbilde.

5.3 Testa rezultāti

Tika atklātas šādas kļūdas:

- Reģistrējoties, ievadot paroli, izmantojot jauno paroļu kešēšanas sistēmu, datubāzes rindā nepietika vietas paroļu hasham, kā rezultātā parole tika saglabāta nepareizi.
- Sūtot ziņojumu čatā, varēja notikt kļūme atbildē no API.
- Bija iespējams izmantot programmu, nenokārtojot autorizāciju.

Pēc tam visas kļūdas tika novērstas.