Tweede opgave van Programming Techniques: Webapplicatie

Jens van der Sloot S4018494
Arthur van der Sterren S4097769

10 april 2024

1 Ontwerp

1.1 Eisen

Frontend:
1. Landing met mogelijkheid tot registreren en inloggen
2. Lijst van alle chats
3. Lijst met (bevriende) gebruikers
4. Chatgeschiedenis met geselecteerde gebruiker
5. Pagina met account instellingen (Publieke / prive gebruiker)
6. Uitlog optie bij instellingen
7. Optie iemand toevoegen aan de chat: maak een groepchat aan
8. Chatbox om berichten te typen en sturen
9. Het kunnen zien van een lijst met publieke gebruikers
10. Chat favourite kunnen maken
11. Duidelijke representatie deelnemers van groepchats
12. Duidelijke representatie van errors als er een probleem is (E.g. huidige gebruiker probeert een onbekende gebruiker te bevrienden)
13. Alle functionaliteit en pagina's zijn beschikbaar op elk device; de website is ofwel responsive

Backend
1. Controle inlogscherm m.b.v. hash
2. Controle nieuw account (orgineel email, gebruikersnaam en tweemaal hetzelfde wachtwoord herhalen, max x karakters en meer)
3. Aanmaak nieuw account
4. Maak een chat aan gegeven ingelogde gebruiker en meegegeven gebruiker
4. Haalt chatberichten op uit Database (Van wie, wanneer, inhoud)
5. Haalt gebruikers op uit database (Bij publieke gebruikerlijst)
6. Mogelijkheid tot favoriet maken en verwijderen van een chat
7. Verstuurt berichten

Database
1. Accountdata (naam, accountID, email, privacy, favourite_chats, send friend requests, en password (hash) (ID is primary key))
2. Vrienden bijhouden en vriendverzoeken (via ID)
3. Chats bijhouden (ID)
4. Opslaan Berichten (ID binnen chat wat MSG, TIME en USER bevat)

1.2 Data

1.2.1 User

username: Gebruikersnaam. (Unique, Indexed)
id: ID van de gebruiker. (Primary Key)
email: Email van de gebruiker. (Unique, Indexed)
password_hash: Hash van het wachtwoord van de gebruiker. (Optional)
privacy: Privacy setting van de gebruiker. (Default: 'private')
friends: Vrienden van de gebruiker. (Property)
is_active: Status of gebruiker actief is of niet. (Default: 1)
last_checked: Timestamp van wanneer de gebruiker het laats actief was. (Default: Current time (UTC))

1.2.2 Post

id: Unieke identificatie voor een post. (Primary key)
body: De inhoud van de post (maximaal 140 tekens).
timestamp: Timestamp van de post (Default: Current time (UTC)).
user_id: Verwijzing naar het ID van de gebruiker die de post heeft gemaakt (index).
author: Dynamische relatie met de User-klasse via de posts-attribuut. Hiermee kunnen we de auteur van een post opvragen.

1.2.3 Chat

id: Unieke identificatie voor een chat. (Primary key)
users: Gebruikers die deelnemen aan de chat. (Many-to-many relatie met 'User')
messages: Berichten in de chat. (Relatie met 'Message')

1.2.4 Message

id: Unieke identificatie voor een bericht. (Primary key)
text: De tekst van het bericht (maximaal 500 tekens).
timestamp: Tijdstempel van het bericht (Default: Current time (UTC)).
user_id: Verwijzing naar de gebruiker die het bericht heeft verzonden (index).
chat_id: Verwijzing naar de chat waarin het bericht is verzonden (index).

1.2.5 Friendrequest

from_user: Verwijst naar de gebruiker die het vriendschapsverzoek heeft verzonden (relatie met 'User').
to_user: Verwijst naar de organisatie die het vriendschapsverzoek heeft ontvangen (relatie met 'Organisation').
status: Laat zien of de friendship is geaccepteerd

accept(): Methode om het vriendschapsverzoek te accepteren.

1.2.6 Friendship

user_id: Verwijst naar de gebruiker die vriendschap is aangegaan (relatie met 'User').
from_user: Verwijst naar de gebruiker met wie de vriendschap is aangegaan (relatie met 'User').
timestamp: Timestamp van de vriendschap (Default: Current time (UTC)).

1.3 API front-back

de API communiceert de functies van de frontend naar de backend. Zo communiceert hij de inputs van het log-in scherm, vrienden functies, group aanmaken maar ook het chatten zelf met de back-end. Voor diepere informatie zie sectie 3; Documentatie API

2 Taakverdeling

Jens:Opzet Flask, front end, back end
Arthur:Ontwerp + bedenken functies, API, verslag

3 Documentatie API

In dit project heeft de API veel functies, hierbij een documentatie van de gebruikte functies.

3.1 GET

/index
Returnt de index pagina. (als gebruiker is ingelogt)

/get_friends
Geeft een lijst van vrienden die zijn ingelogt. (als gebruiker is ingelogt)

/get_friend_requests
Weergeeft de lijst met inkomende friendrequests van gebruiker. (als gebruiker is ingelogt)

/get_user_chats/<usernames>
Laat de berichten zien tussen de huidige gebruiker en de geselecteerde 2de persoon (aangeduid met ¡usernames¿ als username,username, waarin 1ste username = gebruiker en 2de username = 2de persoon in de chat). (als gebruiker is ingelogt.

/get_user_privacy
Geeft de huidige privacy status van de ingelogde gebruiker

/get_public_users
Geeft de lijst met gebruikers die als hun account op public hebben staan

/get_chats
Geeft de lijst van alle chats met ingelogde gebruiker.

/get_chat/<usernames>
Returnt de chat met aangeduide gebruikers.

3.2 POST

/send_friend_request/<username>
Stuurt friend request naar gespecificeerde user. (als gebruiker is ingelogt)

/accept_friend_request/<username>
Accepteert request van gespecificeerde user. (als gebruiker is ingelogt)

/favourite_chat/<usernames>
Boegt chat die gespecificeert wordt door ¡usernames¿ toe aan de lijst met gefavorite chats. (als gebruiker is ingelogt)

/unfavourite_chat/<usernames>
Verwijdert gespecificeerde chat van de lijst met favorieten. (als gebruiker is ingelogt)

/send_message
Stuurt een bericht van huidige gebruiker naar een andere gebruiker. (als gebruiker is ingelogt)

/create_chat/<usernames>
Maakt nieuwe chat aan tussen gespecificeerde users. (als gebruiker is ingelogt)

/set_user_to_public
Zet de privacy van de ingelogde gebruiker naar Public

/set_user_to_private
Zet de privacy van de ingelogde gebruiker naar Private

/chat_add_user
Voegt aangeduide gebruiker too aan bestaande chat

3.3 DELETE

/delete_friend_request/<username>
Verwijdert inkomend friend request van gespecificeerde user. (als gebruiker is ingelogt).

/remove_friend/<usernames>
Verwijdert aangeduide vriend van vrienden lijst. (als gebruiker is ingelogt)

/delete_chat/<usernames>
Verwijdert de chatgroep met aangeduide gebruikers (aangegeven als <gebruiker>,<gebruiker>,<gebruiker>, etc) (indien gebruiker is ingelogt)

3.4 Andere

/logout
Logt de gebruiker uit, gebruikt geen methodes.

/
Gebruikt zowel GET als POST. Met post wordt opgehaalt of de gebruiker is geregistreert, en met POST wordt het ingevulde inlogformulier opgeslagen.

4 Hoe zet je het programma op?

Als je het programma wil runnen zijn er vier stappen:

Stap 1: Clone ofwel kopieer de repository
De repository is nu natuurlijk wel private, dus gebruik de gestuurde versie

```
1     git clone https://git.liacs.nl/s4018494/pt2.git
```

Stap 2: Navigeer naar de gekopieerde folder toe
Stap 3: Installeer de benodigde dependancies voor het project met behulp van de setup.py

```
1     python setup.py install
```

Stap 4: Voer de app uit met behulp van

```
1 flask run
```

en navigeer naar localhost:5000 in je browser. Nu heb je een lokale versie van de app actief!

5 discussie

Ten eerste is het niet gelukt om alles wat gewild was werkend te krijgen, we wilden in eerste instantie de mogelijkheid toevoegen dat als je als gebruiker offline ging anderen konden zien dat je status verandert was, ofwel naar offline. Maar naar even proberen kwamen we hier niet helemaal achter, we hadden wel een concept werkend; op de backend werdt al elke 5 seconden gepinged (als data-update; nieuwe berichten of vriendverzoek) dus hadden we een nieuwe functie toegevoegd die keek of deze ping zou stoppen,

en zoja, de status veranderd zou kunnen worden van de gebruiker. Maar dit concept leverde vele rare problemen op: er kwamen allemaal problemen met de database, waardoor uiteindelijk de hele migraties gereset moesten worden, en met de tijdsnood in het achterhoofd kozen we er toch voor om dit toch over te slaan.

Een ander leuk idee zijn custom profielfoto's alleen het probleem met zoiets is dat dit gelijk super onveilig is (of je moet heel veel checks invoeren), in plaats van dat een gebruiker hier een valide link invoert is dit - en eigenlijk elk tekstvak wat verstuurd wordt naar de db - supergevoelig tot een SQL-Inject aanval. Dus hebben we maar gekozen voor een functie die willekeurige de kleuren maakt per chat, zodat er toch nog wel een beetje gevoel tot custimization is.

Hiernaast was ons git gebruik niet optimaal, we hebben niet veel gepushed. Normaal zijn we gewend om om de paar honderd lines te pushen, maar tijdens dit project liep dit uit tot in de duizenden. Dit lag voornamelijk aan het feit dat we in tijdsnood liepen, en dus de prioriteit naar een mogelijkheid tot rollback lager ligt.

Daarnaast vinden we toch wel dat we een aardig leuk product hebben neer gezet in de zeer korte tijd en vonden we het een interessante ervaring om te werken met Flask, een framework waar we beiden nog geen ervaring mee hadden.