If people only talked about what they understood, the world would be very quiet.
[Albert Einstein]

1 Background and Motivation

As of today, Elon Musk has basically ruined Twitter (formerly and still known as 'Twitter'). Twitter has become a 'reliable' source for fake news, conspiracy theories, anti-science, racism, and all kinds of bullshit. Previously every village had a village idiot everybody knew and ignored. And that was ok. Now these village idiots unite and team up on Twitter and other social networks.

Twitter is a safe haven for trolls, unimaginable stupidity, unreflected bullshitting, hate, lies, and racism. To remedy this problem, Musk did not do anything, but in fact, made things much worse by firing the fake news team. He even invited previously banned users back to the platform. The latter was already convicted to pay one billion dollars in fine for spreading fake news. Musks not-handling of fake news also triggered a couple of competing platforms like BlueSky and Mastodon which however struggle with attracting enough users to make those platforms attractive.

2 General Idea: Fame Profiles

In this project, we will explore how to get a social network right from the get-go. The main idea of this project is inspired by Einstein's famous quote and the novel "Fameland" by Tom J. Petersson. In that novel, Petersson sketches a society in-between a democracy and an epistocracy, i.e., where the ones with knowledge have a bigger say in discussions and decisions.

In Fameland, everything you say, everything you decide, and everything you do is mediated by your skill profile (coined: *fame* or fame profile), with the help of the right software.

2.1 Fame: Basic Idea

For everybody living in Fameland, the society keeps a *skill profile* (in the book it is called *fame*). That profile records and tracks your skills. In contrast to social profiling (we learned about this when discussing the book NSA by Eschbach), this is **not** about your preferences, opinions, political orientation, sexual orientation, the websites you surfed, the books you read, the music you heard, the movies you watched, the stuff you bought, etc. **A fame profile is solely about your skills.**

For instance, if you are good in Computer Science, your profile will have an entry for that:

Computer Science: Knowledgeable

Upon completing a B.Sc. degree in computer science or a related field, that entry may be changed to:

Computer Science: Pro

In addition, the fame profile also tracks subtopics and specializations, also pointing to their parent topics, e.g.

Databases: Jedi $\rightarrow$ Computer Science

Django: Pro $\rightarrow$ Databases $\rightarrow$ Computer Science

Like that, for any given user, a large taxonomy of skills is kept and maintained. The more you learn, the more skills you acquire, the longer your fame profile will become. The profile can be specialized arbitrarily. However, for the sake of keeping things simple, we will use very coarse-granular skills in this lecture project.

2.2 Negative Fame

The fame profile does not only keep positive skills but also *negative* skills, i.e., if for any reason you did something, said something or distributed something that was counterfactual, this may result in *negative fame*. For instance, assume you posted something claiming that the earth is flat. Then this will affect your fame in earth topology negatively[1]:

Earth Topology: Bullshitter → Geology → Natural Sciences → Science

2.3 Professional Education

Notice, that special cases of fame profiles are already used in today's societies. For instance, any kind of professional qualification, school degrees, university degrees, driving licenses, sports qualifications, and any other 'certificate' you may ever obtain are just special cases of fame. For instance, Japanese martial arts like Judo and Karate express fame through differently colored belts. In soccer, basketball, handball the fame of a team is implicitly expressed by the league it plays (or the sum other teams pay for a given player), etc. In fact, any kind of exam, any test you complete may be used to adjust your fame accordingly.

2.4 Today's Ratings vs Fame

Moreover, any kind of customer experience may be considered in your fame profile. For instance, consider two painters Alice and Bob. Initially after completing their professional education, both Alice and Bob may have the same fame entry in their profiles:

Painting: Pro → Craft

However, over time it may turn out that Alice does a much better job, i.e., working more diligently, getting better and better at her job. This will affect her painting fame:

Alice: Painting: Super Pro → Craft

In contrast, for Bob, it may turn out that the he botches too often and makes many customers unhappy. This will affect his fame in painting.

Bob: Painting: Botcher → Craft

This implies, that even though Bob has a professional education as a painter, due to doing a bad job, his fame may be lowered below Pro. This also implies that the separation between formal professional qualifications and people learning the same skills through other means is removed. If you have the same skills as someone who studied that topic at a university, why should there be a distinction?

Today, these kind of quality ratings for painters and other crafts would be tracked through webpages (e.g. Google Ratings) where customers may leave a review which could then be used by others to understand which painter is good and which is not (of course: the existing rating systems do have many problems like: how representative are those ratings anyway?). The fame profile also supersedes these rating portals. Like that it becomes crystal clear who does a bad and who does a good job.

And this just scratches the surface of the power of fame and its implications.

2.5 Fairness, Manipulations, Abuse

When implementing fame for a society, it is important to make sure that the fame profiles are fair, do not get manipulated or abused in other ways (very similar to existing rating portals). Who has control over the profiles? How can we avoid that profiles get abused? Interestingly, all of these problems can be fixed (some of the things are alluded to in the book), but this is beyond the scope of this project (and this lecture).

[1]Interestingly, if you claimed that earth was a potato, you would actually collect positive fame.

3 Project: Integrate Fame with a Social Network

3.1 State of the Project

In this project, we will assume that you are a software engineer at a social network. Your bosses decided to integrate fame into that social network. Your job is to evaluate how that can be achieved by building a POC (proof of concept). Your task is to start from an existing Django app which implements typical parts of a social network application (in the project this is the Django app `socialnetwork`). Existing functionality of this Django app includes:

1. rather complete data models (`models.py`)
2. an internal API (`api.py`) to have a central place to specify functionality independent of html and REST views
3. simple html views (wrapping DRF endpoints), including initial styling
4. timelines and search
5. simple textual posts (attachments not supported yet)
6. users may follow other users
7. user authentication
8. fake data generation through the ORM
9. a basic set of unit tests

You are also provided with a Django app organizing the fame profiles of users (in the project this is the Django app `fame`).

Existing functionality includes:

1. rather complete data models (`models.py`)
2. simple html views (wrapping DRF endpoints), including initial styling
3. fake data creation through the ORM and through views
4. a basic set of unit tests

Both apps were already combined into a common Django project (`famesocialnetwork`). Existing functionality includes:

1. fame profile of users can be reached directly from the timeline
2. posts containing bullshit will not be published (but still be recorded in the database)
3. a mockup "magic AI" algorithm (`socialnetwork/magic_AI.py`) detects the expertise areas for each post to be published (i.e., it randomly generates expertise areas for each post); as with a real topic detection algorithm, sometimes this algorithm cannot determine the truth rating

Also checkout the `README.md` for information on how to install and run the project.

3.2 Your tasks

Your job in this project will be to add some of the missing functionality in three different levels of difficulty:

3.2.1 Difficulty Level 1: Add missing implementations in the API

T1 Change `api.submit_post` to not publish posts that have an expertise area which is contained in the user's fame profile and marked negative there (independent of any truth rating determined by the magic AI for this post).

T2 Change `api.submit_post` to adjust the fame profile of users if they submit a post with a negative truth rating, but only for the expertise area found for the post that has a negative truth rating:

T2a If the expertise area is already contained in the user's fame profile (with any fame level), lower the fame to the next possible level.

T2b If the expertise area is not contained, simply add an entry in the user's fame profile with fame level "Confuser" (hint: negative fame and take a look at `famesocialnetwork/fakedata.py`).

T2c If you cannot lower the existing fame level for that expertise area any further, ban the user from the social network by setting the field `is_active` in model `FameUsers` to `False` disallowing him/her to ever login again, logging out the user if he or she is logged in, and unpublishing all his/her posts (without deleting them from the database).

T3 Implement `api.bullshitters`: It should return a Python dictionary mapping each existing expertise area in the fame profiles to a list of the users having negative fame for that expertise area. Each list should contain Python dictionaries as entries with keys "user" (for the user) and "fame_level_numeric" (for the corresponding fame value), and should be ranked, i.e., users with the lowest fame are shown first, in case there is a tie, within that tie sort by `date_joined` (most recent first). Note that expertise areas with no bullshitters may be omitted.

T4 Implement *automatic communities* that are open for users with a fame level of Super Pro or above in the respective expertise area. Once a user acquires fame level Super Pro or above in some expertise area, the user has the option to join the corresponding community. In the timeline, users can then switch between the *standard mode*, in which they can see all posts, and the *community mode*, in which they can only see posts from the communities they joined. If the fame level of a community member drops below Super Pro due to publishing posts with bad truth ratings in this expertise area, the member should automatically be removed from the community. Specifically, implement the following steps:

- Adapt the data model of `SocialNetworkUsers` to keep track of a user's communities. Remember to run `recreate_models_and_data.sh` to perform migrations afterward.
- Implement `api.join_community` and `api.leave_community`, which add a given user to or remove him/her from a given community.
- Change `api.timeline` to return all posts to be displayed when in community mode (see documentation in `api.timeline`).
- Change `api.submit_post` to automatically remove a user from a community if the fame level for the expertise area of this community drops below Super Pro.

T5 Implement `api.similar_users`: It should return for a given user u_i the list of similar users as `QuerySet` of `FameUsers`. This list should only contain other users with a non-zero similarity score and should be in descending order according to their similarity score, in case there is a tie, within that tie sort by `date_joined` (most recent first). The similarity score $s(u_i, u_j)$ is defined as follows:

$$s(u_i, u_j) = \frac{1}{|E_i|} \sum_{e \in E_i} \sigma(|f(u_i, e) - f(u_j, e)| \leq 100)$$

Here, E_i denotes the set of expertise areas of user u_i and $f(u_i, e)$ denotes the numerical fame value of user u_i in expertise area e. Note that $f(u_i, e)$ implicitly returns ∞ if user u_i has no fame value in expertise area e. The indicator function $\sigma(p)$ returns 1 if the predicate p is true and 0 otherwise. In other words, the similarity score of user u_i with user u_j is the fraction of expertise areas of u_i where u_i's fame level differs from u_j's fame level by a numerical value of at most 100, i.e., where the users have similar fame. Note that this similarity score is not symmetric, i.e., $s(u_i, u_j)$ and $s(u_j, u_i)$ may differ.

Note that you can test whether you implemented these tasks correctly with unit/end-to-end tests provided with the project (see `famesocialnetwork/tests.py/StudentTasksTests`). We recommend to start by disabling all failing tests (e.g. by prefixing the `test...`-methods with a dash: `_test...`. And then enable each test one by one (and fix the implementation in that process). Make sure to not forget any tests in your implementation.

Take care to not overfit your implementations: your implementations must not make assumptions about the data provided (i.e., such that the tests only pass with the given dataset). If you overfit your implementations to the tests, you will not receive points for this part of the project.

3.2.2 Difficulty Level 2: Add additional views

Hint: Stubs for the views are already created in `socialnetwork/views/html.py`.

T6 Implement an html view in the socialnetwork app showing the results of `api.bullshitters`.

T7 Implement the views `join_community`, `leave_community`, and `toggle_community_mode` in the socialnetwork app, allowing users to join and leave communities, and switch between standard mode and community mode in the timeline, respectively. Further, update the `timeline` view and the template `timeline.html` such that users can switch between standard mode and community mode by pressing a button. Additionally, users should see an overview of the communities they are a member of and the communities they may join according to their fame level. Make sure that users can only leave communities they are a member of and join communities they are eligible for by pressing a button. Also, make sure the timeline is updated whenever the user joins or leaves a community or switches between standard mode and community mode.

T8 Implement an html view in the socialnetwork app showing the results of `api.similar_users`.

3.2.3 Difficulty Level 3: Suggest meaningful/creative extensions of the concept and free style (bonus points).

T9 Suggest extensions of the concept. Discuss/criticize its impact on a social network (not more than one A4 page).

T10 Implement some of your ideas.

Hand-In

For difficulty levels 1, 2, and 3, you have to hand-in your entire project as a zip-file via CMS (one upload for all of them). For difficulty level 3, add an additional text file `extensions.txt` in the root directory of the zip-file.

Note that for **Difficulty-Level 1**, we will execute all tests on a fresh project where we simply replace the files

1. `socialnetwork/api.py`
2. `socialnetwork/models.py`

with your versions of those files. Make sure to test your version of those files against a fresh project before handing in.

Solutions must be submitted in teams of 3 to 4 students by June 29th, 2025, 23:59, via your personal status page in CMS using the Team Groupings functionality. Late submissions will not be graded!

Passing Criteria

33% fraction of unit/end-to-end tests passed.

67% code review in front of a tutor. This code review is mandatory and team members who are not able to explain parts of the code do not get any points for the project. Tutors will propose projects for a final round which will be presented in front of Prof. Dittrich. The three best projects will be praised and briefly shown in the lecture.

Note on Plagiarism

As with other hand-ins, you are not allowed to copy code from other groups. We will automatically cross-check solutions for plagiarism. Plagiarized solutions will receive 0 points and hence those teams will not be allowed to participate in the exams.

Note on ChatGPT and other AIs

You are allowed to use ChatGPT and similar tools for the project. According to the university's regulations, no explicit documentation of the use is required. However, remember that all team members must know and understand all code for the code review.