NLPeace: Github Repository

Team members

Name and Student id	GitHub id	Number of story points that member was an **author** on.
Fatima El Fouladi 40108832	seaiam	11
Anum Siddiqui 40129811	AnumSidd	9
Jeff Wilgus 29206345	jeffrey-w	5
David Lemme 40157270	davrine	8
Mira Aji 40041473	miraaji	5
Adam Qamar 40175980	aqa02	5
Shabia Saeed 40154081	shabiasaeed	3
Raya Maria Lahoud 40129965	rayalahoud	8
Nelly Bozorgzad 40289770	nellyb4	6
Joshua-James Nantel-Ouimet 40131733	NanoProd	11

Project summary

NLPeace is a social networking app available both on desktop and mobile. We aim to connect people and foster a safe environment free of hate and offensive content. We are leveraging natural language processing to build a strong language model that will allow for our content moderation to be automatic. Hateful content is thus nipped at the bud. As our network grows, we aim to amass more data to create a stronger language model and ultimately, a safer, more peaceful experience.

Risk

- **Inaccurate Moderation:**
  Harmless content might be inaccurately flagged as hate speech or we might fail to detect actual harmful speech which could lead to user frustration.
  Risk Level: High
  Mitigation Strategy: Regularly update and test the system using diverse examples, including false positives and negatives. Additionally, provide users with the ability to appeal content decisions. We will also add admin users that can perform QA on the moderation.

- **Avoidance Techniques:**
  Users might try to bypass moderation by using coded language or through media to convey harmful speech.
  Risk Level: Medium
  Mitigation Strategy: Update the moderation rules regularly to adapt to new avoidance techniques. Additionally, implement keyword filtering to detect evasive content.

- **User Backlash and Public Relations Issues:**
  Users may perceive the moderation efforts as either too strict or not strict enough, leading to public backlash and negative publicity for the app. Moreover, controversial content moderation decisions could spark outrage on social media.
  Risk Level: Medium
  Mitigation Strategy: Clearly communicate the app's moderation policies and guidelines to users and the public to manage expectations and minimize misunderstandings. Develop a crisis communication plan to manage potential PR issues and controversies effectively.

- **Potential technological obsolescence**
  The AI algorithms and models are quickly evolving; newer, more advanced models and techniques emerge rapidly. This can result in the app's AI

moderation system becoming less effective over time.
Risk Level: High
Mitigation Strategy: Adopt an agile development approach to allow quick iterations and updates. This flexibility will let us integrate newer AI models and techniques as they become available. Our AI models will continuously learn as our user-base grows and as more content is posted, it will be added to our NLP pipeline to train our models.

- **Security Vulnerabilities:**
  Bad coding practices can lead to security vulnerabilities in the code of the application. When this happens attackers or even regular users may be able to gain access to privileged data or functions.
  Risk Level: High
  Mitigation Strategy: All developers should brush up on good coding practices and all code should be peer reviewed.

Legal and Ethical issues

- There might be an ethica and legall issue arising from our collection of user post to build and train our NLP model. We aim to mitigate this by having clear terms of service which detail how we will use user data to make the platform safer and train our model. We will be transparent in our data usage.

- Handling user information such as emails and posts (if their account is private) is exposing us to some legal risks. This risk is explored above.

Velocity

***Project Total*:** 13 stories, 46 points over 4 weeks

Iteration 1 (13 stories, 46 points)

In this iteration, we worked on mainly technical tasks, like setting up the base of our project. This involved setting up the Django application and the developer databases. We added a CI pipeline as well as containerization of the project. Most importantly, we researched hate speech datasets and implemented a NLP model which reached 93% accuracy in detecting hate speech and offensive language.

Iteration 2 (8 stories, 17 points)

TBD

Iteration 3 (11 stories, X points)

TBD

Overall Arch and Class diagram

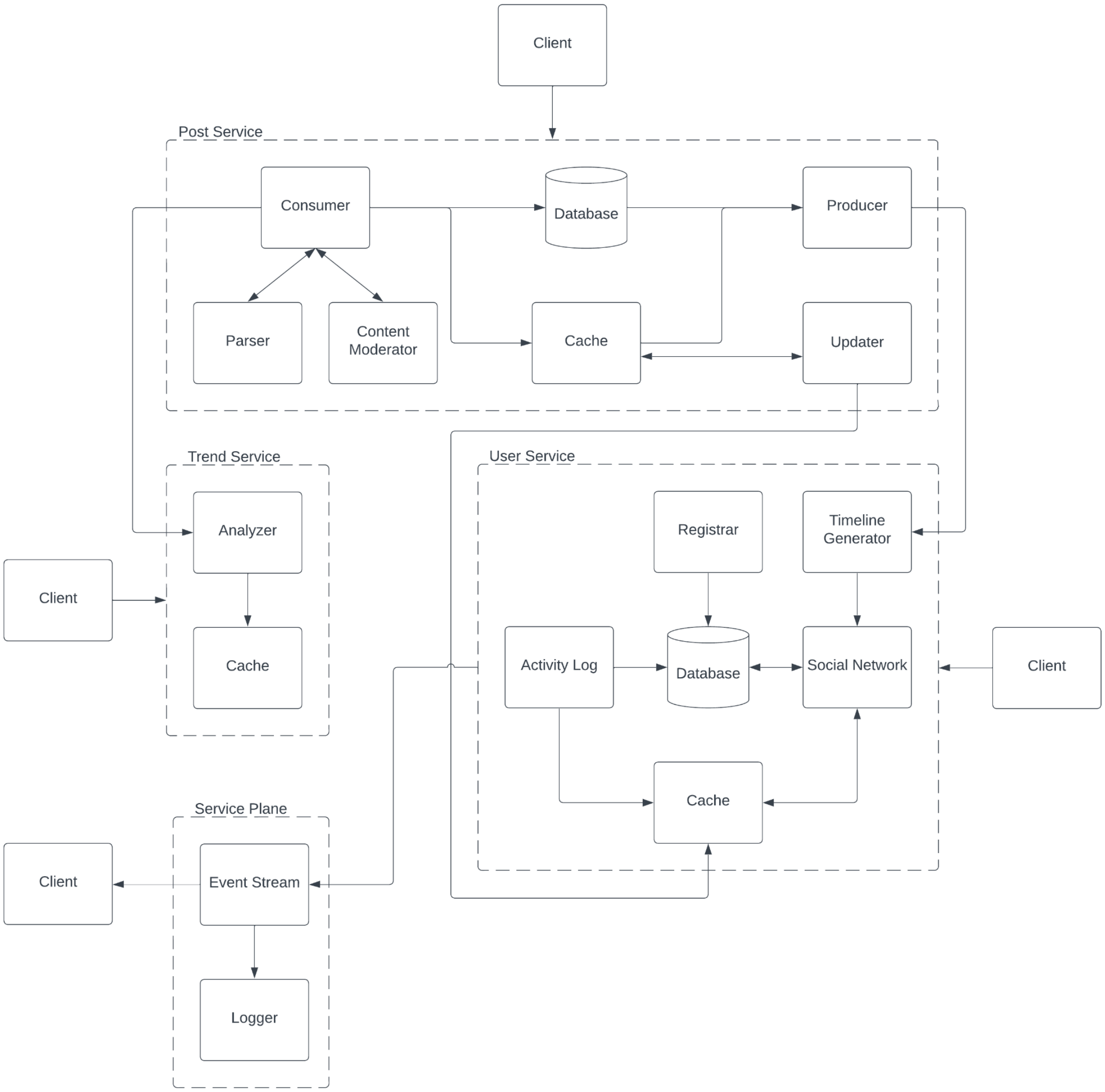

Figure 1: Architecture Diagram

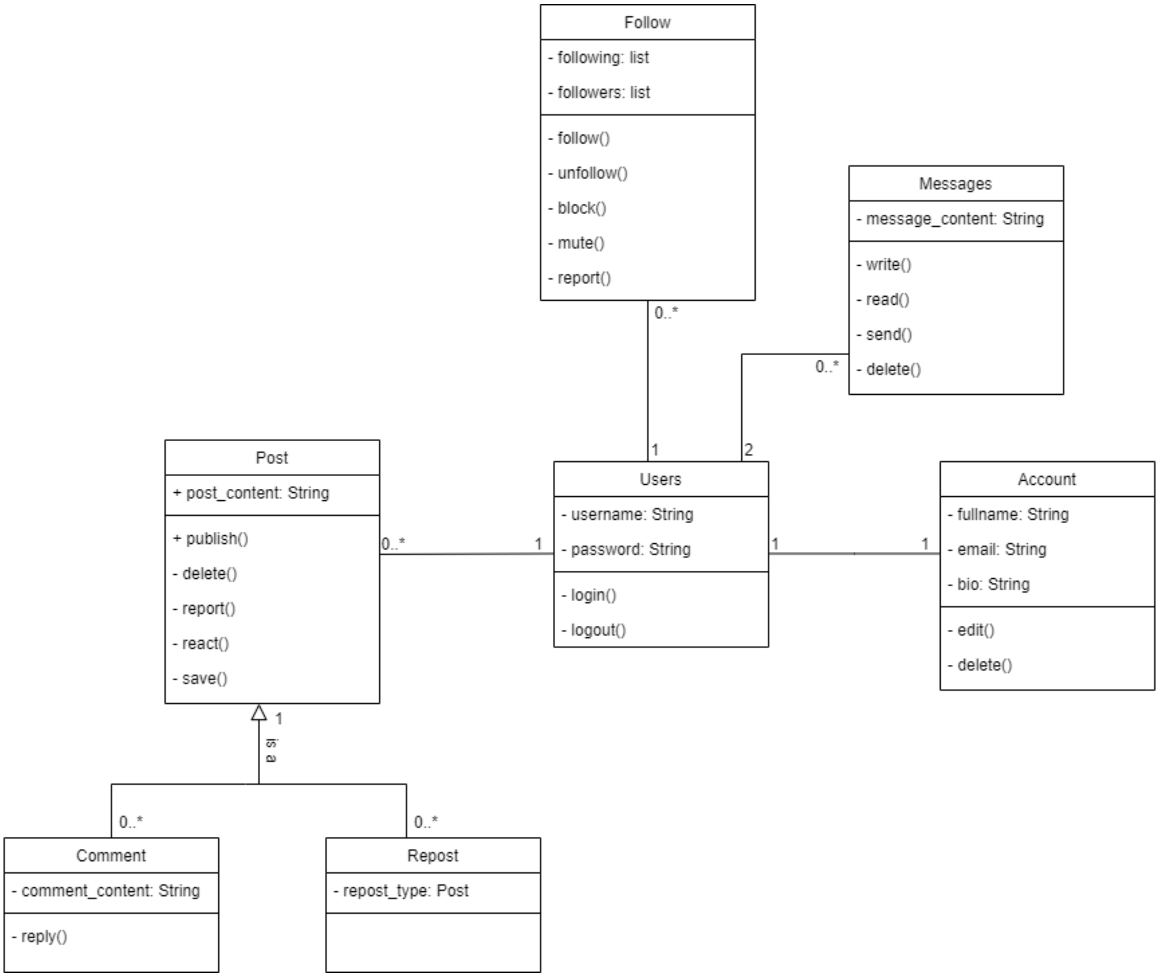

Figure 2: Class Diagram

Infrastructure

Django

Django is a Python web framework that streamlines web app development, handling tasks like database management and user authentication. In our app, Django will manage user accounts, posts, and interactions, providing a secure foundation for building dynamic web applications efficiently. This allows developers to focus on creating unique features while leveraging Django's powerful built-in functionalities.

Pipenv

Pipenv is a Python packaging tool that combines dependency management and environment management. It automates the setup of virtual environments and uses a Pipfile to track dependencies. Deterministic builds are ensured through the Pipfile.lock, promoting consistent project reproduction across systems.

Pylint

Pylint is a static code analysis tool for Python that checks source code against a coding standard, looks for programming errors, and offers suggestions for code improvement. It can also help enforce a coding standard, detect code smells, and identify unused code. Pylint is highly customizable and integrates well with most development environments, making it a valuable tool for maintaining code quality in Python projects.

Name Conventions

Our preferred language is Python. The widely accepted style guide for that language is found here. We may choose to include our linter as part of our CI/CD pipeline to enforce the conventions described by the linked document but will defer that decision until higher-priority matters have been addressed.

Code

*Key files: top **5** most important files (full path). We will also be randomly checking the code quality of files. Please let us know if there are parts of the system that are stubs or a prototype so we grade these accordingly.*

File path with clickable GitHub link	**Purpose (1 line description)**
NLPeace/NLP/models.py	This is where we define the algorithms used to
NLPeace/NLP/main.py	
NLPeace/NLPeace/manage.py	This file holds the main() function for our project.
NLPeace/Dockerfile	This file builds the Docker container for our project.
NLPeace/.github/workflows/django.yml	This file defines the CI pipeline when a PR is made on main.

Testing and Continuous Integration

Each story needs tests before it is complete. If some class/methods are missing unit tests, please describe why and how you are checking their quality. Please describe any unusual aspects of your testing approach.

List the **5** most important test with links below.

Test File path with clickable GitHub link	What is it testing (1 line description)
https://github.com/seaiam/NLPeace/actions	automatic build testing after every PR on main

Continuous integration enables code changes from multiple developers to be merged automatically, in which automated tools test and validate the code before it's integrated. Continuous integration for this Django application was set up using GitHub Actions which is a CI platform provided by GitHub. It works by running a test whenever a pull request is made, and informing the developer whether the tests

have passed or not. So far, we only test that the project is correctly built. As the project advances, we will have more meaningful tests.
Link to CI: django.yml