# Known Issues and Future Enhancements

## Tokens stored unencrypted in database

While we do hash user login passwords, Mastodon Oauth tokens are much more tricky to handle. This is because if we were to hash them before storing them into our database, we will have to unhash them before we can interact with the Mastodon API. This is not a simple task.

Best practices require tokens, such as user access tokens, to be stored in an encrypted secrets vault. This can be done in future hypothetical iterations of the project.

## Database configuration and permissions

Database access by the application needs to be regulated through user-level permissions. In particular, since the test suite drops and creates the database schema for each test, the test suite should not have the permissions to drop the schema for the production database, and should only have access to the test database.

## Infinite Scrolling

We currently have a minimal viable product that amalgamates feeds from different servers. However, a fixed amount of posts is currently being fetched from each server. An important feature to add would be infinite scrolling: As the user scrolls towards the end of the home page, more posts should be fetched and amalgamated from the various servers.

## User Experience Improvements

There are details required to make the user experience smoother. These include:

- Confirmation message should appear when a new account is successfully registered
- Autofocus on fields (eg. on registration page)
- Requirements for a valid password (length, character diversity etc.) should be clearly displayed to the user
- Register/Login buttons should not be clickable if the required fields are not filled

# Retrospective

## What could have gone better?

- **Finding a place and time to meet**. Predominantly due to scheduling conflicts.
- **Merge Conflicts**: more frequent merges would have been better.
- **Being more responsive**: We felt that if we were more responsive as a team we could have avoided some unnecessary confusion.
- **2 approvals for PRs** seems a bit too much: considering the scale of this project, maybe restricting to 1 approval for PR would have been better.

## What went well?

- **Scope was just right -** given the time frame.
- **Learning -** We all had something to take away from this project. We built things based on our collective experience.
- **Delivered what we promised -** we were able to keep up to what we initially planned to do.

## Learnings

- Technical:
  - **Clean Code:** We learnt the importance of clean code and how to write clean code by using software engineering principles.
  - **UI:** Missing functionality to build custom web components
    - Through DOM manipulation
    - That approach could help to prevent merge conflicts bu building encapsulated custom components like navigation bars for example
    - https://developer.mozilla.org/en-US/docs/Web/API/Web_components/Using_custom_elements
  - **Logging and Exception handling**
  - **Testing**

- Non Technical:
    - **Software engineering is more than code! :** Involves getting people to work together, solving “people” problems more than technical problems.
    - **Scoping is very critical:** it’s extremely important to understand the needs and resource constraints that we have and scope the project accordingly. This helps to deliver what we promise.
    - **Task breakdown:** It is critical to break down the tasks without much overlap, so that people can work on it independently and we can move more swiftly as a team without any conflicts.