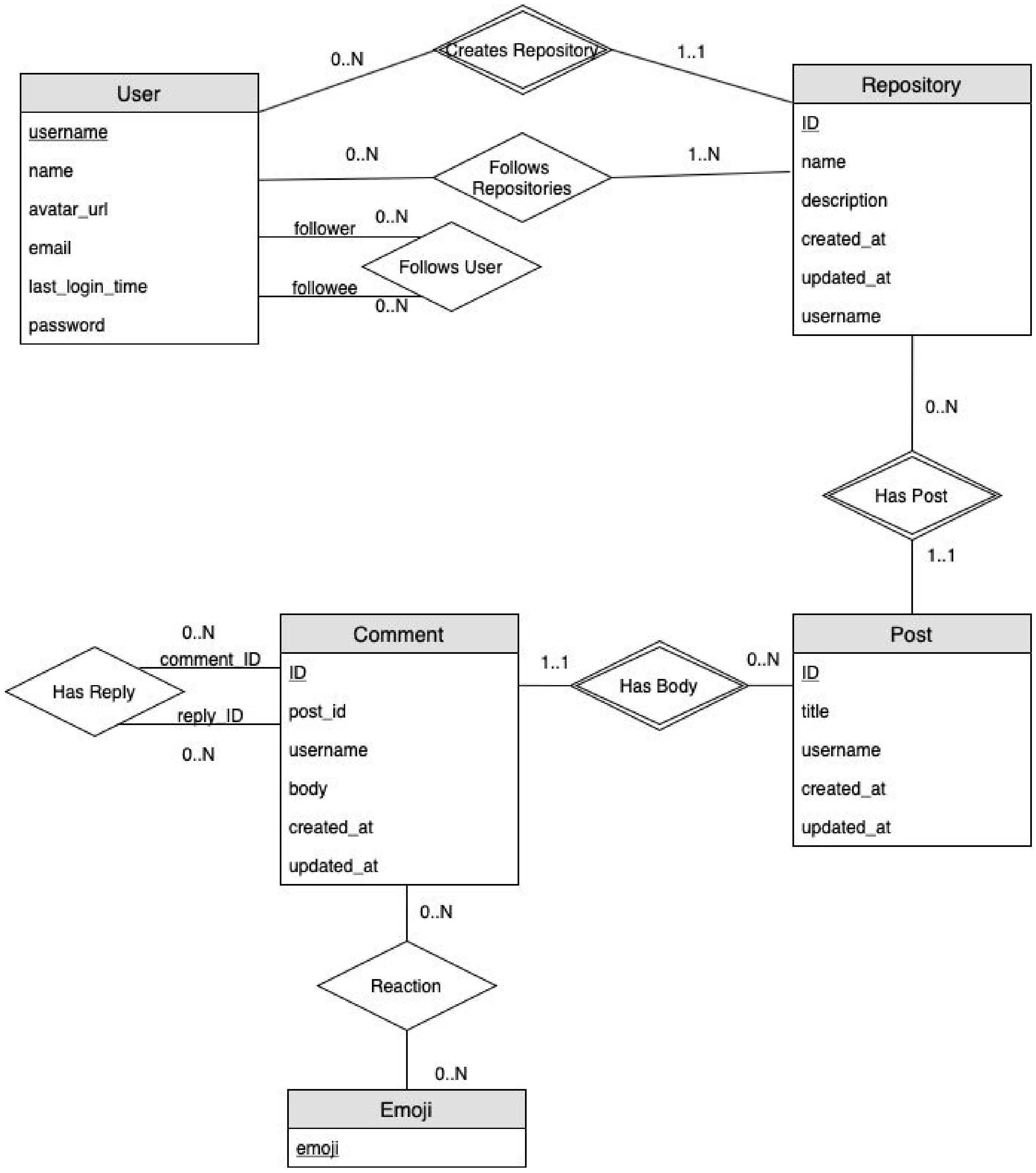
Social Network Project

By Zahin Mohammad and Brian Norman



Description

This Social Network is what we like to call a “Github Companion” Social Network, centered around Github’s “Issues” feature. The Repositories in our network are analogous to “Topics” described in the project requirements, while Issues/Posts in our network are analogous to “Posts”.

Users create repositories (they cannot exist without a creator) and users post issues to Repositories (cannot exist without a creator). Additionally, a user can create multiple repositories and a repository can have multiple issues. Therefore the repository creation and post creation entity sets are weak-entity sets.

In our Network we let users post comments to Issues, as well as replies which themselves are comments in a nested Reddit-like fashion. Comments can have Reactions which are thumbs up/down, laugh emoji, and the rocket ship emoji. This can be extended to every emoji given our schema and model, but we chose to limit it for ease of data scraping. As a bonus, our Social Network provides basic Markdown support for issue comments.

For user authentication we simply store the username and password of users in a User table, which is not encrypted. Because we predictably could not scrape user password data from the Github API (pity!) we chose to default everyone’s password to “password” for this proof of concept. Our current front-end application does not support creating new users, but has the infrastructure available to do so (a simple POST request with the new data would be sufficient). For the proof of concept the users that exist in the sample data were used. For a more refined proof of concept, it is possible to salt and hash the passwords for more security.

The sample data was retrieved from the github api, and revolves around a list of specified users. The current data is based around the user “atulbipin”, but can extend to multiple users. There is no special reason that this user was chosen other than the fact that they did not follow too many users and repositories (making it easier and faster to run our web-scraping script). The web-scraper takes a list of specified users and finds their repositories, and all the comments/issues/reactions created in those repositories. Following this, the script gets a list of users the specified users follow and runs a modified version of the web-scraping of the specified users, we can refer to these users as secondary users and secondary repositories/issues/comments/issues. The reason the web-scraping was limited for secondary users is because the time it takes to complete web-scraping was taking too long, and github has API limits that made it hard to continuously develop and refine the script. For the purpose of sample data, this proved to be efficient.

Some caveats on using the github api for sample data was handling special characters, and importing this into the database. The method for importing data was using a CSV file, and as such some special characters posed problems such as “,” and “;”. This was solved by replacing any special character with their string representation such as “;” became “SEMICOLON”. Obviously this could potentially pose some data integrity loss if an issue or comment had the words “SEMICOLON”, but for the purpose of this proof of concept it was sufficient. When inserting data, the server string-escapes all special characters before insertion.

Lastly, the technologies used to create this proof of concept will be discussed. For the database MYSQL was used, running inside of a docker container. The front-end application is a “reactjs” application. The server that facilitates the front-end requests and communicates with the database is an “Express” server. To run this project, detailed “readme” guides are provided (only tested on macos and ubuntu devices).