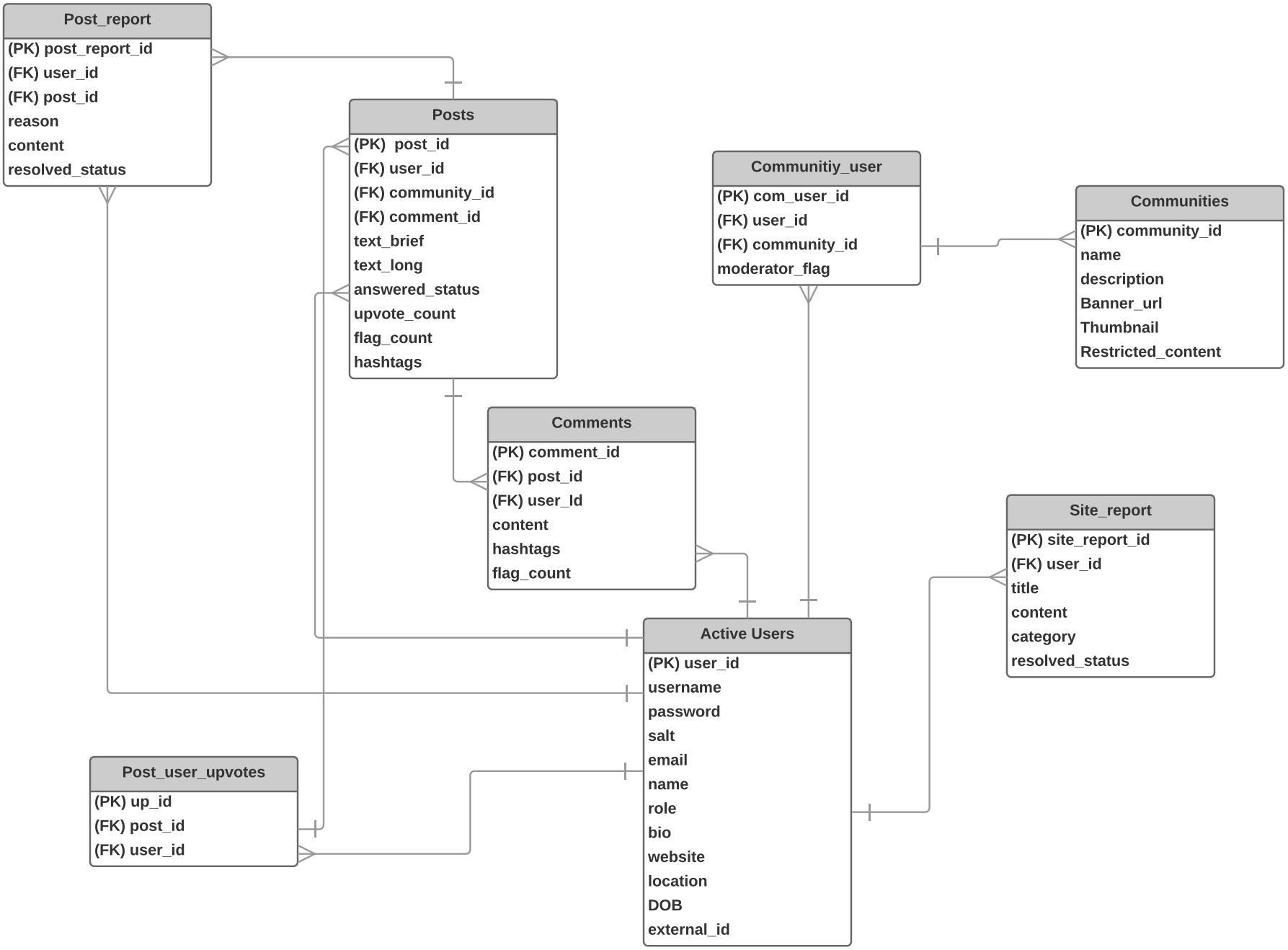
AMABLE E-R DIAGRAM

Jack Barry, Max Bender, Chris Cordaro, Dan Jast, Ethan Turkeltaub
September 21, 2016



Entity Relationship Diagram Documentation

Entity Definitions and Data Descriptions:

Active_Users: An active user has a unique identifier called user_id that is sequentially assigned to the user when they create an account with Amable. The other fields are maintained for login credentials and operational business use and research. The data types of the fields are as follows: User_id is an integer, username is a string, password is a string that will be salted and hashed for security, salt is a string, email is a string, name is a string, role is a string, bio is a string, website is a string, location is a string, DOB is a date, external_id is an integer.

Posts: An active user has the ability to create posts. Each post has a unique identifier generated sequentially. A post has several foreign keys that include user_id, community_id, and comment_id. The user_id is the unique identifier of the active user who the post belongs to. The community_id indicated which community the post belongs in. The comment_id is a list of the unique identifies for the comments that are linked to the post. The content of the post includes a text_brief which is up to 140 character and is what is seen on the news feed, the content also includes a text_long which is a 5000 character limited description of the post and is only seen if a user clicks to see the description. The answered_status of the post shows active users if the kindness has helped the owner of the post and is changed by the owner of the post between open and enlightened. Each post also has an upvote_count that is incremented each time an active user who hasn't already upvoted the post upvotes the post. The flag_count is incremented everytime an active user submits a report of the post. Hashtags are used to bring attention to certain groups and trending themes. The data types of the fields are as follows: post_id is an integer, user_id is an integer, community_id is an integer, comment_id is an integer, text_brief is a string, text_long is a string, answered_status is a Boolean, upvote_count is an integer, flag_count is an integer, hashtag is a string.

Post_report: A post_report is meant to keep track of posts that have been flagged by active users as inappropriate. The report as a unique identifier that is created sequentially as well as foreign keys to keep track of who the report belongs to and what the flagged post is. The port_report has a 140 chracter limited reason and a 5000 character limited content. The resolved status is changed by the system or moderator of the post's community between open and resolved. The data types for the fields are as follows: post_report_id is an integer, user_id is an integer, post_id is an integer, reason is a string, content is a string, resolved status is a Boolean.

Community_User: The community_user entity serves the purpose of connecting the active users and the communities as well as tracking if a user is a moderator for a specific community. The data types for the fields in community_user are as follows: com_user_id is an integer, user_id is an integer, community_id is an integer, moderator_flag is a Boolean.

Communities: Communities represent the interest groups of Amable users. The community_id is a unique identifier that is generated sequentially. The name is a 140 character limited title and the description is a 5000 character limited description of the community. The banner and thumbnail are for presentation and the restricted_content field is a flag to show if the community contains age restricted material. The data types for the fields in communities are as follows: Community_id is an integer, name is a string, description is a string, banner_url is a string that represents a url connected to an image, thumbnail is a string that represents a url connected to an image, restricted_content is a Boolean.

Comments: Comments are 140 character limited responses to posts. Each comment has a sequentially generated unique identifier and foreign keys to keep track of the post the comment is in response to and the user who made the comment. The content of the post is limited to 140 characters. Hashtags are included in the 140 characters and are used to bring attention to certain groups. The data types for fields in the comments table are as follows: comment_id is an integer, post_id is an integer, user_id is an integer, content is a string, hashtags is a string, flag_count is an integer

Post_user_upvotes: post_user_upvotes keeps track of who upvoted a particular post. This is to display who upvoted the post and to make sure an active user cannot upvote a post more than once. It has a unique identifier that is generated sequentially. The data types for post_user_upvotes are as follows: up_id is an integer, post_id is an integer, user_id is an integer.

Site_report: The site_report table is to track feedback that active users send to us about the interface design and functionality. Each report will have a unique identifier generated sequentially. The report will also have a user_id to track who generated the report. The title of the report is a 140 character limited field, the content of the report is a 5000 character limited description of the issue. The category groups the reports into interface design and interface functionality. The resolved_status is changed by the admin between the categories of open, being worked, resolved, or unconsidered. The data types for the site_report are as follows: site_report_id is an integer, user_id is an integer, title is a string, content is a string, category is a string, resolved status is a string.

IT Requirements

1. PureFlex System
   1.1. Physical system requirements
      1.1.1. Storage Capacity: We currently have 75 GB of storage allocated to our team on the system. If we need more, we will contact the system administrators.
      1.1.2. Speed requirements / response time parameters: All our machines will share 8GB of RAM which should be plenty. Webservers and command line VMs will have 1 socket and 2 CPUs, where a Desktop or Windows machine will have 1 socket and 4 CPUs. All VMs will have thin partitioning to preserve disk space.
   1.2. Virtual system requirements
      1.2.1. OS to be supported: The operating system we are going to use is still undecided. We are either going to be using Ubuntu 14.04 server, or SELinux 2.4, because it adheres to the standards set by the DOD, which is a requirement set by our client.
      1.2.2. Number of images expected: We are only going to need 1 image. This image will host the webserver (Nginx), and the database (Postgres). If we need to perform load balancing/high availability solutions for high amounts of concurrent users, we would spin up multiple VMs and manage them as a cluster to balance load upon the multiple machines.
   1.3. Connectivity
      1.3.1. Network considerations: All machines will be connected to Foxnet. SSH will be available to access machines offsite once a VPN connection is established.
      1.3.2. Interconnection to what other systems: Connection to other systems will be provided when the need for other systems is established.
2. Reliability
   2.1. Service Level Agreements
      2.1.1. Uptime requirements: The servers will be up and running as long as our servers do not exceed the limitations set by the system administrators.
      2.1.2. Response time requirements: The system will be as responsive as the hardware specs allow it to be.
3. Recoverability
   3.1. Where things are backed up: Backups of files and databases will be stored locally in our development environment. We will also download an OVA of our machine periodically to save the work and progress on the server.
   3.2. Transient Data: Certain data will not need to be stored in backups, because it will be changing very often.
4. Security and Privacy
   4.1. Database
   - We will need a postgresql database installed
   - Need a postgres user amable/“some secure password”
   - Need a database with full permissions for “amable” user called “amable”
      4.1.1. Access controls by userid / roles: The “amable” database should only be accessible by user amable
      4.1.2. Update vs. Access: We will do database administration through the postgresql super user
   4.2. Account information
      4.2.1. User data: encrypted database

4.2.1.1. Personal / registration
4.2.1.2. Communities /Kindness

4.3. Admin access controls

4.3.1. Adding new users, deleting old: Only “postgres” user should have the ability to modify any users

5. Maintenance

5.1. Planned down time requirements

5.1.1. Database maintenance: Database updates are done through version migration. Seamless updates

5.1.2. Updates to Website/Application; These updates will be rolled out as developers see necessary and are prepared to roll them out.

5.1.3. Times of year when IT does maintenance: The System Administrators will let us know if there are any days in which maintenance will be performed on the system our servers are hosted on.