Saad Bhuiyan, William Cao, Ethan Chen, Peihua Huang (Team SPEW)
SoftDev1 pd2
P #00: Da Art of Storytellin'
2019-10-21

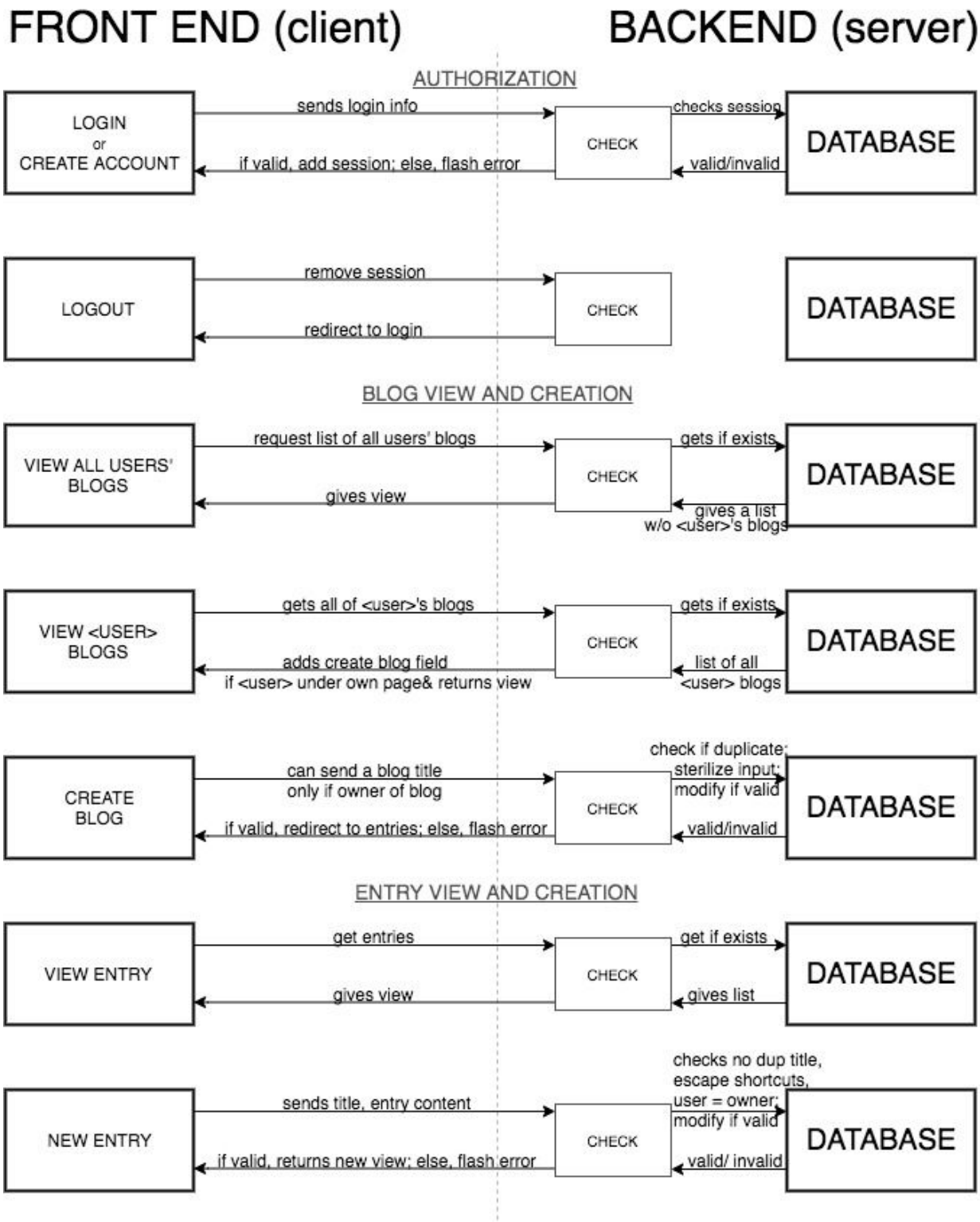
Team: SPEW

Project Name: SPEWING BLOGS

Work division:

Project Manager: William Cao
Frontend: Ethan Chen
Database: Saad Bhuiyan
Backend: Peihua Huang

Component Relationship

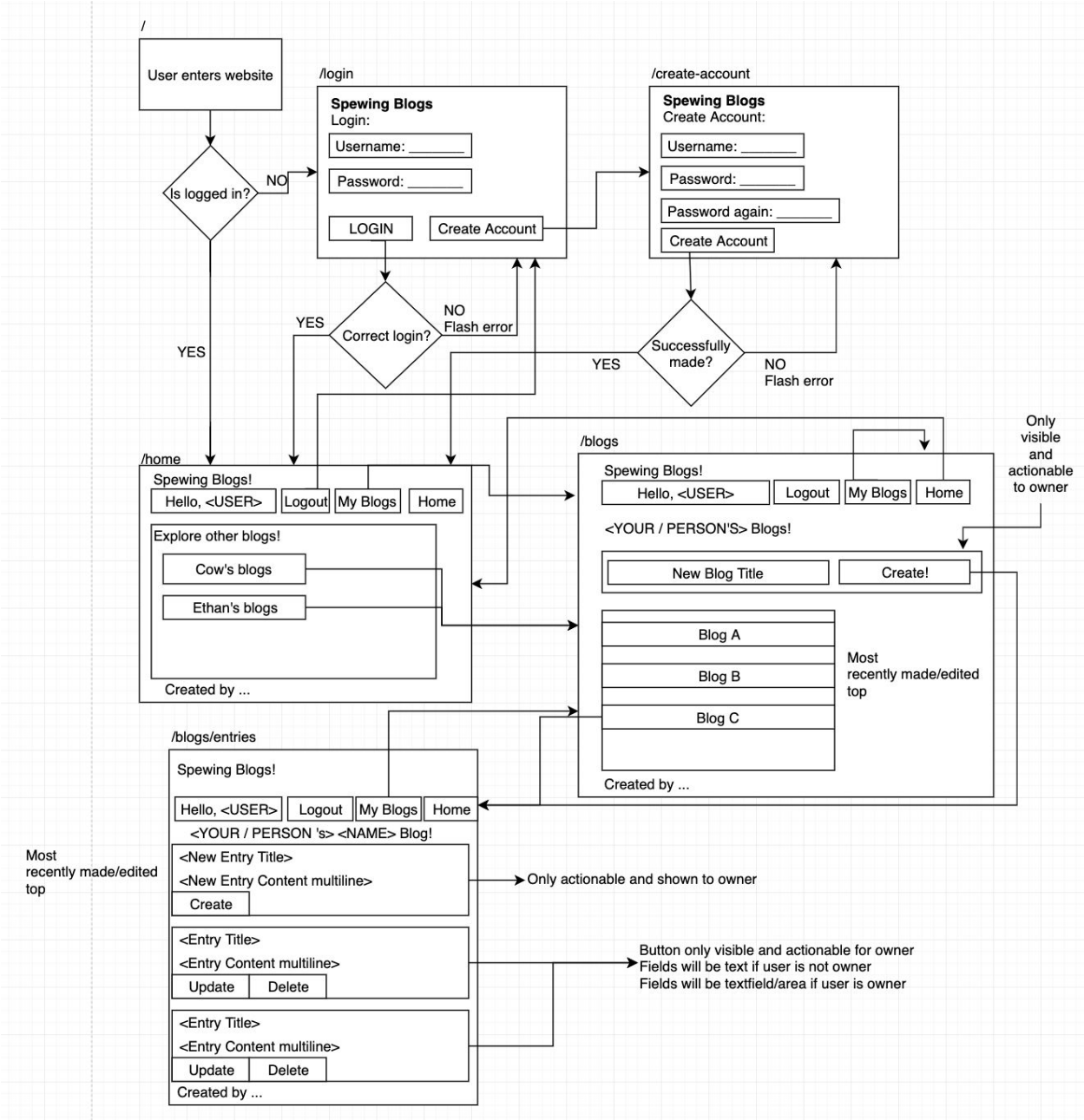


Backend (Flask)

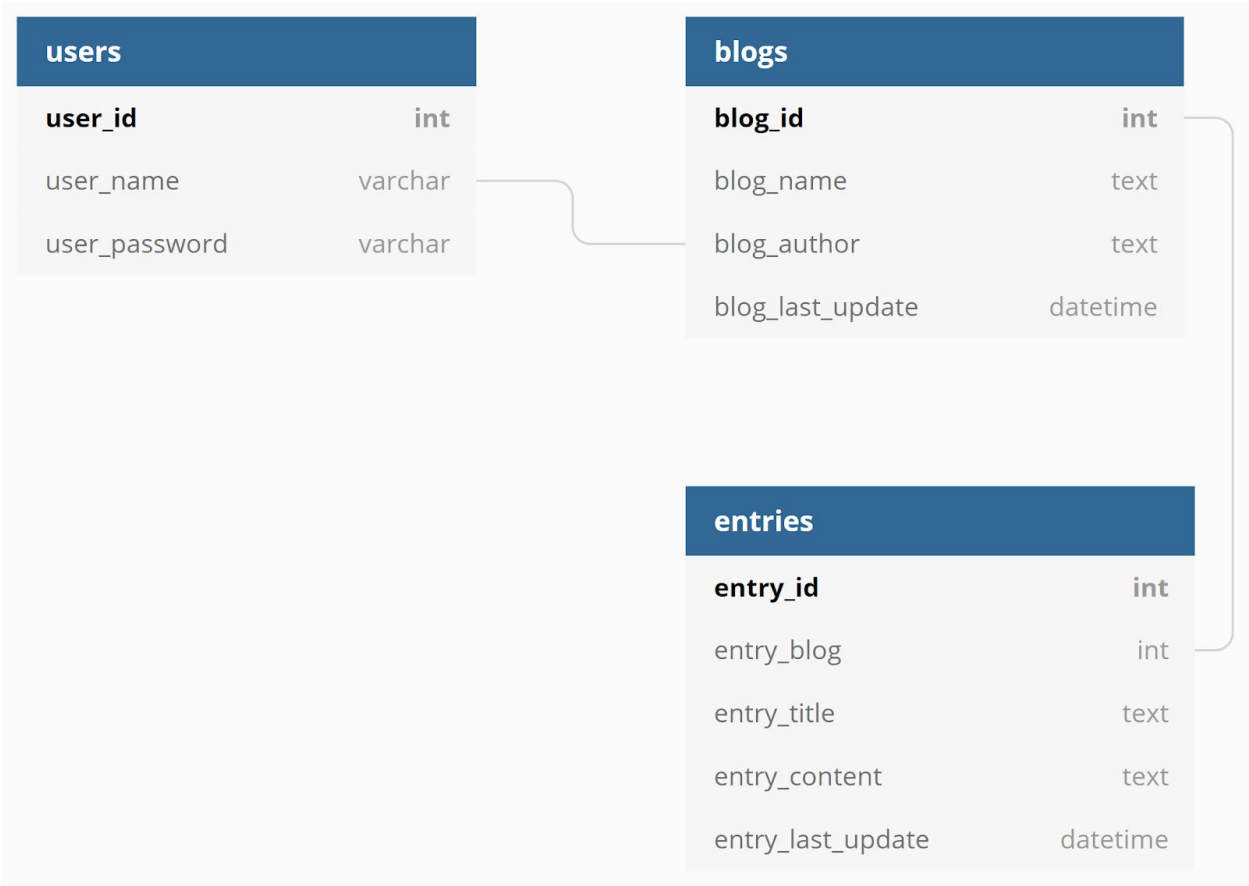
- /
    - Function checks if there is a current session open
        - If there is (meaning user is logged in), redirect to /directory
        - Else, redirect to /login
- /login
    - Load login.html template
    - If username and password is entered, check with database to see if it matches.
        - If it does, add a session (store username and id) and redirect to /directory
        - Else, flash error message
    - If create account button is clicked, redirect to /create-account
- /create-account
    - Load create account html template
    - If username not in database, add username, password, and id to database (meaning account was created successfully) and redirect to /blogs
    - Else if bad username (check single quotes, username already exists, passwords don't match, empty fields), flash error message
- /home
    - Get username from session and display on top of html page
    - Get and display list of usernames (excluding the logged in user)
        - If user clicks on one of the usernames, make a GET request to redirect to /blogs with that GET request
    - Logout button: if pressed, pop session and redirect to /login
    - My Blog button: redirect to /blogs viewing currently logged in user blog
- /blogs
    - Loads html template with all the blog titles of the user that is currently logged in, if there is no GET request.
        - Else, display the blogs of the user in GET request.
    - Check with session to see if user logged in matches the owner of the blogs.
        - If yes, show create! Button and textbox for user to create a new blog
            - If blog title is not found in the database, add the new blog to the database, give it an id, and redirect to /blogs/entries with GET requests blog_id and blog_title
            - Else, if duplicate blog, flash error
    - Logout button: if pressed, pop session and go back to /login
    - My Blog button: if pressed and user is not in their own blog, redirect to user's blog
- /blogs/entries
    - Get the list of entries for the blog returned from GET request in time order (based on last edit time) from the database and display it in chronological order (most recent first).

- Check with session to see if user logged in is the owner of the blog
    - If yes, show create, edit, update, cancel and delete button and the choice to input a new entry title and entry content.
        - Create button: If entry title is not a duplicate, add the entry to database. Display the new entry above the most recent entry.
            - If entry title is a duplicate: flash error.
        - Edit button: If pressed, text in the entry all gets put in a text box where user can edit.
            - Update button: If pressed, the entry text in database is replaced with the new text, timestamp is updated, and entry is now displayed at the top.
            - Cancel button: If pressed, the text box disappears and the entry is reverted to the version stored in database.
        - Delete button: Entry title and entry is deleted from the database and thus removed from website.
- Logout button: if pressed, pop session and go back to /login
- My Blog button: redirect to /blogs displaying the blogs of the current user

Front end:


/
User enters website
Is logged in?
NO
YES
/login
Spewing Blogs
Login:
Username: _____
Password: _____
LOGIN
Create Account
/create-account
Spewing Blogs
Create Account:
Username: _____
Password: _____
Password again: _____
Create Account
Correct login?
YES
NO
Flash error
Successfully made?
YES
NO
Flash error
/home
Spewing Blogs!
Hello, <USER>
Logout
My Blogs
Home
Explore other blogs!
Cow's blogs
Ethan's blogs
Created by ...
/blogs
Spewing Blogs!
Hello, <USER>
Logout
My Blogs
Home
<YOUR / PERSON'S> Blogs!
New Blog Title
Create!
Blog A
Blog B
Blog C
Most recently made/edited top
Created by ...
Only visible and actionable to owner
/blogs/entries
Spewing Blogs!
Hello, <USER>
Logout
My Blogs
Home
<YOUR / PERSON 's> <NAME> Blog!
<New Entry Title>
<New Entry Content multiline>
Create
Only actionable and shown to owner
<Entry Title>
<Entry Content multiline>
Update
Delete
<Entry Title>
<Entry Content multiline>
Update
Delete
Created by ...
Button only visible and actionable for owner
Fields will be text if user is not owner
Fields will be textfield/area if user is owner
Most recently made/edited top

Backend (Database)



Database file contains three tables - `users`, `blogs`, `entries` - with relevant information in each, as shown in the schema above.

`db_manager.py` will contain all the methods that the front end needs to work with the database (i.e. getting, adding, verifying, etc.)

`add_login(username,password)` adds new login credentials to the `users` table.
`verify_login(username, password)` takes login credentials and checks the `users` table to verify that the login credentials are correct.
`get_usernames_with_blogs()` returns a list of all usernames that have blogs by checking the `blogs` table.
`get_blogs_for_username(username)` returns a list of all the blogs a username has created by checking the `blogs` table.
`create_blog_for_username(username, blog_title)` creates a blog for a username in the `blog` table.
`get_entries_for_blog(blog_id)` returns all the entries (as tuples with the title and content) for a blog ordered by most recently updated.

`is_owner(username, blog_id)` verifies whether the user is the owner of the blog.
`add_entry(entry_title, entry_content, blog_id)` adds an entry to the `entry` table.
`remove_entry(entry_id)` removes an entry given the entry's id.
`get_entry_id(entry_title, blog_id)` returns the entry's id given the title and blog.
`get_blog_id_from_title(username, blog_title)` returns the blog's id given the title and author.