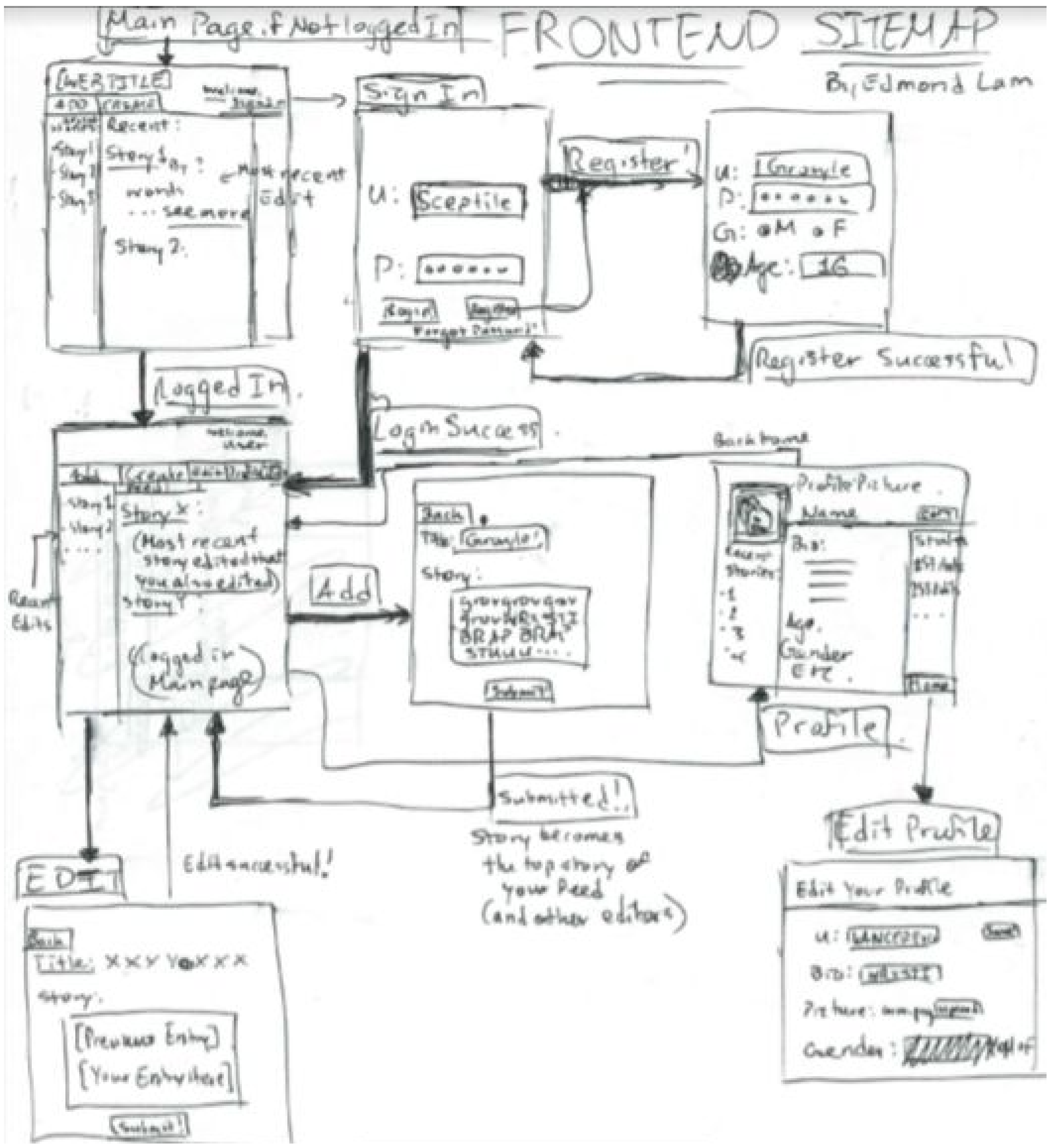
Team Pending
Noah Fichter, Julius Freyra, Edmond Lam, Rodda John, Jessica Titensky
SoftDev1 pd8
HW10 -- De Art of Storytellin’
2016-10-27

Database Schema

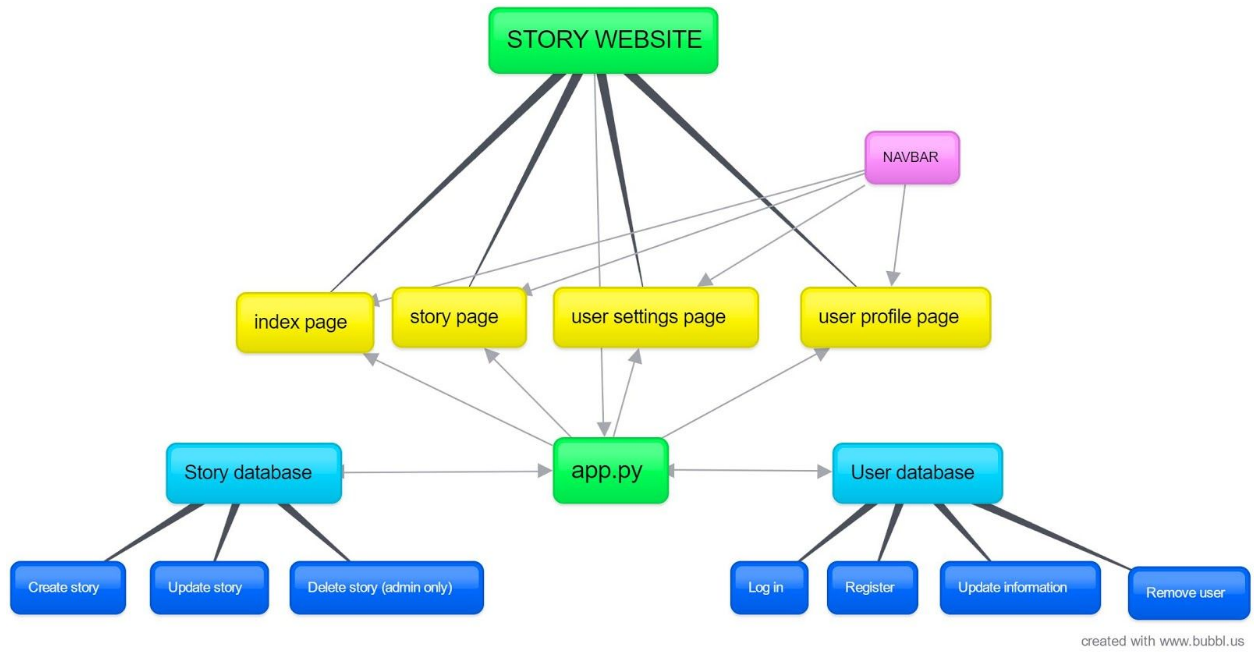
Stories					
Stories ID (int)	Title	Latest update (text)	Timestamp of last edit (text, epoch)	Timestamp created (text, epoch)	Contributor IDs (text)
1	“None Were There Then And.”	“And then there were none.”	1477591036151	1477590036151	“3,7,888, 9”

Users					
User Id (int)	Username (text)	Password, hashed (text)	Age (int)	Email (text)	Posts contributed to IDs (text)
1	laughing-octo-pancake	4375yt45hg5y45t7845h	123	“moc.liamg@gmail.com”	“3,14,15, 9265”

Sitemap


FRONTEND SITEMAP
By Edmond Lam
Main Page if Not logged In
WEBTITLE
Recent:
Most recent Edit
...see more
Sign In
U: Sceptile
P:
Login
Register
Forgot Password?
Register!
U: Grovyle
G: M F
Age: 16
Register Successful
Logged In.
Login Success.
Recent Edits
Story X:
(Most recent story edited that you also edited)
Story Y:
(Logged in Main page)
Add
Back
Title: Grovyle
Story:
Submit
Back home
Profile Picture
Name
Edit
Recent Stories
Bio:
Age
Gender
Etc.
Profile.
Edit Profile
Submitted!
Story becomes the top story of your feed (and other editors)
EDIT
Edit successful!
Back
Title: XXYYXXX
Story:
[Previous Entry]
[Your Entry here]
Submit!
Edit Your Profile
U:
Bio:
Picture:
Gender:

Component Map



Roles

- **Rodda** - Project Manager
    - Ensure the following roles are being carried out
    - Ensure teamwork
    - Ensure everyone is happy with their roles
    - Handle minor coding tasks as necessary
    - Handle any modifications to this document
- **Edmond** - Frontend Manager (HTML, Jinja2)
    - Users will only be able to access all pages except index.html if they are logged in - if not, they are redirected to home page
    - ***this is handled by Julius but is fitting to put here to help guide reading through the next few items***
    - Every page:
        - Settings - updating information
        - Profile - posts the user has contributed to
        - Displays a scrollable navbar with all stories' clickable titles (recently updated at the top)
    - index.html
        - Displays a scrollable body with all stories that the user viewing has contributed to (information + last update)
    - story_id.html
        - If user contributed to this post already, show the full story
        - If not, show the title, last update, a text field to contribute, and a button to submit
    - profile.html
        - Place for user to update password, email, and any other information we may ask for in the future
    - user_id.html
        - Displays a scrollable body with all stories that the selected user has contributed to (information + last update, unless the user viewing has contributed to it as well, in which case a link to view the full story is added)
- **Julius** - Frontend-Backend Linker (Python, Flask)
    - Basically a connecting body between entirely frontend (Edmond) and entirely backend (Noah & Jessica)
    - Functions that take Noah and Jessica's functions and connect them to app.py to connect them to Edmond's hmtl files

- Login/Register Methods
  - register_new_user(username,password,age,email)
    - Passes values given to register in user_manager.py
  - login(username,password)
    - Passes values given to login in user_manager.py
  - delete_account()
    - Passes session value of username to remove in user_manager.py
  - update_account()
    - Passes session values of updated fields password or email to either update_password or update_email in user_manager.py depending on which form is submitted
- Story Methods
  - Mode - integer variable denoting whether or not a user is contributing or viewing
  - get_story_list(userid)
    - Returns list of stories to be displayed on feed and respective elements to be put on sidebar
    - If mode is viewing (1), then get_story_list returns a list of stories the user has contributed to to display on the feed
    - If mode is contributing (0), then get_story_list returns a list of stories the user can contribute to to display on the feed
    - Bases order on timestamp of last edit
  - get_story(storyid,userid)
    - Returns a story to be displayed on the full story page
    - If userid is in the contributor ids field of the story entry in the database, returns the text within a story's specific text file, showing the whole story
    - If userid is not in the contributor ids field of the story entry in the database, returns the last update of the story
  - write_to_story(storyid,userid,addition)
    - Passes values storyid, userid, addition, and the time to update_story in story_manager.py
  - make_story(userid,addition)

                - Passes values userid, addition, and time to create_story in story_manager.py
- **Noah** - Database Manager #1: Stories (Python, SQLite3)
    - story_manager.py
        - Possibly turn this into a class instead of a bunch of functions (i.e. Story object) [see below]
        - create_story(userid,title,timestamp,text)
            - Creates new entry in database:
                - userid goes in string for users that have contributed
                - title, timestamp copied into database
                - timestamp copied into last updated timestamp
                - text copied into last updated text
                - new text file created with new postid which then goes into the database
        - update_story(userid,postid,timestamp,text)
            - Checks if userid has already contributed to this post, if so, end
            - Adds a comma and the userid to the string for users that have contributed
            - Opens the file post<id>.txt, appends text to it
            - Updates last updated text with given text
            - Updates last updated timestamp with given timestamp
        - delete_story(postid)
            - Deletes a story's entry in the database as well as its text file entirely
            - Only for admin/testing use, will not be accessible to users
- **Jessica** - Database Manager #2: Users (Python, SQLite3)
    - user_manager.py
        - Possibly turn this into a class instead of a bunch of functions (i.e. User object) [see below]
        - register(username,password,age,email)
            - Checks if the user exists, ends if it does
            - Creates new entry in database including given information (w/ hashed password) as well as a new user id and an empty string for posts contributed to
            - Possibly certain password requirements
        - login(username,password)
            - Checks if the user exists, ends if it doesn't

            - Checks if, when hashed, the password matches that of the user's in the database
        - remove(username)
            - Checks if the user exists, ends if it doesn't
            - Removes the user from the database entirely
        - Update functions
            - update_password(username,password,newpassword)
                - Checks if the user exists, ends if it doesn't
                - Checks if the password is correct, ends if it isn't
                - Updates the user's password in the database
            - update_email(username,email)
                - Checks if the user exists, end if it doesn't
                - Updates the user's email in the database

Alternate Structure

- Python Objects
    - Alternate to using multiple methods in a python file like get_latest_update(), get_storyid(), etc. in a story_manager python file
    - All story data is stored into an instance of a story object; All user data is stored into an instance of a user object
        - Can be easily manipulated and called by <storyid>.<datafield>
    - Improves organization and modularity
    - Backend would have the ability to create and remove rows from the database and get an object (returning an instance of either a User or a Story object) and to 'commit' changes, saving the state of that object to the database