Team SeQueL - Joseph Lee, Yevgeniy Gorbachev, Eric Lau
SoftDev1 pd1
P00 -- The Art of Storytellin'
2019-10-28

Selected scenario: Web Log

Front-end Feature Listing *(minimum viable product features are indicated by *)*

| Features only for users logged in; anything logged-out users get is the login/registration page **|**

Account-local

- Account creation with encrypted password*
- Create blogs*
- Create or edit entries within own blogs*
  - View edit histories for own blogs*
- Profile Page*
  - View listing of own blogs*
- Settings Page
  - Change password
  - Change username

Sitewide

- Every page contains a heading of links to <home>*, <profile>*, <settings>, <logout>*
- View other users' blogs*
  - Directory with all blogs, sorted by timestamp of latest entry of blog*
  - Search for keywords within title
- View edit history for other users' blog entries
- Comment on other users' blog entries*
- View other users' profiles*
  - Contains links to user's blogs*

Database Layout

Table name	**Record content**
users	userid, username, password[1]
blogs	userid, blogid, title
entries	blogid, entryid, versionid[2], timestamp[3], content
entries_arc	blogid, entryid, versionid, timestamp, content
comments	blogid, entryid, commentid, userid, timestamp, content

[1] Hashed
[2] Starts counting from 1
[3] Unix time of the server

HTML Templates

- login.html
  - `Directed` from “/” if user is not logged in
  - `Displays` text input boxes for username and password
  - On click <submit>, `redirects` user to “/login” through a POST request
  - On click <create new account>, `redirects` user to “/register” through a GET request
- register.html
  - `Directed` from “login.html” on click <create new account>
  - `Displays` text input boxes for information needed to create a complete account
  - On click <create account>, `redirects` user to “/register” through a POST request
- home.html
  - `Directed` from “/login” if credentials are verified
  - `Directed` from link <home> on navigation bar
  - `Displays` all blogs in order of most recently edited
  - On click <blog_whatever>, `redirects` user to “/blog” with whatever’s blog_id
- search.html
  - `Displays` all blogs that match the search request
- profile.html
  - `Directed` from link <profile> or <profile_userid>
  - `Displays` all blogs of that user
  - `Displays` create new blog link if user is viewing their own profile
  - On click <create new blog>, `redirects` user to “/create_blog”
- settings.html
  - `Directed` from link <settings>
  - `Displays` form to <change username> and <change password>
- blog.html
  - `Directed` from “/blog”
  - `Displays` all entries of that blog
  - `Displays` a <view entry history> link for each entry
  - `Displays` comment section at the bottom, with a text input box for new comments
  - `Displays` an <edit> and <enter_entry> link if viewing user’s own blog
  - On click <view_entry_history>, `redirects` user to “/entry_history” with corresponding entry_id
  - On click <edit>, `redirects` user to “/blog/<blog_id>/<entry_id>/edit”
  - On click <enter_entry>, `redirects` user to “/new_entry”
- enter_entry.html
  - `Directed` from link <enter_entry>
  - `Displays` input text box for a new entry
- entry_history.html
  - `Directed` from link <<view_entry_history>
  - `Displays` all versions of entries
- layout.html
  - The links that take you to <home>, <profile>, and <settings>
  - This is the parent for jinja inheritance

Task Division

- Joseph
  - App routing and redirecting
  - Implementing front-end
- Yevgeniy
  - Everything database related
    - Fetching a blog's entries, fetching a list of blogs, etc.
    - Adding users, blogs, entries, comments to database
- Eric (Project Manager)
  - Helping out with frontend and backend tasks when applicable
  - Team management
  - Devlog and updating design doc

Component Map

Frontend (read)	**Relationship**	**Backend**	**Function**
login	Verifies username and password hash equality	`users`	`utl.acc.verify_acc (username, password)`
get userid	Given a username, returns the userid	`users`	`utl.acc.get_userid (username)`
create account	Given a username and password, add account to `users` if requirements are met	`users`	`utl.acc.create_acc( username, password)`
search	Searches for the presence of the input string within blog titles and returns all matches as a dictionary array.	`blogs`	`utl.search (keyword)`
count	Get the number of blogs that exist	`blogs`	`utl.blogs.count()`
get blog's descriptors[4]	Uses `blogid` to retrieve data from `blogs`	`blogs`	`utl.blogs.describe (blogid)`
get blog's entries	Uses `blogid` to retrieve data from `entries` Returns a list of dictionaries in descending order by `entryid` containing `entryid, versionid, timestamp, content`	`entries`	`utl.blogs.read_entries (blogid)`
get entry history	Uses `entryid` to retrieve data from `entries_arc` Returns a list of dictionaries in descending order by `versionid` containing `entryid, versionid, timestamp, content`	`entries_arc`	`utl.entries.read_entries_h (blogid, entryid)`

[4] Title and author

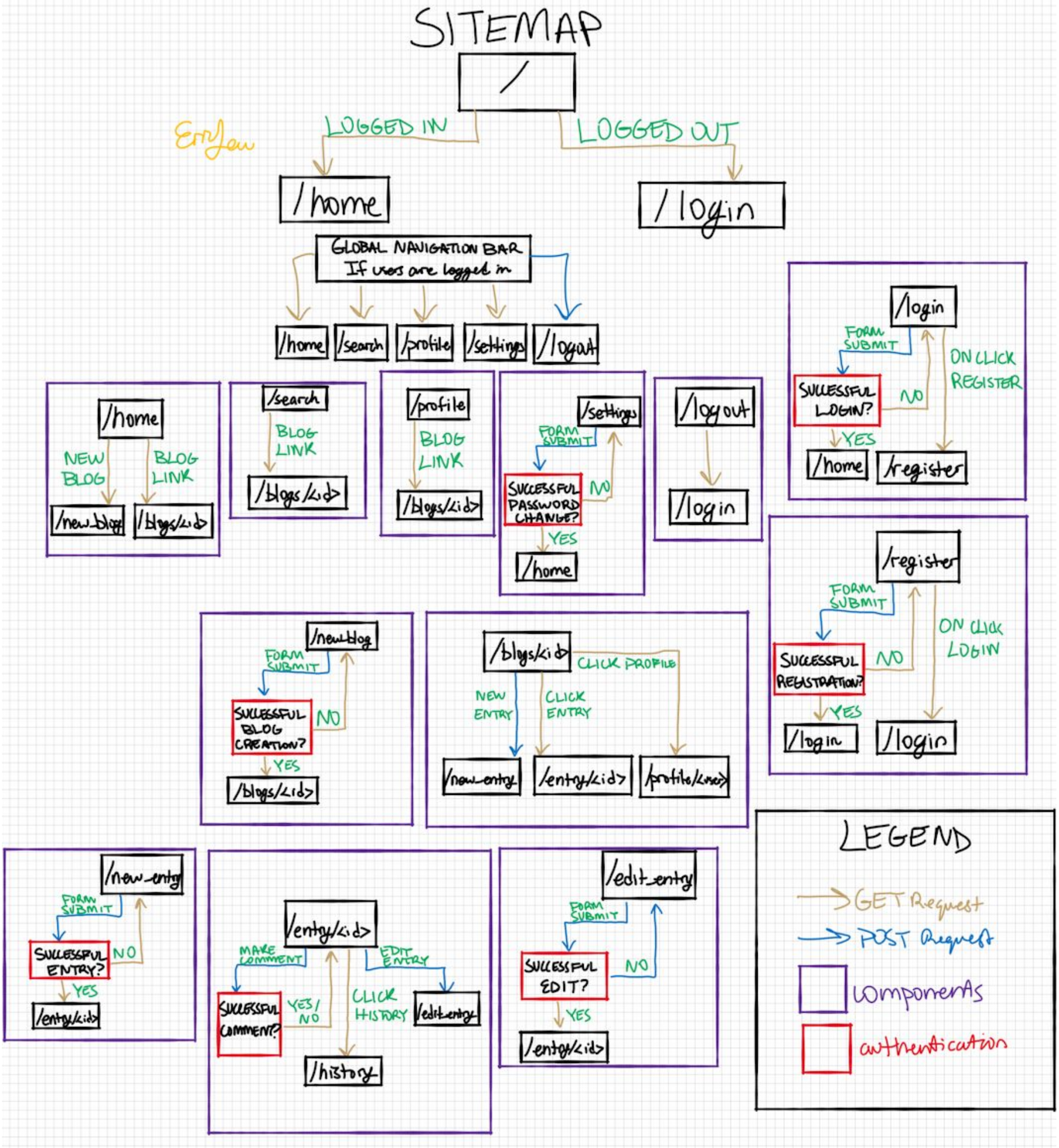
Frontend (write)	Relationship	Backend	Function[5]
get comments	Uses `entryid` and `blogid` to retrieve data from `comments`	`comments`	`utl.entries.read_comments (blogid, entryid)`
register	Validates username[6], hashes password, and generates an id, pushing the values into the users table.	`users`	`utl.acc.create_acc (username, password)`
create blog	Creates a new blog associated with the creating user.	`blogs`	`utl.blogs.create_blog (userid, title)`
Create new entry	Creates a new entry for a blog of the user's choosing, inserts into `entries` and `entries_arc`	`entries` `entries_arc`	`utl.entries.create_entry (blogid, content)`
Edit entry	Edits the corresponding entry in `entries` with the new content and `versionid` and copies the edited entry into `entries_arc`	`entries` `entries_arc`	`utl.entries.edit_entry (blogid, entryid, content)`
Change username or password	Overwrites the current password's hash with the new password's hash or overwrites the current username with the new username	`users`	`utl.acc.edit_acc (userid, new_un=, new_pw=)`[7]
Add comment	Generates and inserts a comment record based on the content and the blog, entry, and user ids.	`comments`	`utl.comments. create_comment (blogid, entryid, userid, content)`
Count comments	Returns the number of comments in a blog entry	`comments`	`utl.comments. count_comment (blogid, entryid)`
delete comment	Deletes a comment from the database	`comments`	`utl.comments. delete_comment (blogid,entryid, commentid)`
delete entry	Deletes an entry from the database	`entries` `entries_arc`	`utl.entries.delete_entry (blogid, entryid)`

[5] All listed functions return a boolean representing the operation's success.

[6] Verifies absence of control characters and uniqueness of name

[7] Uses `**kwargs`: input has to be `edit_acc(name, new_un = “something”` (optional argument), `new_pw = “something else”` (optional argument))

Sitemap


SITEMAP
/
ErnJew
LOGGED IN
LOGGED OUT
/home
/login
GLOBAL NAVIGATION BAR
If users are logged in
/home
/search
/profile
/settings
/logout
/login
FORM SUBMIT
ON CLICK REGISTER
SUCCESSFUL LOGIN?
NO
YES
/home
/register
/home
NEW BLOG
BLOG LINK
/new_blog
/blogs/<id>
/search
BLOG LINK
/blogs/<id>
/profile
BLOG LINK
/blogs/<id>
/settings
FORM SUBMIT
SUCCESSFUL PASSWORD CHANGE?
NO
YES
/home
/logout
/login
/register
FORM SUBMIT
ON CLICK LOGIN
SUCCESSFUL REGISTRATION?
NO
YES
/login
/login
/new_blog
FORM SUBMIT
SUCCESSFUL BLOG CREATION?
NO
YES
/blogs/<id>
/blogs/<id>
CLICK PROFILE
NEW ENTRY
CLICK ENTRY
/new_entry
/entry/<id>
/profile/<user>
/new_entry
FORM SUBMIT
SUCCESSFUL ENTRY?
NO
YES
/entry/<id>
/entry/<id>
MAKE COMMENT
EDIT ENTRY
SUCCESSFUL COMMENT?
YES / NO
CLICK HISTORY
/edit_entry
/history
/edit_entry
FORM SUBMIT
SUCCESSFUL EDIT?
NO
YES
/entry/<id>
LEGEND
GET Request
POST Request
components
authentication