Team Jelly Jam Pancakes: Jacob, Jeremy, Prattay
SoftDev Pd 07
P00 Design Doc
November 2022
Target Ship Date: 2022-11-15

Scenario One: Your team has been contracted to create a collaborative storytelling game/website.

- Users will have to register to use the site.
- Logged-in users can either start a new story or add to an existing story.
- When adding to a story,
    - Users are shown only the latest update to the story, not the whole thing.
    - A user can then add some amount of text to the story.
- Once a user has added to a story, they cannot add to it again.
- When creating a new story,
    - Users get to start with any amount of text and give the story a title.
    - Logged in users will be able to read any story they have contributed to on their homepage (the landing page after login).

PROGRAM COMPONENTS:

Python File-

- app/__init__.py (Website runner):
    - will run the website, serve and retrieve data using Flask
    - render login.html and home.html
    - Get and post request for user logins

- app/db_builder.py (Database builder)
    - will create/edit the database for story/line storing
    - will create/edit a database to store users and passwords
        - this will be checked to determine if a user has already created/added to a story

HTML Files-

- /templates/welcome.html
    - Home page for those entering the website
    - Buttons for logins / registering
- /templates/register.html
    - Page for new users to create their accounts

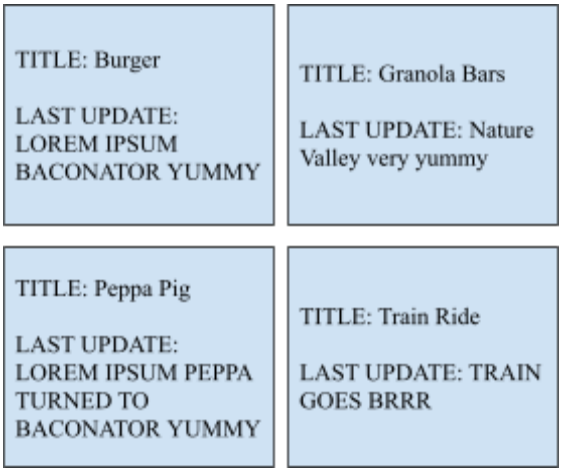
    - Will have UserID form, and PW & retype PW form
    - Will login the user immediately after account creation (?)
- /templates/login.html
    - General login page for registered users
    - Security features TBD
- /templates/feed.html
    - The home page where the user can decide to create a new story, or browse through loaded previews of already created stories
    - Previews may only show the latest update/line added to a story, but will show a user the whole story if the user has already edited the story
        - For a telephone game like story creating process and to follow RULES
- /templates/search.html
    - Searching page when user clicks on search button
    - Probably just to search up titles, tags, and genres (maybe even authors?)
- /templates/create.html
    - A separate page that is accessed from feed
    - Used to create a new story with many different inputs
    - Make users add a title, genres?, tags, and actual story content

- /templates/editing.html
    - Opens an editing window (similar to when you edit a file on github where the user can input new lines to a story

- /templates/stories.html
    - Retrieves data from stories.db and displays the contents of stories onto the webpage

CSS File(s)-

- Display the stories in a social media-like feed, like an Instagram FYP, but instead of posts it is stories and their thumbnails/titles and latest line added.
    - Most upvotes? Most updated/edited? Search feature for genres/titles.

- 

- Will add more detail in later versions

DB Files-

- Logins.db
  - User ID | Password
  - Stored in text
- Stories.db
  - Story ID | Title | Whole story | User IDs | Latest Line added
  - Stored in text

WEBMAP:

"/"

- Root webpage, displays login.html
- Asks to input username and password, or create a new one

"/home"

- Home page after the user logs in, displays previews of stories using feed.html

"/create"

- Creates a new story, uses edit.html

"/<story-title>"

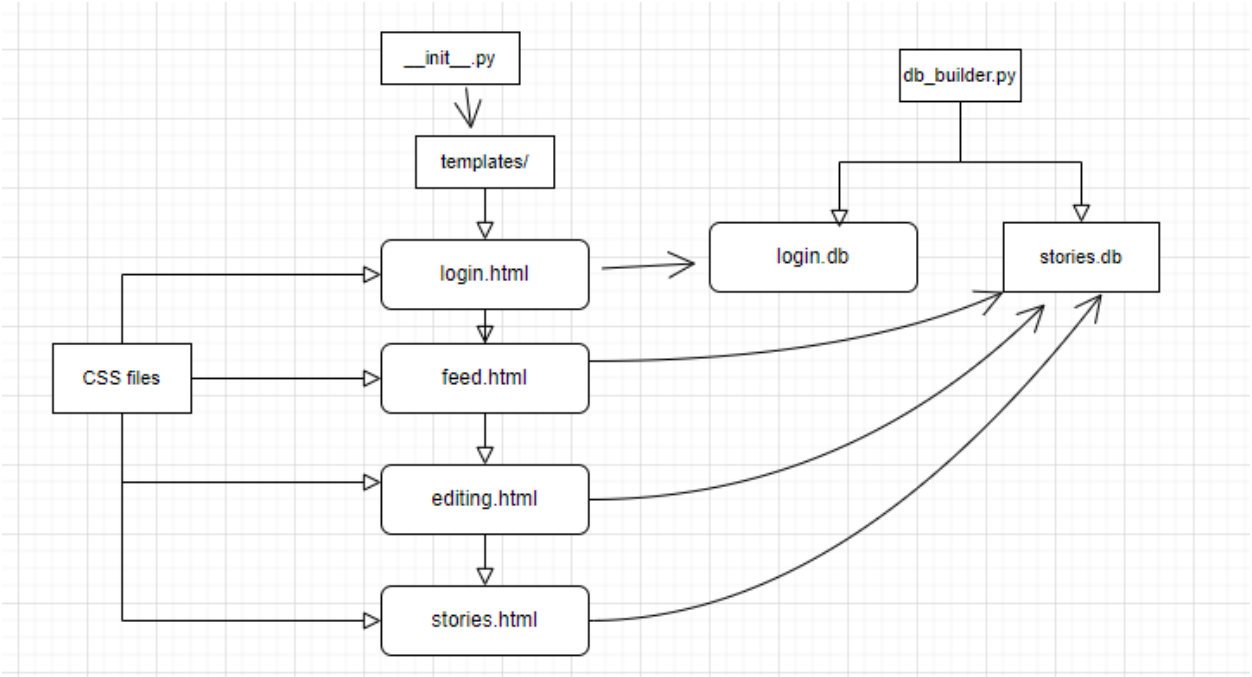
- Displays contents of a story

"/<story-title>/edit"

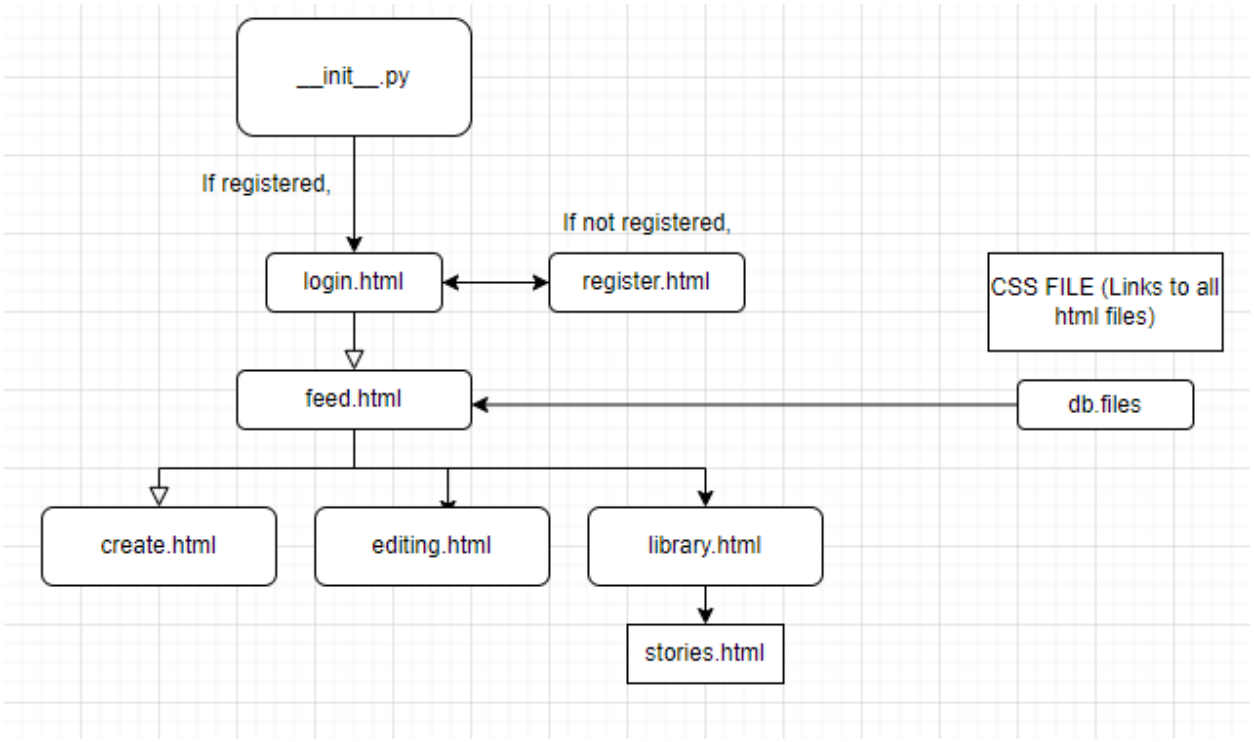
- When a user chooses to edit a specific story that already exists, uses edit.html

COMPONENT MAP:

OLD:



NEW:

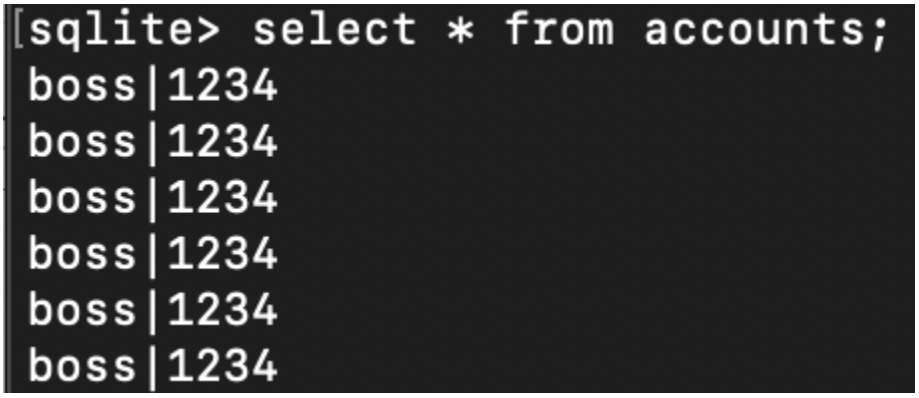


Register:

1. register.html allows new user to input new username
    a. If username is already found in the "accounts" table, then an error message pops up and they pick another name
    b. If username is not taken, then they input a password
2. SQLite stores the new credentials in accounts.table by adding the new entry to the bottom of the "accounts" table
3. Once the user has successfully registered, take them back to the landing page

    Errors:

- *FIXED:* The SQL database keeps storing the same user/password combination as a new entry each time



Log in:

1. login.html allows user to input a username and password combination
    a. If username not found, then error message pops up saying "username not found"
    b. If username found, then check password
        i. If the corresponding password is found, then log in
        ii. If the corresponding password not found, then error message "incorrect password"
2. Login page should have a registration button for new users

Navigating the Feed:

1. Create a story, edit a story, view a story
    a. If users click into an existing story:
        i. Feed.html checks if they have edit records
            1. If they do, then redirect to view.html
            2. If not, proceed to edit.html

Create a Story:

1. createStory.html records username, date-time, story title, text
    a. If the story name is already taken, then tell user to pick another one
2. createStory.html sends data back to the database
3. Database stores storyID, storyTitle, storyContent, username, creation date, creation time information in stories.db

Editing a Story:

1. edit.html searches stories table
    a. Allow user to contribute (template should be same as create.html)
        i. Once a user contributes, edit.html sends the entry (storyID, storyTitle, storyContent, username, edit date, edit time) to stories tables

Feed Page (Structure):

1. Start a New Story
2. Library (Existing stories with most recently edited at top, click story for full view)
    a. Stories the user has edited
    b. Stories the user has not edited
3. Logout

Browsing the Library:

1. Stories the user has edited at the top (sorted by most recently edited)
2. Stories the user has not edited (sorted by most recently edited)

Library Structure:

1. Rediscover
    a. Library searches for stories that the user has edited
    b. For each story, Library adds the most recent entry to exhibits.db
2. Explore
    a. Library searches for stories that the user has not edited
    b. For each story, Library adds the most recent entry to exhibits.db
3. Library displays most recent entry on feed (optional: show original creator)
4. If user clicks on feed then redirect them to view.html or edit.html, depending on whether or not they have an edit record for that story

Database and Tables

1. Accounts (Stores username/password combo)
2. Stories (Stores storyID, storyTitle, storyContent, username, date edited, time edited)

Accounts

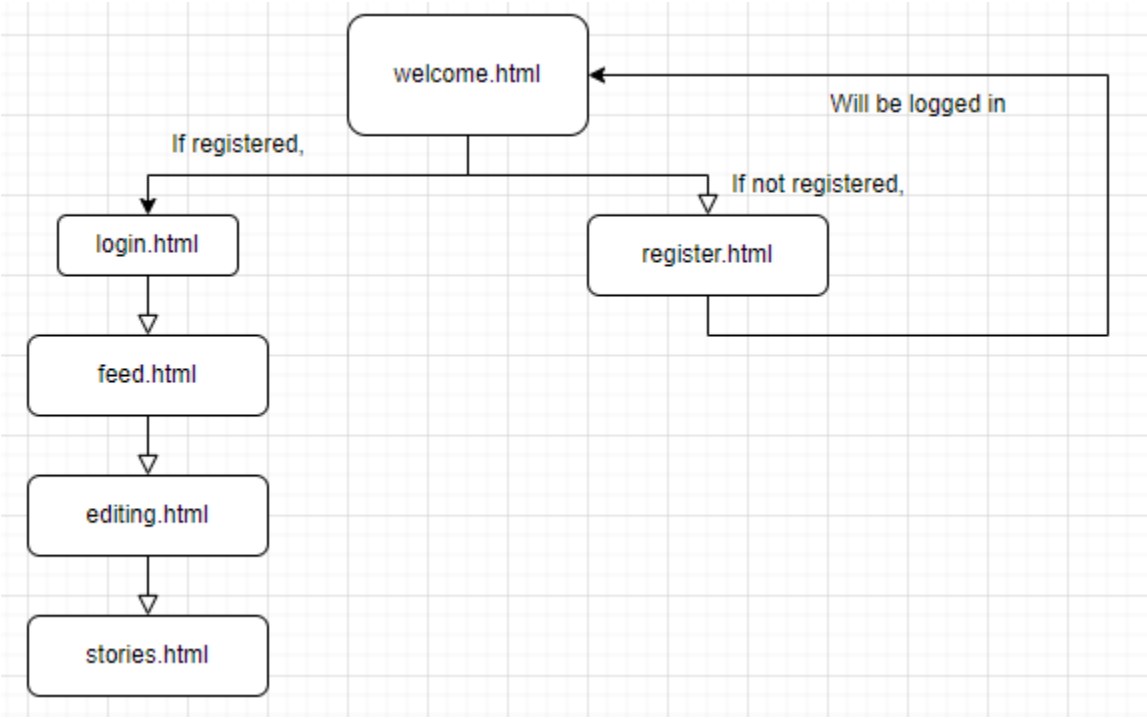
Username	Password
Germy	Jeremy
Pancakes	Syrup

Stories

StoryID	storyTitle	storyContent	username	Date (y-m-d)	Time (H-M-S)
	Foody Guide	“I like food”	Germy	2022-12-25	14-51-09

	Foody Guide	“I drink soup”	Pancakes	2022-12-27	00-13-46

SITE MAP FOR FRONT END:



TASK BREAKDOWNS:
Jerm: PM, will work on SQL and db organization (db_builder
Prattay: will work on Flask and __init__
Jacob: will work and create on the HTML and CSS files

Components of Website:

Priority List:

1. **Register**
2. **Login**
3. **Create Story**
4. Edit Stories
5. Search for Stories