Team fourCoffeePeanuts: Ryan Wang (PM), Eliza Knapp, Yaying Liang Li, Jesse Xie

SoftDev

Design Document

2021-10-28

Program Components:

- Flask
    - Flask will pull html code from the templates folder, use data inputted in python, and display the contents on localhost with Jinja2
    - Flask will accept data from forms to add to sqlite3 tables
    - Flask will keep track of the current session the user is in
- Python
    - Python will import the required modules in order to maintain a functioning website
    - Python will send over relevant data from the database to flask
    - Python will execute commands to create and add to sqlite3 tables with data from flask forms
- SQLite3
    - Sqlite3 will create data tables in a db file which will be stored in a data folder
    - Data about users and stories will be obtained from inputs in forms on the page generated by flask

Database Organization:

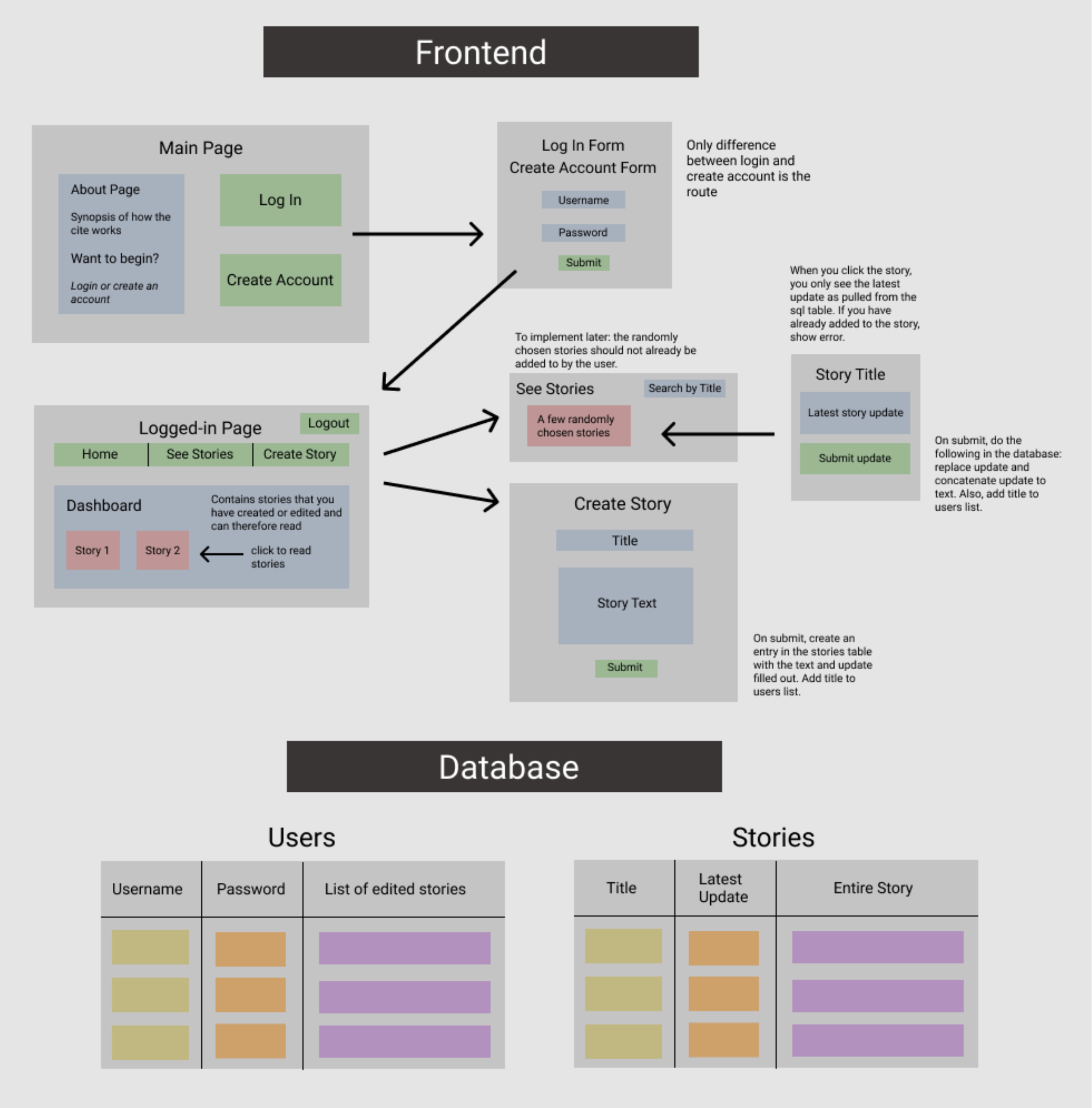
- The username and password combinations will be stored in a table called Users, along with the stories that they have created/edited (username TEXT, password TEXT, stories TEXT)
  - The stories will be stored as text in the following format: “story1, story2, storyN”, where the names of the stories are separated by commas.
- The stories created and edited by users will be stored in a table called Stories. This table will contain the name of the story, along with the latest update to the story and the entire story itself (name TEXT, latestUpdate TEXT, fullStory TEXT)

Tasks:

- Create main page to direct to register/login with buttons (Yaying, Ryan)
- Allow users to register and log in, save user/pass data in Users table (Ryan, Eliza)
  - Prevent users from accessing the rest of the website without logging in
- Create dashboard to allow users to view/read stories they have contributed to (Eliza, Jesse)
- Allow users to create new stories and save new story data from form in Stories tables (Jesse, Ryan)
- Allow users to find existing stories and edit them, modify data in Stories table accordingly with data from form (Jesse, Yaying)
  - Prevent users from viewing anything but the latest update

- Prevent users from making edits to stories they have already contributed to

Figma:



Only difference between login and create account is the route

When you click the story, you only see the latest update as pulled from the sql table. If you have already added to the story, show error.

To implement later: the randomly chosen stories should not already be added to by the user.

Contains stories that you have created or edited and can therefore read

On submit, do the following in the database: replace update and concatenate update to text. Also, add title to users list.

On submit, create an entry in the stories table with the text and update filled out. Add title to users list.

Users

Username	Password	List of edited stories
🟨	🟧	🟪
🟨	🟧	🟪
🟨	🟧	🟪

Stories

Title	Latest Update	Entire Story
🟨	🟧	🟪
🟨	🟧	🟪
🟨	🟧	🟪

Flask and HTML File Specification

Flask

- *app.py* contains routes: / (main page), login, logout, dashboard, browse (the see story page), update (update existing story), create (create new story)
- *Functions.py* contains any helper functions we might need to process data that is retrieved from the database to render the correct html pages. Some functions might be:
    - Return random story
    - Return random story that isn't on the list of edited stories
    - Return all the stories that are edited
    - Return stories that have words related to a search

HTML (in /templates folder)

- MainPage.html - renders text as shown in figma and contains login and create account buttons.
- LoginCreate.html - this will render the form to either login or create an account. We will pass values into this template from app.py to determine which it will do.
- Dashboard.html - this will render the dashboard with the stories the user has edited which are passed in through app.py. It will have options to create a story or update an existing story through buttons.
- CreateUpdateStory.html - this will render the form to either create a new story or to update an existing story based on a value passed by app.py.