Chalice: Anastasia Lee, Andy Shyklo, Jack Blair, Jady Lei
SoftDev
P00
2024-10-23
TARGET SHIP DATE: 2024-11-04

Scenario One: Your team has been contracted to create a collaborative storytelling game/website, with the following features:

- 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.
        - Limit roughly to a sentence, 20 words.
- 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).
        - Potentially include the ability to view the story as plaintext, or with highlights that allow for the viewing of contributor and similar data. This could be one of several QOL life features involving sorting and user interaction.

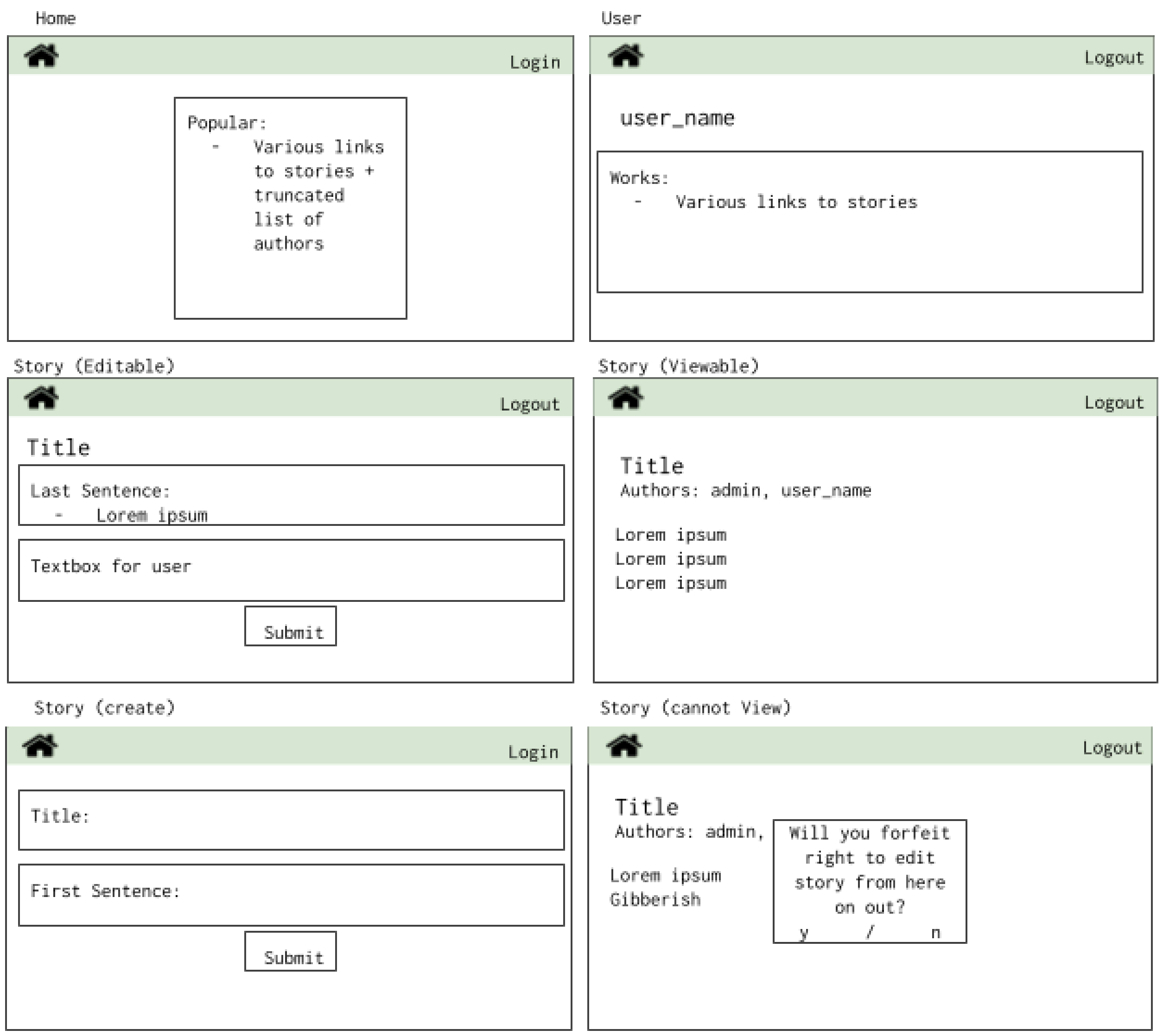
Program Components:

- Login/logout, register for new users
    - Admin can view and edit everything
        - Useful for debugging & testing
        - Allows for moderation of potentially harmful or useless content
    - Regular users:
        - Can only view the most recent edit while editing
        - Can only view a whole story after they edit it
- Stories:
    - List of story titles and IDs on homepage
    - Ability to add stories
    - Ability to edit stories
    - Ability to view stories
        - Show edit history
- Extra Features:
    - Ability to comment on texts

Front-end(HTML, CSS, Displaying info):

- **Home Page**
    - NavBar: Site logo (links to homepage), login/logout
    - List + preview of hosted content (of story links)
- **User Account Page**
    - Displays username, contributions,

- **Story (Viewing)**
    - Prompt to view + title, authors, and text
- **Story (editing)**
    - Last update, textbox, and submit button



Back-end(Flask Nav/Requests):

- Login information stored or retrieved via flask session request using POST. If the account doesn't exist, run code to ask if you want to make an account, otherwise retrieve login info from SQL database.
- Specific user data stored with SQL with user ID compared to posts/views, and depending on this status, some stories may be marked as locked (not contributed) or unlocked (contributed).
- Posts have version history, creation date, contributors, and other parameters that will be held in an existing, separate SQL database.

- All redirects to specific parts of the website, and their subpages, are routed via flask render_templates and specific website elements.
- Stories should be updated both locally and globally around the website as it is being changed, this could be done with update requests (needs more research).

Database Organization:

- What NEEDS to be stored?(EVERYTHING LISTED IS A TABLE):
    - login information,
        - Access level(admin or not),
        - Viewable stories,
        - Editable stories,
    - stories,
        - story + story_id(key)
            - “chapters” + chapter_id(key)
        - Chapter metadata
        - list of authors, relative to chapters
        - *Extra: comment, comment metadata, add comment, comment on comments*
- Actions:
    - Account management:
        - User logs in
            - Check username in login table and authenticate user.
        - User registers new account
            - Adds new username and password to login table, session logged in.
    - User adds to a story
        - Check if user can add to story
            - Select most recent update to display
            - Make sure that updates are synchronized
        - Take request and enter it into the story
        - Store contribution metadata
        - Add story to user’s viewable
    - User creates a story
        - Create entry in stories with title
        - Store contribution + its metadata
        - Add story to user’s viewable
        - Adds story to all users’ editable
    - User views a story
        - Check if viewer can view the story
            - If not prompt viewer to forfeit right to edit
                - Add story to user’s viewable
        - Select all story contributions and display to viewer

Username	Password	Access	Viewable	Editable
admin	******	Admin	Full list	Full list
user_name	******	Normal		

Story ID	Story name
1	abc

Story ID	Story name
2	def

User Database

Story Database

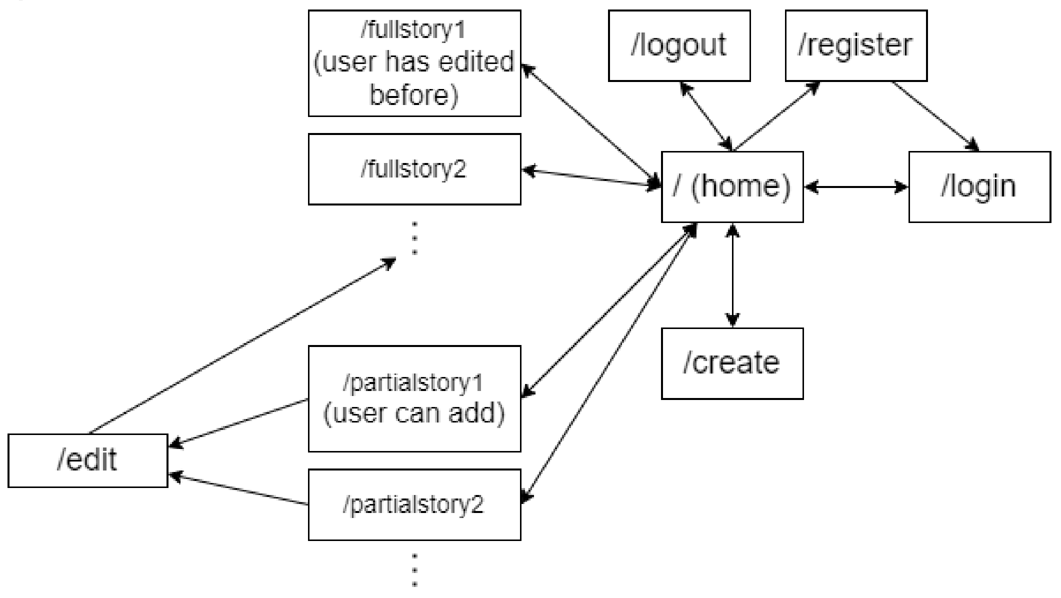
Story ID	Story name	Authors	Chapters
1	abc	[table]	[table]
2	def		

comments?

Order	Author
1	Admin
2	user_name

ID	Content	Contributor	Date
1	“d”	Admin	10/24/2024 12:52 A.M.
2	“e”	Admin	10/24/2024 12:52 A.M.
3	“f”	user_name	10/24/2024 12:53 A.M.

Site Map:



Task Assignments:
Anastasia: PM + Flask person 1
Andy: SQL Database + HTML/CSS
Jack: Flask Person 2?
Jady: SQL Database + HTML/CSS