Scrum PrePlanning - Software Requirements

Home page

Front end:

1. View only page, clickable functionality to sign up/login

Back end:

1. List of categories (top 4-6)

Data storage:

1. Product category

Login

Front end:

2. Edit view (username, password)
3. Session authorization?
4. Forgot password?

Back end:

1. Post login (Body Require: username and password) (return: token , userId)

Data storage:

1. Validating Username, and password

Signup page

Front end:

1. Edit view (name, location, age, gender, password)

Back end:

1. Put register(Body Require: username, password, full_name, address, age, gender) (return: none)

Data storage:

1. Store username

2. Store password
3. Store full_name
4. Store address
5. Store age
6. Store gender

Product list

Front end:

1. search (location, category, text)
2. display list, each item includes:
    a. title
    b. description
    c. thumbnail image (if available)
    d. location (City, State / online)
    e. post type (donation/request)

Back end:

1. Get items
   (Query Parameter: rage [e.g 1-10])
   (return: array of item [refer to front end 2])

Data storage:

1. Get item from table

Product item

Frontend:

1. More information about the item (description, more pictures, date posted).
2. Request
3. Show info about the owner? (username/first name?)

Front-end: 1. Restore the item.
2. API - userId , item id

Backend:

1. Post request_item (Querry: item_id) (Body: user_id) (return: 200)

Data storage:

1. Store requested_item

Post Item

Front-end:

1. Edit view (name, description, category, location, (upload images)?, condition), type (donation/request)? Decide whether to combine into one form
    a. Post title/Name - freeform text input, string, 64 characters limit
    b. Description - freeform text input, string, character limit
    c. Category (drop down) (add later?) - string
        i. future sprint: decide whether to include ‘Other’, if other is selected, magically a text box opens
    d. Location
        i. Don't want to expose people's addresses online
        ii. City - string, state - string, zip - integer
        iii. Online (digital good)
    e. Condition - string
        i. Dropdown/radio button
        ii. Physical good: like new, good, some wear, etc.?
        iii. Digital
    f. Upload Image (optional)
        i. Types: jpg, png
        ii. Max 3 images
        iii. Limit on size (max 1MB per file?)
    g. Date posted - date.
    h. If the item is a document - upload document (compressed file/ zip, jar).

Back end:

1. Put post_item (Body: name, description, category, location, images, condition, create_time, user_id) (return: 200)
2. Put request_item (Body: name, description, category, location, images, condition) (return: 200)

Data storage:

1. Insert new request (name, description, category, location, images, condition)

Account information:

Profile:

Front-end:

1. API (user id) - name, location, age,

Backend:

1. Get requested_item (Query: user_id) (return: user information)

Data storage:

1. Retrieve requested_item
2. Send appropriate item(s) to backend

My Requests - view list of requests made (name, category)

Front-end:

2. API (user id) - list of requests

Backend:

2. Get requested_item (Query: user_id) (return: array of requested_item)

Data storage:

3. Retrieve requested_item
4. Send appropriate item(s) to backend

My Posts: view list of posts made (name, date, image, description)

Front-end:

1. API (user id) - list of posts (status (received **X requests**))

Backend:

1. Get posted_item (Query: user_id, rage) (return user posted_item (number of requests))

Data storage:

1. Retrieve posted_item
2. Store posted_item

Potential Receivers List of the post.

Front-end:

1. List of users (receivers) - name & message?
2. Click button (Approve) - API to backend (receiver id, user id).

Backend:

1. Get requester (Query: item_id) (return: list of requester on an item)
2. Post: approve (Query: item_id, user_id)

Data storage:

1. Retrieve list
2. Send list to backend