02/02/2018

Note Taker: Ruochen Jia

<u>**Topics Discussed**</u>

During the meeting, the team discussed the user stories to be completed in sprint 2, the necessary queries required for this, and the division of labour in setting up dynamic content for each *.ejs* page.

<u>**Queries**</u>

1. Top_ten_questions([“newest”, “views”, “votes”])
    a. Returns an object with properties
        i. Question_title
        ii. Username
        iii. Num Votes
        iv. Num views
        v. Answers
        vi. Date
        vii. Time
2. Get_question_from_search(search_string, sort_by=[“newest”, “views”, “votes”])
    a. Returns an object with properties
        i. Question Title
        ii. Question id
        iii. Username
        iv. Num votes
        v. Num views
        vi. Date
        vii. Time
3. Get_forum(question_id)
    a. Returns an object with properties
        i. Question Title
        ii. Question Body
        iii. User asked
        iv. Question points
        v. Answers
        vi. User answers
        vii. Score
        viii. Date
        ix. Time
4. Store_answer({user_answered: int, answer_body: “”, q_id: int})
    a. Stores an answer for a question, increment the number of answers
5. Store_question({user_asked: int, q_title: “”, q_body: “”, tags: [“”, “”, ..., “”]})
    a. Stores a question

6. Get_user_info(user_id)
    a. Returns an object with properties
        i. Username
        ii. First name
        iii. Last name
        iv. Number of questions answered
        v. Number of questions asked
7. Get_user_questions(user_id) – gets all questions that a user has asked
    a. Returns an object with properties
        i. Question ids
        ii. Question titles
        iii. Dates
        iv. Times
        v. Number of views
        vi. Number of answers
        vii. Number of votes
8. Get_user_answers(user_id) – gets all answers that a user has submitted
    a. Returns an object with properties
        i. Question ids
        ii. Question titles
        iii. Answers given
        iv. Dates of answers
        v. Time of answers
9. Add_new_user({email: "", first_name: "", last_name: "", password: "", gender: [male, female, other], country(optional):""})
    a. Return an object with properties
        i. If email is already in use
            1. {message: "failed due to email in use", failed: true}
        ii. Else
            1. {message: "successfully signed up", failed: false}
10. Attempt_logic({email: "", password: ""})
    a. Return an object with properties
        i. If email exists and password is valid
            1. {message: "Logged In", success: true}
        ii. Else
            1. {message: "E-mail and password don't match", success: false}

<u>Other Tasks</u>

1. Make an *About* Page
2. Add *About & Contact* links in the footer
3. Add a map to the *About Page*

For each page's form, the backend nodejs should be able to capture the form object and display the entries to the console (for possible debugging later).

1. All forms must be displayed via (Console.log(request.query (for GET) / request.body (for POST)))
2. In order to get this functionality, each form must have their respective nodejs express.js listener in the form of app.get(…) or app.post(…). Please do a google search to find this functionality

Separation of tasks

1. All members must include header.ejs and footer.ejs into their allocated pages
2. Home.ejs -> Jason
3. Forum_page.ejs -> Anthony
4. Ask.ejs -> Ruochen
5. About.ejs, contact.ejs -> Alex
6. User_profile.ejs -> Ruochen, Anthony
7. Log-in.ejs -> Saleh
8. Sign-up.ejs -> Abishand
9. Search.ejs -> Tom