CIS557 Project
Group-Based Social Network Web APP

Summary:

You will implement a group-centric social network (similar to Reddit).

- Users can create a public or a private group. Users can only post content into a group.
- Users can request to join a public group or can be invited into a group
- Users can only be invited to join a private group
- Only the group administrators can exclude and invite members to a (private) group
- The group creator is automatically the group administrator and can make other members administrators
- An administrator can promote other members as administrators or revoke the administrator status of another member
- A group has topics (or tags) that must be provided when the group is created.
- Each group has its board (or feed) where members can post text, audio, image, video content
- Users can belong to 0 or more groups
- Users can only have a private conversation (chat) with people (one or many) with whom they share at least one group
- The app must list all the public groups and users should be able to filter groups based on their tags/topics
- The main view should list all the groups that the user belongs to
- Members can flag a post for deletion
- Only group admin can delete other members posts
- A member can always delete their own post

Setup: GitHub Project

We will use GitHub projects and **Extreme Programming** (XP) methodology when implementing this project. As a reminder, XP provides 29 simple rules to be followed in terms of **Planning**, **Managing**, **Designing**, **Coding**, and **Testing.**

1. You will create and configure a **project in your GitHub repository**. (see useful links below).
2. You should create a **wiki page in your GitHub repository**; it will be used to document the project (see the documentation section below).
3. Each user story should be listed in the **GitHub's tracker** as an **issue**. You must label your issues and assign them to specific member(s) of your team

Implementation: Features

You will implement the following features (see table below).

- It is up to you to define how you will implement them and the details of the user interface.
- We encourage you to write user stories for each feature and must provide your user points for each user story.
- The number (level) in parentheses is used for grading.
- Each feature must be fully tested: unit tests, functional tests, integration tests.
- You must use Jest and cypress for testing, and Travis for continuous integration
- Your unit test should achieve the highest code coverage possible
- You will use MongoDB or MySQL as a database engine
- You will use Express as your web server (on top of Node.js)
- Your implementation will use a RESTful API for frontend/backend communication
- You will use ReactJS as framework to build your app
- Check with the course staff before using a 3rd party software, packages, or modules when implementing your apps
- You should explain your design decisions on your GitHub wiki page
- You should make sure that your web application does not crash during usage/testing
- You must deploy your app on Heroku or AWS
- Each team member should contribute to the project. GitHub contribution will be used when grading the project to assess individual contributions

All entries (posts) and messages must support image, audio, or video attachments

#	Features
1	User registration(0)
2	Login/Auth (0)
3	User profile page(0)
4	Create a public group (0)
5	Create a private group (0)
6	Add / remove administrators (0)
7	Request and join a public group (0)
8	Invite a user into a group (public and private) (0)
9	Leave a group (0)
10	Filter groups by topics (tags) (0)
11	Post into a group (0)
12	Flag a post for deletion (0)
13	Delete a post (0) author and admins
14	Show list of public groups (0)
15	Display posts in a group (0)
16	Comment (reply) on a post (0)
17	Interactive REST API documentation using Swagger (0)
18	Hide a member's post (1)
19	Edit/Delete a comment - author (1)
20.0	Send/receive private text message (1) - asynchronously

20.1	Send/receive private audio message (1) - asynchronously
20.2	Send/receive private image message (1) - asynchronously
20.3	Send/receive private video message (1) - asynchronously
21	Provide several ways of sorting groups in the main view: newest message, number of posts, number of members (2)
22	Support for Hashtag filtering in posts and comments (2)
23	@mentions in posts and comments (2)
24	Groups suggestions (3)
25	Live update (3) - posts / messages
26	Notifications(3) - posts / messages
27	Delivery / Read receipts for messages (3)
28	Analytics for groups (4)
29	Pagination (4)
30	Infinite scroll (4)

Validation:

- Your JavaScript must be clean, readable, and **ESLint error and warning-free.** For this assignment, we will **use the Airbnb Javascript style.**
- Your HTML file(s) must pass validation at http://validator.w3.org.
- Your CSS files(s) must pass validation at http://www.css-validator.org/.

Grading:

You should schedule a demo with your TA during exam week.
Assuming that each feature is implemented completely (description of US and issues in GitHub project, tests, implementation, and deployment) and that all user stories within each level are implemented, your work will be graded as follows:

Implementation:
You must implement:

- **All level 0 and 1** features
- **2 level 2** user stories
- **3 level 3 or 4** user stories

For a feature to be considered complete:

- The code must be thoroughly tested (unit, functional) and aim at the highest code coverage level. We will not accept code coverage below 50%. Your Travis-CI interface must display the code coverage
- The feature must be deployed on the cloud. We will not grade features not deployed
- You should comment your code https://code.visualstudio.com/docs/languages/javascript#_jsdoc-support

Project management

- You must define your milestones (sprint)
- All your features/tasks must be linked to an issue in GitHub
- Your issues must be part of a milestone and have labels and assignees https://guides.github.com/features/issues/
- You must use the proper GitHub flow https://guides.github.com/introduction/flow/
  You will get continuous (weekly) feedback from your TA for this category
  You must not fail more than 30% of the iterations (weekly grades)

UI Design

- You should wireframe your entire app
- You must prototype at least 4 user stories (in addition to registration and login).

Documentation

You will use the **Wiki** to document your project: Below is a sample table of contents

- **Design (View)**
    - Interface design
        - Include links to your wireframes and prototype(s)
    - Describe your main React components
    - You will use JSDoc to generate a documentation website
- **REST API**:
    - To be implemented in swagger. Include a link to your Swagger document
    - You will generate your swagger documentation and push it to your GitHub repository

- **Interface (Controller)**: all the (non-helper) CRUD functions you are implementing in your controller
    - You will use JSDoc to generate a documentation website
- **Database (Model)**
    - *Entity-relationship model*
        - You will list all your *entities and their attributes*
        - You will list the relationships between your entities and their attributes (if applicable)
        - You will provide your *Entity-relationship diagram with the cardinalities of the relationships*
    - Translate your *Entity-relationship model* into a *relational schema*
        - You will list all your tables, identify primary and foreign keys
    - Provide a *NoSQL* version of your Entity-relationship model
        - You will list all your *Collection(s) and documents*
    - Create a SQL script file that will contain SQL queries to create and populate your database. *MySQL teams only*
        - You can use the *mysqldump* client to perform this task
    - Create an export file of your MongoDB instance database. *MongoDB teams only*
        - You can use the *mongodump* module to perform this task

- **Security**

    - For each feature listed below:

        - describe the approach you used or the decisions you made (for example password must be at least X characters long and contain at least one special character).
        - List any library, package, or framework you used
        - Where it is implemented (filenames)
    - Features
        - Access control
        - Input validation
        - Account lockout policy
        - Specific HTTP status code
        - Exceptions Handling
        - Secured random token
        - Prepared SQL Statements with parameterized queries (SQL DB only)

Disclaimer:
This is a live document and changes may be made in the future.