CIS557 Project
Microblogging, Messaging and Streaming Social Network Web APP

(Twitter + Live Streaming)

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** listing and describing your user stories and story points.
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.
- You will write user stories for each feature and must provide your user points for each user story.
- The number (level) in parentheses are used for grading.
- Each feature must be fully tested: unit tests, functional tests, integration tests.
- You must use Jest for testing, Selenium for automation, and Travis for continuous integration
- Your unit test should achieve the highest code coverage possible
- You will use MongoDB or MySQL as database engine
- You will use Express as webserver
- 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 (like JQuery) when implementing your apps

- You should explain your design decisions in 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
- Each team member should contribute to the project. GitHub contribution will be used when grading the project to assess individual contributions

Microblog 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	Follow/unfollow user (0)
5	Block a follower (0)
6	Post a new microblog / post (0)
7	Delete a microblog (0)
8	Display posts (microblogs) of “following” contacts (0)
9	Comment (reply) on a post (0)
10	Hide a contact’s post (0)
11	Edit/Delete a comment (0)
12	Interactive REST API documentation using Swagger (0)
13.0	Send/receive private text message (1) - asynchronously
13.1	Send/receive private audio message (1) - asynchronously

13.2	Send/receive private image message (1) - asynchronously
13.3	Send/receive private video message (1) - asynchronously
14	Start a new live stream (1)
15	Display how many people are watching a live stream (1)
16	Comment on a live stream (1)
17	Show list of live streams (1)
18	Join (merge) a contact's live stream (2)
19	Comment on a merged live stream (2)
20	Support for Hashtags filtering in posts and comments (2)
21	@mentions in posts and comments (2)
22	Contact suggestions (3)
23	Live update (3) - posts / messages
24	Notifications(3) - posts / messages
25	Delivery / Read receipts for messages (3)
26	Analytics for posts (4)
27	Analytics for live streams(4)
28	Pagination (4)
29	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 the last week of class.
Assuming that each user story 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 levels 3 and 4** user stories

For a feature to be considered completed:

- 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
- All your features/tasks must be linked to an issue in GitHub
- Your issues must be part of a milestones 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 1 user story. Except the ones related to features # 1 and 2.

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
- **REST API**:

  To be implemented in swagger

- **Interface (Controller)**: all the (non-helper) CRUD functions you are implementing in your controller
    - A table with the following headings
        - Function name
        - Parameters and their types
        - Purpose of the function
- **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.