Design Document

Team 30

Anqi Yu, yu208@purdue.edu

Purpose

In an age of information overload, we're inundated with social media and ceaseless flow of news with questionable contents. In Modesty web app, users can take control over the sources of media contents as well as how much time they want to spent on it. To achieve better balance between living in seclusion and plugging in social networks for hours, my app will let users to input their choice of media outlets and social accounts that they genuinely want to follow. By having users set their own daily time limit, I hope users can regain a sense of agency around the matter of media consumption.

Functional Requirements:

- As a user, I would like to create and log into an account.
- As a user, I would like to input source of media feed from twitter user accounts.
- As a user, I would like to input source of media feed from my Pocket account.
- As a user, I would like to set my daily time limit.
- As a user, I would like to be notified when my media consuming time exceeds specified time limit.
- As a user, I would like to be able to set a new password.
- As a user, I would like to be able to retrieve my password if I have forgotten it.
- As a user, I would like to be able to delete my account.

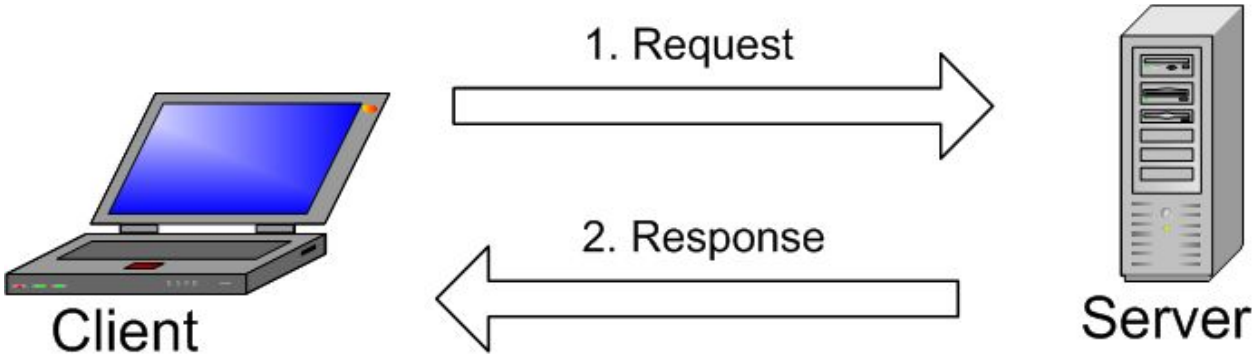
Non-Functional Requirements:

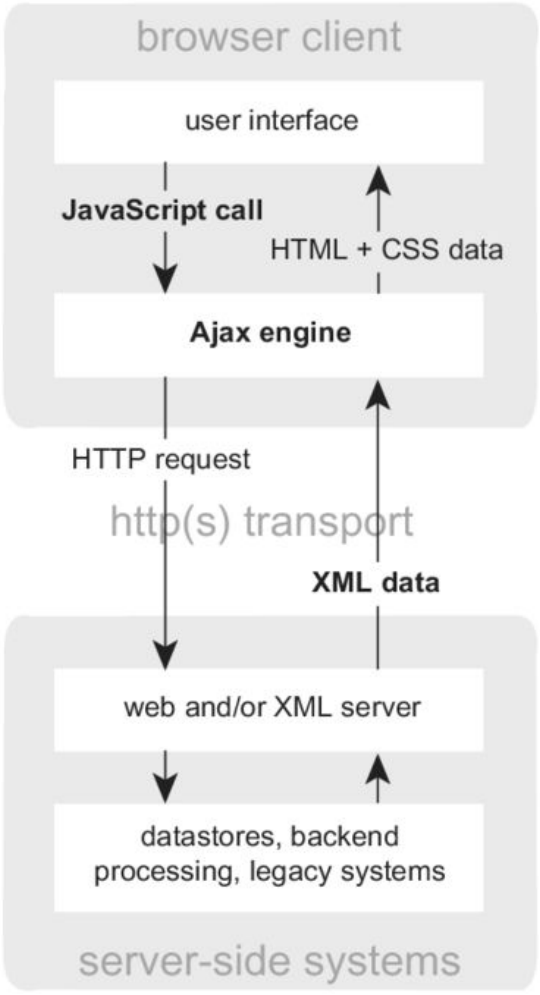
- This app works on web browser on desktop as well as cell phones (responsive web design).
- The user interface should be simple and clear.

Design Outline

Design Decisions and Components:

Modesty app will use a basic Web Client-Server model. Server will be a connected to a database for read and write operations. Web client will be talking to server through HTTP request, to be more specific, Ajax would be used to help send and retrieving data asynchronously.





Ajax
web application model

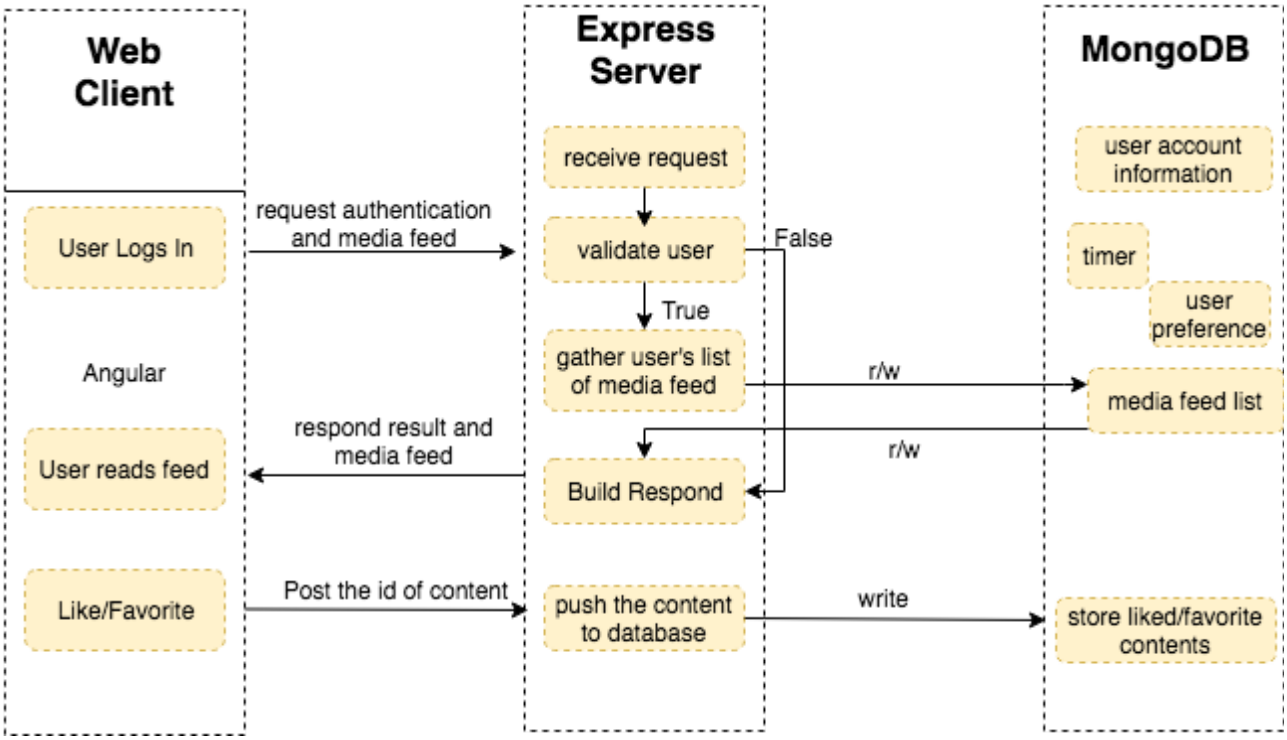
Client-Server Interactions:

When client logged in successfully, server will record the timestamp and send the media feed to client. When the time limit is up, user will receive a notice. And no more media feed is available. When user liked a post, client will send the id of the post to the backend. All system setting and user's preference should be stored and updated on the database.

High Level Overview

There are three components in our client server model in modesty web app.

1. Client
    a. Runs on web browser
    b. Request and respond to the server
    c. Data received from the server will be formatted and presented in the form of media feed, i.e., the author of this content, avatar if provided, media content, a “like” button.
2. Express Server
    a. Receives client requests
    b. Verify clients’ requests
    c. Processing users media feed into a list
    d. A timer that keeps track of users’ active on-site time
3. MongoDB
    a. Stores user’s media feed as a list
    b. Store user’s likes, timer, and preferences
    c. Fetch user’s media feed list to server



Design Issues

Functional Issues

Issue 1: Does user need to create a account in order to log in?

- **Option 1:** User need a new account to login just in case user's facebook account or twitter account changes or deactivated.
- **Option 2:** User can login with their google account. One less username and password to keep track of.
- **Decision:** Option 1. User need to sign up the first time they log in. Their browser can remember their account. In case they forget their password, user can reset their password with the email address they used to sign up with.

Issue 2: Does user need to authenticate their Twitter/Facebook account?

- **Option 1**: For all the social networks user want to input, they have to get authenticate from the platform's API. This is the only method to retrieve contents from protected accounts.
- **Option 2**: Only allow public contents that does not requires user authentication to be fetched. This will speed up the development process.
- **Decision**: A mix of both options. I would start the develop process with the web crawler retrieving only public contents without authentication. After the front end client has it's basic functionality tested, I would add the authentication for each of the linked social account.

Issue 3: Should user interact with others in the social media feed ?

- **Option 1:** no interaction, this app would be a read only app; The purpose of this app is to limit time spent on social media. Having back and forth conversation online is time consuming.
- **Option 2:** limited interaction, user can "like" a post.
- **Option 3:** full interaction if user is authenticated with that social network.
- **Decision:** Option 2. This is a prototype app that has only one developer. Taking the workload into consideration, having a "like" function and no other interaction is a feasible design choice. If time allow, user's liked content can be tracked in the backend for analysis. Or even have a NLP picks out the most likely to be "liked" feed and served at the top of the timeline.

Nonfunctional Issues

Issue 1: What kind of database should be used?

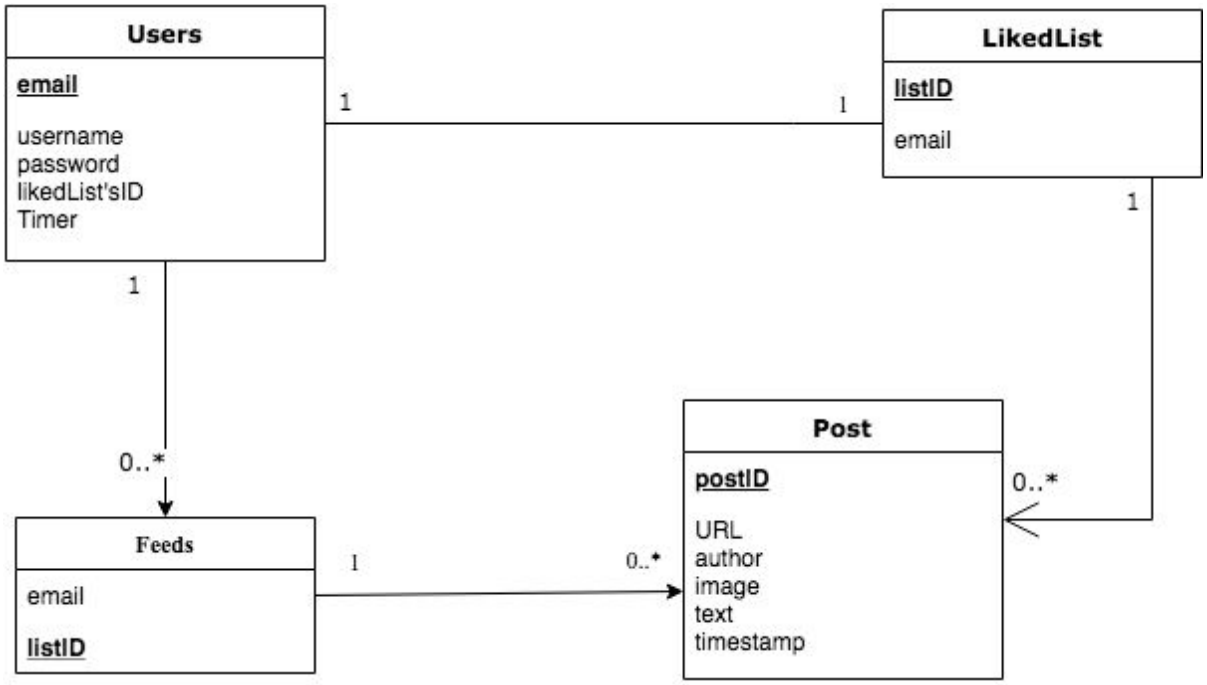
- **Option 1:** MongoDB
- **Option 2:** Redis
- **Option 3:** mySQL
- **Decision:** Option 1. Although Redis is fast and can be used as cache to improve performance, this app does not have APIs that need to be called 10 times per user per seconds. Since most of the data need to be stored is in JSON form, relational database would not be of much use.

Issue 2: What kind of framework should be used for the server?

- **Option 1:** NodeJS
- **Option 2:** Django
- **Option 3:** Express.js + plugin
- **Decision: Option 3.** While both Node JS and Django are brilliant web framework, this project requires fast development and possible major system design change after sprint 1. Express can user used as a standalone server one its own, and add plugins or python script as child process if needed.

Design Details

Class Diagram

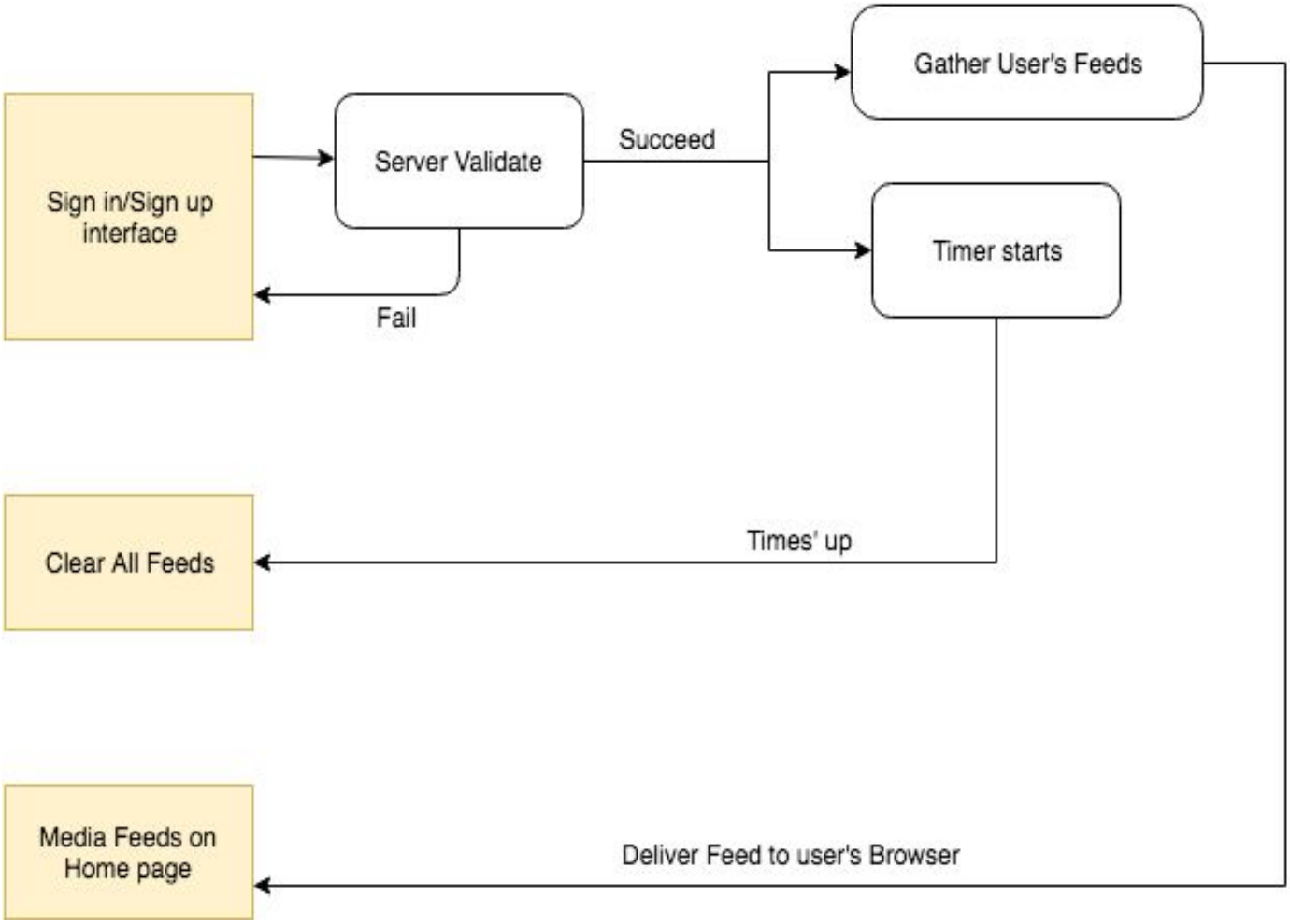


Description and Interaction of Classes:

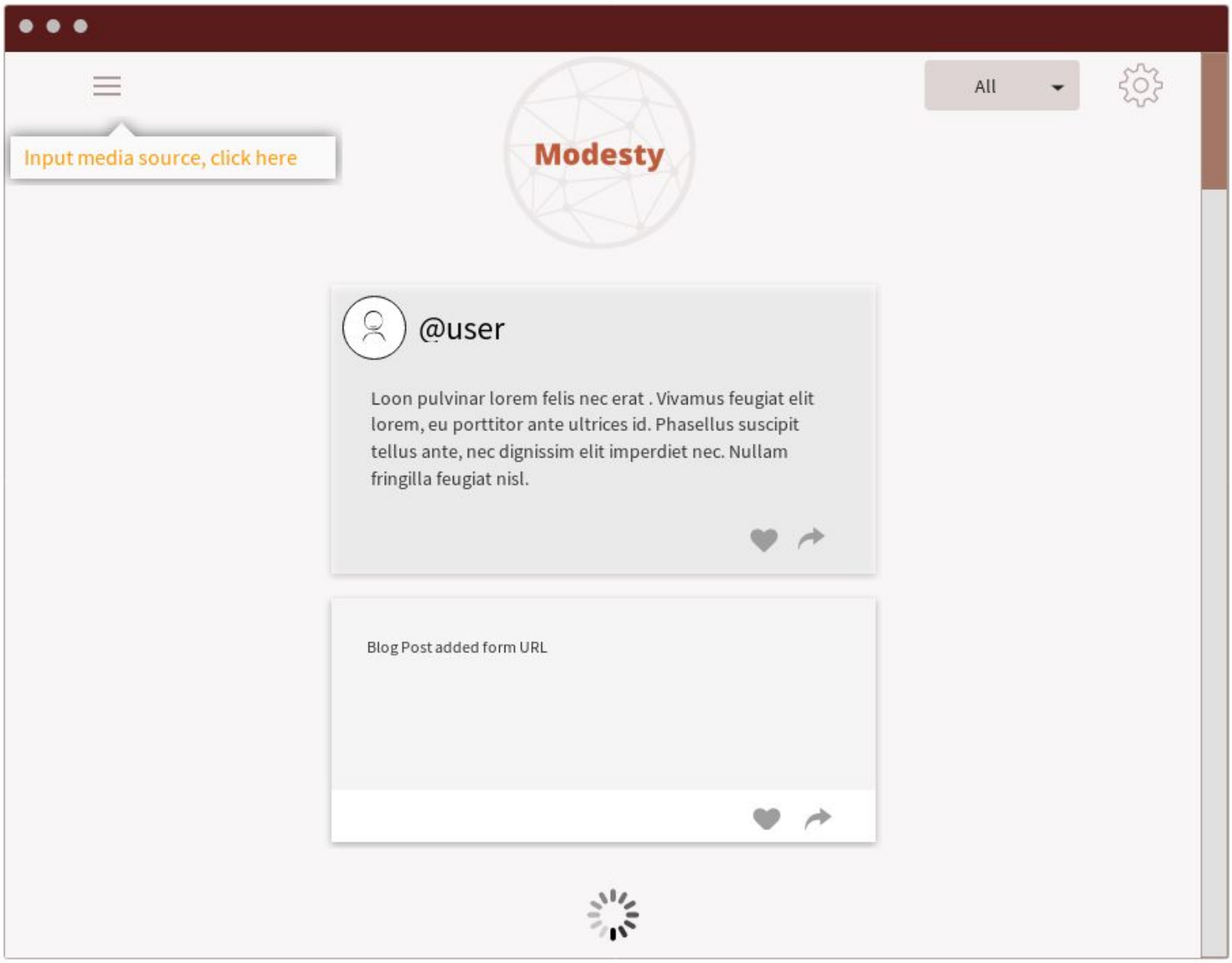
- User
    - Users will be uniquely identified by their email
    - Username and password can be duplicated
    - Each user can have only one LikedList where all of the liked post is stored
- LikedList
    - Each list is uniquely identified by listID
    - 0 or more posts can be stored under one listID
- Post
    - Each Post has its own unique ID in the database
    - Post includes the author, images, text, timestamp, or it URL
- Feeds

- Each user has only one feeds
- Each feed is uniquely identified by its listID
- One feed list can have 0 to many posts

Sequence Diagram



UI mockup


All
Input media source, click here
Modesty
@user
Loon pulvinar lorem felis nec erat . Vivamus feugiat elit lorem, eu porttitor ante ultrices id. Phasellus suscipit tellus ante, nec dignissim elit imperdiet nec. Nullam fringilla feugiat nisl.
Blog Post added form URL