Blog Application

Software Design Document

3/15/19

Software Developers:
Kevin Lin
Stanley Lin

Table of Contents

Overview

1.1 Goals and objectives

The goal of this project is to develop a blog web application (representation of current tumblr.com 2019).

The objective is to create a blog web application that allows users to sign up and login using an email and password or facebook account. Users can customize their profile page for visitors. Visitors will be able to see all blog posts by a user unless the user has made their post/profile private. Users can set their entire profile private or restrict certain posts. Restrictions have the ability to make a post viewable only to followers or the user. Users will have the ability to utilize a search bar to search for posts containing keywords. The search bar will search titles, tags, and content.

1.2 Statement of scope

- As a user I want to see blog posts relevant to my preferred interests so that I do not have to look/read things I am not interested in.
- As a user I want to be able to create my own personal profile so that I can track all of my posts, activities, and customize my profile page for visitors.
- As a user I want to be able to search for blog posts using a search bar so that I can look for blog posts I am interested in (matching tags, post titles, and writing).
- As a user I want the ability to reset and or request a password change for my account in case my password has been compromised, forgotten, or any other reason.
- As a user I want to be able to follow other users so that I can view all of their current posts and future posts.

- As a user I want to be able to make my posts private so that only I can see a certain post or only allow certain users see this post.
- As a user I want to be able to create blog posts so that I can share them with the public, certain users, or just myself.

1.3 Software context

The end product is a blog web application for users to view and share blog posts.

1.4 Major constraints

No major constraints to mention currently (will keep this updated). Only issue is we will need to do research on software testing ie unit and integration tests. We will look into Jest for unit and integration testing and TestCafe for UI tests.

Data design

2.1 Internal software data structure

- Dynamic Array
- Hashmap

2.2 Global data structure

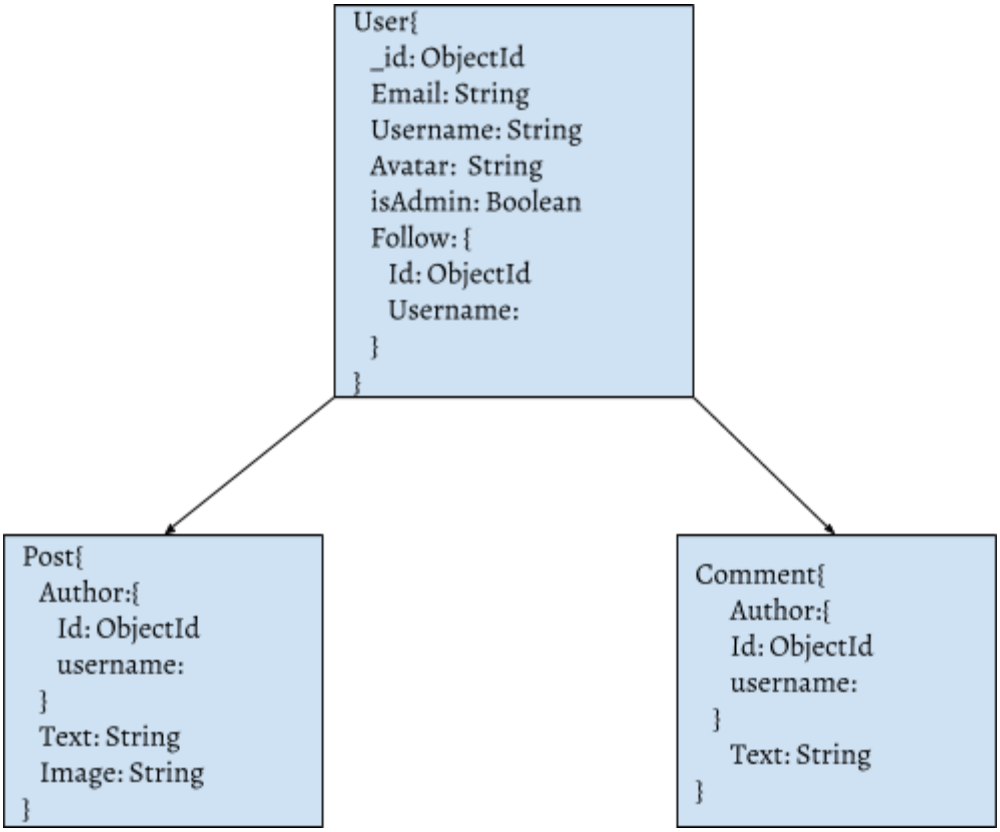
- Dynamic Array
- Hashmap

2.3 Temporary data structure

No temporary data structures will be used.

2.4 Database description

Will use NoSQL database (MongoDB). Will use a one to many relationship between users/authors to posts and comments. One user/author can have many posts and many comments, but one post and one comment can only have one author. Sample database model shown below:

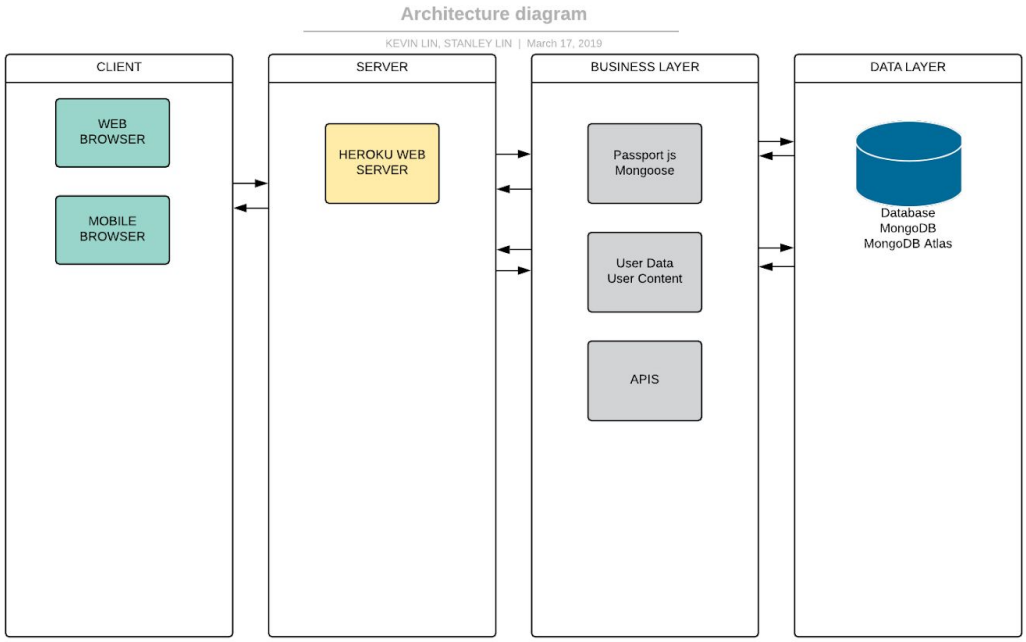


Architectural and component-level design

3.1 System Structure

(Will be updated/reviewed constantly)
Front end: Bootstrap 4, HTML5, CSS, and React.
Authentication & authorization: Passport js, will use facebook and local strategy.
Back end: Node.js and express.
Database: MongoDB.

3.1.1 Architecture diagram



3.2 Description for Web Browser

3.2.1 Processing narrative for Web Browser

The web browser will be an entry point for users to access the application.

3.2.2 Component Web Browser interface description

The web browser will receive inputs from the user via keyboard and mouse (you know like most web applications?). Output from the web browser will be delivered to the user via HTTP requests and rendering HTML. Depending on the input request from the user will result in what is outputted to the user from the web browser.

3.2.3 Component Web Browser processing detail

3.2.3.1 Design Class hierarchy for Web Browser

Not applicable.

3.2.3.2 Restrictions/limitations for Web Browser

Browser limitations on certain CSS or HTML features.

Possible limitations on cross web browser platform support.

3.2.3.3 Performance issues for Web Browser

Performance issues will depend on the browser being used for the application and the user's computer hardware.

3.2.3.4 Design constraints for Web Browser

No design constraints at the moment (will keep it updated).

3.2.3.4 Processing detail for each operation of Web Browser

3.2.3.5.1 Processing narrative for each operation

User enters web application URL into web browser URL search bar. Application loads onto landing page. Displays to user the options to sign up or login. Selecting sign up brings the user to the sign up page, user enters required information and is then redirected to the application home page. Selecting login brings the user to the login page, user is allowed to sign in via email and password or facebook. If credentials are incorrect or any other error occurs, user will be notified and redirected to login page. If login is successful user is directed the application home page.

3.2.3.5.2 Algorithmic model for each operation

Not applicable

3.3 Dynamic Behavior for Web Browser

3.3.1 Interaction Diagrams

Not applicable

3.4 Description for Mobile Browser

3.4.1 Processing narrative for Mobile Browser

The mobile browser will be an entry point for users to access the application.

3.4.2 Component Mobile Browser interface description

The mobile browser will receive inputs from the user via touch screen and on screen keyboard (similar to most web applications running on a mobile device). Output from the mobile browser will be delivered to the user via HTTP requests and rendering HTML. Depending on the input request from the user will result in what is outputted to the user from the mobile browser.

3.4.3 Component Mobile Browser processing detail

3.4.3.1 Design Class hierarchy for Web Browser

Not applicable.

3.4.3.2 Restrictions/limitations for Web Browser

Mobile limitations on certain CSS or HTML features.

Possible limitations on cross mobile browser platform support.

3.4.3.3 Performance issues for Mobile Browser

Performance issues will depend on the browser being used for the application and the user's mobile hardware.

3.4.3.4 Design constraints for Mobile Browser

No design constraints at the moment (will keep it updated).

3.4.3.4 Processing detail for each operation of Mobile Browser

3.2.3.5.1 Processing narrative for each operation

User enters web application URL into mobile web browser URL search bar. Application loads onto landing page. Displays to user the options to sign up or login. Selecting sign up brings the user to the sign up page, user enters required information and is then redirected to the application home page. Selecting login brings the user to the login page, user is allowed to sign in via email and password or facebook. If credentials are incorrect or any other error occurs, user will be notified and redirected to login page. If login is successful user is directed the application home page.

3.4.3.5.2 Algorithmic model for each operation

Not applicable

3.5 Dynamic Behavior for Mobile Browser

3.5.1 Interaction Diagrams

Not applicable

3.6 Description for Heroku Web Server

3.6.1 Processing narrative for Heroku Web Server

Heroku is the server provider for our application.

3.6.2 Component Heroku Web Server interface description

Create a new node.js application on Heroku.

3.6.3 Component Mobile Browser processing detail

3.6.3.1 Design Class hierarchy for Heroku Web Server

Not applicable.

3.6.3.2 Restrictions/limitations for Heroku Web Server

Limitations on what Heroku server can provide.

3.6.3.3 Performance issues for Heroku Web Server

Performance issues will depend on Heroku

3.6.3.4 Design constraints for Heroku Web Server

Not applicable.

3.6.3.4 Processing detail for each operation of Heroku Web Server

3.6.3.5.1 Processing narrative for each operation

Once application is in a state where it can be deployed. We will create a new application on Heroku and deploy the application.

3.6.3.5.2 Algorithmic model for each operation

Not applicable

3.7 Dynamic Behavior for Heroku Web Server

3.7.1 Interaction Diagrams

Not applicable

3.8 Description for Passport js w/ mongoose

3.6.1 Processing narrative for Passport js w/ mongoose

Passport js will be used for our applications user authentication and authorization.

3.6.2 Component Passport js w. mongoose interface description

Install Passport js node package into our application and setup allowing two different strategies local strategy and using Facebook login.

3.6.3 Component Passport js w/mongoose processing detail

3.6.3.1 Design Class hierarchy for Heroku Web Server

Will use mongoose to make creating users in our database with Passport js easily.

3.6.3.2 Restrictions/limitations for Passport js w/mongoose

No limitations.

3.6.3.3 Performance issues for Passport js w/mongoose

No performance issues

3.6.3.4 Design constraints for Passport js w/mongoose

Not applicable.

3.6.3.4 Processing detail for each operation of Passport js w/mongoose

3.6.3.5.1 Processing narrative for each operation

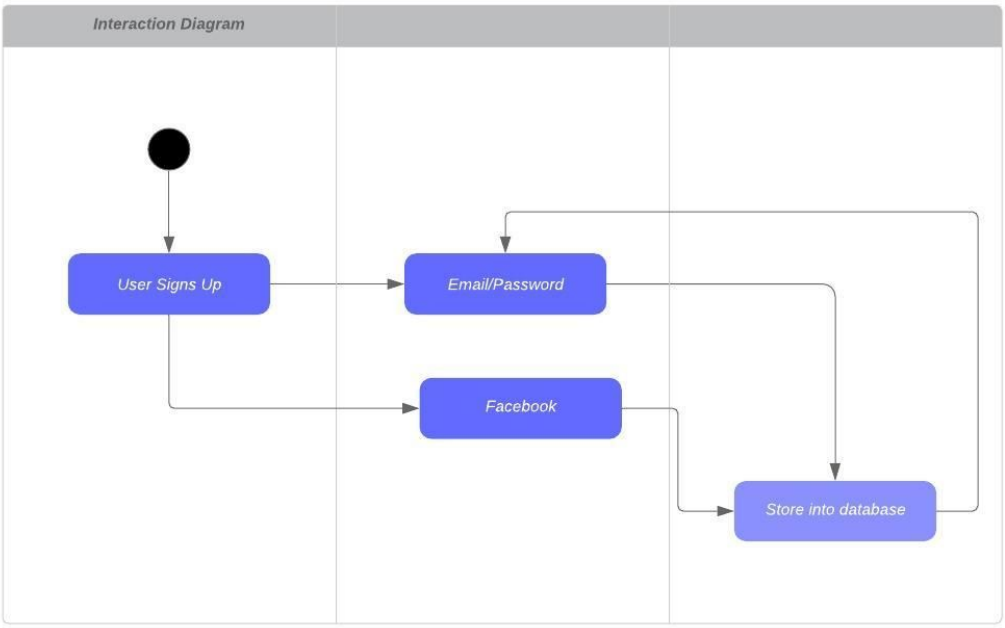
Once user model is setup we will use included functions with Passport js to store user data in mongodb.

3.6.3.5.2 Algorithmic model for each operation

Not applicable

3.7 Dynamic Behavior for Passport js w/mongoose

3.7.1 Interaction Diagrams



User interface design

Restrictions, limitations, and constraints

Testing Issues

Milestones