Team2000s Project Proposal

Brett Jia (bbj2110)
Soorya Kumar (ssk2234)
Zixiong Liu (zl2683)
Mavis Athene U Chen (mu2288)

Part 1:

Our project idea is to create a web application that allows users to post educational blogs of varying topics for their readers. The web application will allow users to register accounts and create their own blogs, allowing for writing/submitting posts, editing posts, and deleting posts. Additionally, we intend to leverage the Twilio API to enable readers to "follow" a user and sign up for push notifications whenever a new blog post is submitted. We anticipate adding support for rich text in the blog posts, and we intend to add support for code and syntax highlighting in the blog body for programming-centric blogs.

Registration and authentication in this application will be implemented with single sign-on and connect to an identity provider such as Google Accounts (exact identity provider to be determined). We believe that this will provide a smoother user experience to blog writers as they will not need to memorize yet another username and password. Additionally, this allows us to use OpenID Connect, which are standardized single sign-on protocols provided by identity providers such as Google Accounts. Session timeout will be set to 1 day (to be finalized) and a logout button will be provided in the application. The readers of these blogs can be anonymous and do not need to authenticate, although if they wish to sign up for text notifications (or unfollow bloggers), they will need to register and authenticate.

The application will also support searching blogs based on topics or keywords. We expect that this search functionality will be powered by Elasticsearch.

Our intended project demo will be on the blog application from the perspective of a blog writer and a blog reader. We will show single sign-on authentication functionality, as well as blog posting, editing, and deleting functionality, and additionally the text-based subscription functionality using the Twilio API.

The blog application server will need to store persistent data about user accounts, user subscriptions, and the blog posts themselves. Blog posts will support embedded photos, which will be stored server-side as well. We plan on using a relational database (MySQL) to store this information.

Finally, the blog application will call out to the Twilio API to send text message notifications to users who have followed a blog writer. The application will leverage Twilio to send out texts whenever a blog writer submits a new post.

If there is ample time, some stretch goals we can implement include:

- A feature for users to add comments on a blog post.
- A newsfeed landing page that aggregates posts from all bloggers that a reader has followed.
- A "suggested bloggers" widget on the landing page that contains bloggers related to bloggers that a reader has followed.
- A periodic text notification containing new suggested bloggers to follow.
- A mechanism for a reader to "block" a certain blogger from showing up in the suggested bloggers list.
- Full deployment of our application on Amazon Web Services with immutable infrastructure built with Terraform, Packer, and Chef.

Part 2:

Blog Subscription: As a blog reader, I want to be notified immediately when my favorite bloggers post a new blog entry so that I can be up to date with what the blog writers have to share.

My conditions of satisfaction are:

- Every time my favorite/followed bloggers submit a blog post, I want to receive a notification.
- For bloggers that I am not interested in, I do not want to receive a notification about their posts.
- If I want immediate updates, I should be able to receive notifications within a minute of the blog being submitted.
- I should receive at most one notification per blog post and not multiple notifications for the same post.
- I prefer to receive a late notification than no notification at all.
- I should be able to modify my notification preferences to determine the frequency of notifications.
- I should be able to unfollow blogs that I am no longer interested in reading.

Blog Post Creation: As a blog writer, I want to be able to create new blog posts with a rich text editor, so that I can create expressive and comprehensive posts that can better convey the educational content I want to convey from my blog.

My conditions of satisfaction are:

- When I am logged into my account I should easily be able to start creating a new blog post using a “Create” button or something of that sort.
- I should be able to save a new blog post as a draft, so that I can return to work on the post at another time if I am not finished with my post.
- Saved drafts that I create should be only visible to me and not to other users.
- I should be able to create my blog post with a rich text editor that allows me to embed photos, include code snippets, and allow for other types of syntax formatting and highlighting common in other applications.
- Once I have completed a blog post to my satisfaction, I should be able to post the blog post so that it is available to all of my followers to read.

Blog Post Editing: As a blog writer, I want to be able to edit blog posts that I have already posted to my followers with a rich text editor if needed so that I have the flexibility to make adjustments to my initial posts and am not locked in when I have made a mistake on initial posting.

My conditions of satisfaction are:
- I should be able to see a dashboard of all the posts I have ever made.
- I should be able to select a post on the dashboard that I want to edit.
- I should be able to use the same rich text editor I used to create the Blog post to edit it.
- Once I press a button like “Post” or “Update”, I should be able to publish my updated post in place of my original post. Any follower that then accesses that blog post should only see the updated post
- I should be able to save my edits as a draft, without posting them, if I don’t want to work on my updates in one sitting.
- The original blog post should still be available to my followers if they access the original post until I have posted my edits/updates. At that point any follower that accesses my post should see the updated version.

Blog Post Deletion: As a blog writer, I want to be able to delete blog posts that I no longer want my followers to be able to see, so that I can remove posts that might be outdated or misleading or otherwise no longer relevant.

My conditions of satisfaction are:
- I should be able to see a dashboard of all the posts I have ever made.
- I should be able to select a post on the dashboard that I want to Delete.
- Once I click on a “Delete” button, that should delete my post from my dashboard, and the application as a whole

- My followers should not be able access my article anymore if I have deleted it. If they have a link to the article that I have deleted and try to use it (perhaps from a notification), then they should be routed to a page that tells them that the article is no longer available.

Blog Post Author Page: As a blog writer, I want to have an author page where my followers can read more about me and see the list of all the articles I have ever written so that potential followers can easily decide if my content is for them and whether or not they would like to follow along.

My conditions of satisfaction are:

- I should be able to see a dashboard of all the posts I have ever made. On this page I should have the option to edit my name as it appears to my followers.
- On my dashboard I should also have the option to set a About me field, so that I can describe myself and the type of content that I like to write for my followers and potential followers to see.
- When a follower or potential follower searches for me in the applications, or reads one of my blog posts and clicks on my name on that blog post, they should be directed to my author page
- The follower or potential follower should be able to ready my “About me” and should be given the option to follow/subscribe to me if they are interested with the click of a “Follow” button.

Part 3:

The general approach to the conduct acceptance testing for our project is using the black-box testing method and more importantly the user acceptance testing. The majority of the testing can be done through the GUI, however, it may change as the project progresses and our implementation is finalized. Most of our testing will be centered around blog posts and how users interact with it, so we expect to perform our acceptance tests manually by clicking through our application’s GUI and following a list of scenarios to test.

Blog Subscription

- Acceptance tests that would correspond to pass:
    - Input would be a user posting a blog and the result expected would be that all its subscribers will receive a notification
    - Input would be a user posting a blog and the result expected would be that all its subscribers will receive a notification in a minute.
    - Input would be a user posting a blog and the result expected would be that all its subscribers would receive notifications corresponding to their preferences
    - Input would be a user updating notification settings and the result expected would be that the desired frequency is applied immediately.

    - Input would be a user selecting an option to unfollow a blog and the result expected would be that the user should no longer be notified of new posts (until the user decides to re-follow).
- Acceptance tests that would correspond to fail:
    - Input would be a user posting a blog and the result is users that didn't subscribe to the blogger received notifications
    - Input would be a user posting a blog and the result is subscribers did not receive a notification.
    - Input would be a user posting a blog and the result is subscribers receive a notification several minutes or hours later.
    - Input would be a user unfollowing a blog and the blog author creates a new post, and the result is the user who unfollowed gets a notification.
- Special cases
    - Subscribers should not be notified when a blog post has been published but the blog author decides to edit the post and repost the same blog post.

Blog Post Creation

- Acceptance tests that would correspond to pass:
    - Input would be user saving the post into drafts and the result expected would be the same exact post at the saved states in the drafts folder
    - Input would be the user created a post and the result expected would be that their followers would be able to view it.
- Acceptance tests that would correspond to fail:
    - Input would be user logging in and the result is there is no way for a blog post to be created.
    - Input would be user saving the post to drafts and the result is that the post is visible to other users too
- Special cases
    - If the user writes a blog that exceeds the character or size limit, the user should be warned and prevented from submitting the blog post.
    - The user should be prevented from editing a post that has already been deleted.
    - The user should be prevented from publishing a draft that has already been deleted.

Blog Post Editing

- Acceptance tests that would correspond to pass:
    - Input is the user at a dashboard, result expected would be all the posts the user has made and a way to proceed to the editing page.
    - Input is the user "Updating" the post, result expected would be that the new edited post will replace the old one.

- Acceptance tests that would correspond to fail:
    - Input is the user “Updating” the post, result is that subscribers still have access to the old post.
    - Input is the user attempting to “Edit” the post, result is that the old content is not available anymore.
- Special cases
    - The user makes unpublished edits to the post but cancels the edits, the original post should still be unchanged.

Blog Post Deletion

- Acceptance tests that would correspond to pass:
    - Input is the user at a dashboard, result expected would be all the posts the user has made are visible and the user has a way to proceed to choose a post to delete.
    - Input is the user deleting a post from their blog, result expected is that the post no longer appears on their dashboard and a follower who tries to access the post through a notification link is routed to a page that says “This article is no longer available” or something similar.
- Acceptance tests that would correspond to fail:
    - Input is the user has deleted a post from their dashboard but the post is still visible on their dashboard
    - Input is the user has deleted a post from their dashboard but a follower is still able to access the blog post by using a link from a notification alert.
- Special cases
    - The user should not be able to delete a post twice.
    - The user should not be able to delete someone else’s post.

Blog Post Author Page

- Acceptance tests that would correspond to pass:
    - Input is any user loading an author page dashboard, result expected would be a page that contains a list of all the posts the user has ever written.
    - Input is an author loading his/her own author page dashboard and clicking on a button next to the author’s name, result expected would be an editable field being displayed for the user to update the author name.
    - Input is an author submitting (press enter or press a button) an edited name, result expected would be that the new author name is propagated through the system and displayed in any subsequent load of the author’s blog posts and dashboard page.
    - Input is an author loading his/her own author page dashboard and clicking on a button next to the author’s “About me” field, result expected would be an editable field being displayed for the user to update the author’s “About me” field.

    - Input is the author submitting (press enter or press a button) an edited “About me” description, result expected would be that the new author “About me” section is propagated through the system and displayed in any subsequent load of the author’s page dashboard.
    - Input is any user loading a blog page and clicking on the author’s name, result expected would be that the user is redirected to the author’s page dashboard.
    - Input is any user loading an author page dashboard that is not his/her own, result expected would be that there is an option displayed to follow (or subscribe) to the author’s blog.
    - Input is any user clicking on a follow (or subscribe) option on an author page dashboard, result expected would be that the option is changed to indicate subscription is successful (e.g. grayed out or text changed to “Subscribed!”).
    - Input is any user clicking on an unfollow (or unsubscribe) option on an author page dashboard, result expected would be that the option is changed to indicate the unsubscription is successful (e.g. grayed out or text changed to “Unsubscribed!”).
- Acceptance tests that would correspond to fail:
    - Input is any user loading an author page dashboard, but the page does not contain the entire list of all the posts the user has ever written.
    - Input is an author loading his/her own author page dashboard and clicking on a button next to the author’s name, but the user is unable to update the author name.
    - Input is an author submitting (press enter or press a button) an edited name, but users of the application see the old (prior to update) author name propagated through the system and displayed in any subsequent load of the author’s blog posts and dashboard page instead of the updated author name.
    - Input is an author loading his/her own author page dashboard and clicking on a button next to the author’s “About me” field, but the user is unable to edit the author’s “About me” field.
    - Input is the author submitting (press enter or press a button) an edited “About me” description, but users of the application see the old (prior to update) author “About me” section propagated through the system and displayed in any subsequent load of the author’s page dashboard instead of the updated “About me” section.
    - Input is any user loading a blog page and clicking on the author’s name, but the user is not redirected to the author’s page dashboard as they would expect to be redirected to.
    - Input is any user loading an author page dashboard that is not his/her own, but the user does not have an option displayed to follow (or subscribe) to the author’s blog.

    - Input is any user clicking on a follow (or subscribe) option on an author page dashboard, but the user receives no feedback from the application that they have successfully subscribed to that author.
    - Input is any user clicking on an unfollow (or unsubscribe) option on an author page dashboard, but the user receives no feedback from the application that they have successfully unsubscribed from the author.
- Special cases
    - An author should not be able to select the follow (or subscribe) option on his/her own author page.
    - A user should not be able to edit the name of an author page that is not his/her own.
    - A user should not be able to edit the "About me" section of an author page that is not his/her own.
    - A user should not be able to unsubscribe from a blog that he/she is not already subscribed to.
    - A user should not be able to subscribe multiple times to the same blog.

Part 4:

We intend to use the following technologies for our project:

- Node.js (runtime)
- MySQL (data store)
- OpenID Connect (authentication)
- Elasticsearch (search)
- Visual Studio Code (code editor) + Docker (development environment)
- NPM (package manager) + Docker (environment)
- Prettier (style checker/code formatter)
- Jest (unit testing/coverage)
- LGTM (bug finder)