Critika

CS 30700: Software Engineering 1
Design Document

Team 8:
Brandon Sung
Connor Todd
Cyrus Santiago
Keith Tan
Ken Sodetz
Julien San Diego

WHAT WENT WELL

In general, we set up a lot of the necessary building blocks for our product to run. We made a lot of important backend routes and features, and most of the UI for the website. Building new features will be a lot easier now with the backend and frontend set up, and we should be able to work faster and implement even more in the following sprints.

User Story #1 - 28 hrs
As a user, I would like to be able to register and verify a Critika account.

#	Task Description	Estimated Time (Hours)	Team	Owner
1	Set up Back End server	3	Back End	Keith
2	Set up Mongo database	2	Back End	Keith
3	Set up deployment server	2	Back End	Keith
4	Create UI form to enter information about user	2	Front End	Julien
5	Create button to send information to Back End to register	1	Front End	Julien
6	Create API route to register user and add them to database	6	Back End	Keith
7	Implement API route to generate a verification token	6	Back End	Ken
8	Implement mailer to send verification token	3	Back End	Ken
9	Create Unit Tests	3	Back End	Connor

Completed:

The servers and database setup took a little longer than anticipated, but they were the first things completed for this entire project. The UI form to sign up and register were done without any issue, and the API routes (/register and /verify-email) were made. The mailer sends an email to the email provided by the user with a verification token that they must use to verify the email, along with a message from the team. The backend has extensive error checking, and the unit tests made check all possible registration issues (register without username, email, invalid password, etc.). All unit tests were passed.

User Story #2 - 13 hrs
As a user, I would like to be able to login to my Critika account

#	Task Description	Estimated Time	Team	Owner

		(Hours)		
1	Create UI form to for entering login information	2	Front End	Julien
2	Create button to initiate login	1	Front End	Julien
3	Create API route to login user and generate authentication token	5	Back End	Keith
4	Allow user to maintain a session on web client	2	Back End	Keith
5	Create Unit Tests	3	Back End	Connor

Completed:

After you register for an account, the obvious next step is to sign in. The sign-in form for the UI prompts a user for their username and password, and the backend does a check to make sure the user has a registered account with the provided password (with the /login route). Once signed in, the backend also creates a authentication token (self made), which the server uses to check to see if the user logs in. This token is used as a check in any function involving the user, and the token is returned as a header once the user successfully signs in. The unit tests check for possible issues, especially with the authentication token, and the tests all pass.

User Story #3 - 15 hrs
As a user, I would like to be able to reset my password if I forget it

#	Task Description	Estimated Time	Team	Owner

		(Hours)		
1	Create form and screen to reset old password to new password	2	Front End	Julien
2	Create button on login page to reset password	1	Front End	Julien
3	Set up mailer to send emails to users	2	Back End	Ken
4	Send reset token to user's email	2	Back End	Ken
5	Create API route to change user's password in the database	5	Back End	Keith
6	Create Unit Tests	3	Back End	Connor

Completed:

People forget things, including passwords, so we wanted to make sure if that is the case, the user would be able to reset it and make a new one. Using the existing framework build for emailing users, a random four digit token was generated and sent to the user, which was also saved to the user schema. A route was created to allow the user to put in their email and four digit token received from their email and then reset their password. Unit tests for the backend were successfully implemented and passed, and the frontend is set up accordingly to handle these tasks.

User Story #4 - 10 hrs
As a user, I would like to be able to view to my Critika profile

#	Task Description	Estimated Time (Hours)	Team	Owner
1	Create profile screen where user-specific attributes are displayed	2	Front End	Julien
2	Create buttons or clickable elements that navigate to profile screen from dashboard	2	Front End	Julien
3	Create API route to retrieve profile information	3	Back End	Keith
4	Create Unit Tests	3	Back End	Connor

Completed:

One of the most important features of a website that involves user interaction is the user profile. The information for the user (rating, stats, username) is nicely displayed on the UI. You can navigate the UI easily, and the /account route sends the user object to the client. All important information is stored in the user object. The route passes all unit tests.

User Story #5 - 10 hrs

As a user, I would like to be able to edit my Critika account

#	Task Description	Estimated Time (Hours)	Team	Owner
1	Create edit button on profile page	1	Front End	Julien
2	Create UI form where all user info is editable	3	Front End	Julien
3	Create submit button for when user is done editing	1	Front End	Julien
4	Create API route to update user info in the database	2	Back End	Keith
5	Create Unit Tests	3	Back End	Connor

Completed:

As of now, when signing in, the UI only prompts for a username (which is unique, a password, an email, and a security question). Since the username is unique and we decided to make it uneditable, and we already have a route for changing password, this task only involves changing the security question and email for now (we may add more options later, but that involves trivial edits to the code). The UI for edit profile is made nicely, and the /edit-info route changes the email and security question in the database. All unit tests are passed.

User Story #6 - 11 hrs
As an admin, I would like to be able to view all registered users

#	Task Description	Estimated Time (Hours)	Team	Owner
1	Create UI for seeing all registered users	2	Front End	Cyrus
2	Create search bar and filter for finding specific users	1	Front End	Cyrus

3	Create UI for viewing a registered users reported submissions and comments	2	Front End	Cyrus
4	Create option menu for admins to manage accounts	1	Front End	Cyrus
5	Create route from database of all current registered users and their report info	2	Back End	Keith
6	Create Unit Tests	3	Back End	Connor

Completed:

One of our routes (/all) returns a JSON object containing every user object with their information. This route passes all unit tests, and works as intended. In the front end, there was a page created for admins to view all of the users.

User Story #7 - 9 hrs
As a user, I would like to be able to add other users so that I can collaborate with them further. (9 hours)

#	Task Description	Estimated Time (Hours)	Team	Owner

1	Create a button to add another user as a friend	1	Front End	Cyrus
2	Create UI for viewing a list of all friends	2	Front End	Cyrus
3	Create a button to accept another user's friend request	1	Front End	Cyrus
4	Update database user object to store other user in array of friends	2	Back End	Connor
5	Create Unit Tests	3	Back End	Connor

Completed:

One future feature we want to add is the ability to add friends and form community groups or chats with those friends. We are going to focus on that later, but we decided to add this user story in sprint 1 just to get started. The /add-friend route adds the string that contains the friend's unique username to an array of friends in the database, and the UI buttons and forms were created.

User Story #8 - 11 hrs
As a user, I would like to be able to submit a form for submission. (11 hours)

#	Task Description	Estimated Time (Hours)	Team	Owner
1	Create button to submit a new post to be critiqued	1	Front End	Cyrus
2	Create UI form where a user can upload their submission	3	Front End	Cyrus
3	Create submit button that will upload the user submission	1	Front End	Cyrus
4	Update database user object to include the submission	3	Back End	Keith
5	Create Unit Tests	3	Back End	Connor

Completed:

One of the most important features on a website that aims at user to user interaction is the ability to submit a post about something. The UI for submitting a post to be critiqued asks for a title and text (with community to follow in future sprints). The /add route in the backend adds the post to the database and passes all unit tests.

User Story #9 - 14 hrs
As a user, I would like to be able to easily see submissions from other users I can give feedback to. (14 hours)

#	Task Description	Estimated Time (Hours)	Team	Owner
1	Create dashboard page button	1	Front End	Cyrus
2	Create dashboard page logo link	1	Front End	Cyrus
3	Create API query to receive other user submissions	3	Back End	Keith
4	Parse information from database and send a response to the client	3	Back End	Keith
5	Create home page UI where other user's submissions are shown	3	Front End	Cyrus
6	Create Unit Tests	3	Back End	Connor

Completed:

The next logical step after being able to submit a post is the ability to see other users posts. This is accomplished via the /available route in the backend. As of now this simply returns all submissions by a user, but in the future will be tweaked to show only select submissions.

Kommentar [1]: @keith.a.tan@gmail.com
Assigned to Keith Tan

User Story #11 - 11 hrs
As a user, I would like to be able to remove submissions as I see fit. (14 hours)

#	Task Description	Estimated Time (Hours)	Team	Owner
1	Create button to "remove" submission	1	Front End	Brandon
2	Create option window to confirm	2	Front End	Brandon

	removal			
3	Remove submission from the database	3	Back End	Ken
4	Update user statistics to reflect submission removal	2	Back End	Ken
5	Create unit tests to confirm addition and removal is successful	3	Back End	Connor

Completed:

As of now, both the backend route and the front end button and alert are set up. They just have to be integrated together in our sprint 2. The route successfully removes submissions from the database. The frontend button, when clicked, brings up an alert for the user to confirm that they are trying to remove that specific submission currently showing on the carousel.

User Story #12 - 11 hrs
As a user, I would like to be able to edit submissions as I see fit. (7 hours)

#	Task Description	Estimated Time (Hours)	Team	Owner
1	Create button to “Edit”	1	Front End	Brandon
2	Create options window to edit submission attributes	3	Front End	Brandon
4	Update changes in the database	4	Back End	Keith
5	Create Unit tests	3	Back End	Connor

Completed:

After submitting a post to be critiqued, if the user wants to edit their post (to fix typos or maybe add more information), then they should be able to do so. The UI form has an edit button with options to edit the various submission attributes. The /edit route updates the database with the new, edited info.

User Story #13 - 6 hrs
As a user, I would like to be able to specify my skill level for submissions. (6 hours)

#	Task Description	Estimated Time (Hours)	Team	Owner
1	Create slider for skill level	1	Front End	Brandon
2	Add skill level to submission object	2	Back End	Ken
3	Create Unit Tests	3	Back End	Connor

Completed:

As of now, there is a slider option that appears on both the “new submission” and “edit submission” page. On the back end, each user submission has a numerical value attached to it, that represents the skill level of the user for that specific submission. We will have to integrate the front end slider with the back end route in Sprint 2.

WHAT DID NOT GO WELL

User Story #10 - 12 hrs

As a user, I would like to be able to easily access my submissions. (9 hours)

#	Task Description	Estimated Time (Hours)	Team	Owner
1	Create button to navigate to submissions from the profile screen	1	Front End	Brandon
2	Create submissions page where all past submissions can be viewed	3	Front End	Brandon
3	Create dropdown filters to specify which types of submissions to view	2	Front End	Brandon

4	Create backend route to retrieve submissions for the specific user	3	Back End	Keith
5	Create Unit Tests	3	Back End	Connor

Currently, we have a front end submissions page. Which has a carousel to show all the submissions the user has. However, the front end page is not yet integrated with the back end routes. The carousel will need to show a thumbnail for each submission and also show the details about the submission either below or on a separate page. We will need to add these features while also integrating the front and back end in Sprint 2

HOW WE COULD IMPROVE

For some of our user story tasks, we overestimated the time it would take to complete each task. For this first sprint, some of our team needed to learn general web development tools. This significantly slowed down each task. However, now that we are all fairly on the same page, we will be able to more accurately estimate the time necessary to complete each task. Although each team now have a general idea of what to expect, there is still plenty of room for improvements.

Our six person team was split into two differing sub teams; front end and back end. While individually, the front end / back end work as expected, integration turned out to be much more difficult than expected, given our limited combined experience in web application development. However, now that we all have many more hours of experience with the technologies being utilized, this is not expected to be an issue, as once we figure out one integration task, then all others will fall into place.

Additionally, since there are six members in the team, meeting conflicts is inevitable. However, during the first sprint, there was a number of meetings in which multiple team members were not present, which potentially caused some fallbacks in developing the product. Another way to

further improve is making sure there's a time period in which everyone is committed in attending a group meeting. That way, discussions regarding the project's progression and development is relayed in a timely and efficient manner.

On the other hand, on the development side, there is also room for improvement. In our first sprint, we were unable to successfully implement integration and thus, there was a lack of frontend and backend communication. A way to improve the progression of the project, we should implement redux in order for all the components to use a shared store of information.