BoilerScout Sprint 3 Planning Document

Team 15

Baris Dingil, Hardy Montoya, Jacob Miecznikowski, Selin Ovali, Terry Lam

Sprint Overview

During this sprint, our team's goal is to wrap up the two last big features of our application, which is implementing functionality for a dedicated forum and private messaging between users. The backend team will be focused on writing tests for not only these new features, but also our current API too. The frontend team will be working diligently to improve user experience and UI design this sprint, as well as integrating with our last features. By the end of this sprint, users will be able to interface with all parts of our BoilerScout application.

Scrum Master

Hardy Montoya

Meeting Schedule

Standup meetings weekly on Tuesdays and Wednesdays (with Project Coordinator), longer group meetings on Sunday evenings.

Risks/Challenges

Moving forward onto sprint three, the frontend developers will have the most pressure on them to keep up. Not only will they be creating new views and components for our new features, but they will be tasked with refactoring and redesigning large parts of our application for the sake of a better UI/UX. Because of this, it will be imperative that each member of the backend team communicates clearly and works together with the frontend developers in order to push out new functionality on the client. If any member of the backend team is lagging behind on their work, that puts greater pressure on the frontend developers who will need to do their own testing and implementations.

Sprint Detail

User Story #1

As a user, I would like to view a variety of forums that contain relevant threads, so that I can participate in multiple topics of discussion (such as Mentor Search, Class Help, etc).

Task #	Task Description	Estimated Time	Owner
1	Design a relational model for the forum feature and create corresponding database tables.	2	Terry
3	Make alterations to ALL existing database tables and their foreign/primary keys to allow for cascading UPDATES and DELETION.	2	Terry
3	Expose a backend endpoint that will interface with our database when we want to retrieve existing forum entities.	1	Terry
4	Write the controller logic that handles parsing GET requests for returning existing forums for the frontend to display.	1	Terry
5	Expose an admin-only endpoint (accessible only via our backend) that can add or remove forum entities to our database.	1	Terry
6	Create Community view that displays the variety of forums	1	Selin

Acceptance Criteria

- ➔ Given that this user story is implemented successfully, the backend team will have access to an endpoint to add or remove forums as we see fit for our users (however, forums usually remain the same).
- ➔ Given that this user story is implemented successfully, if a forum entity is deleted or updated in our database, any threads that reference that forum (via a unique id) will also be deleted.
- ➔ Given that this user story is implemented successfully, if a user entity is deleted or updated in our database, any other tables with rows that reference that user (via a unique id) will also be deleted or updated.
- ➔ Given that the controller logic is implemented successfully, if a request is made from the client to view forums, a list of forum entities will be returned from the database for the

frontend to display.

- ➔ Given that the front end Community view is implemented successfully, user will have a list of the variety of forums they can access that links to the appropriate forums.

User Story #2

As a user, I would like to view a forum containing different threads from other users, so that I can see what other Purdue students have posted.

Task #	Task Description	Estimated Time	Owner
1	Secure the API endpoint (token verification)	1	Terry
2	Expose a backend endpoint that will interface with our database when want to to retrieve threads under a certain forum.	1	Terry
3	Write the controller logic that handles parsing GET requests when retrieving threads under a particular forum.	1	Terry
4	Write the controller logic that handles returning thread entities from the database that reference a particular forum (newest to oldest)	3	Terry
5	Write unit tests to check proper response output from the given endpoint(s).	1	Terry
6	Create a Forum view that displays a list of threads started by users.	2	Jacob
7	Retrieve and display the thread entries on the client-side for a specified forum from the database and display them.	3	Jacob
8	Manual Testing of Frontend	1	Jacob

Acceptance Criteria

- ➔ Given that the logic to retrieve threads is implemented correctly, when a GET request is sent to the appropriate endpoint, a list of thread entities will be returned in the response body and displayed.
- ➔ Given that the forum feature is implemented correctly, when a query is inputted that does not match any threads, none will be displayed and the user will be notified that no results were found.
- ➔ Given that the forum feature is implemented correctly, when a user clicks into a particular forum on the client from the community page, he/she will be able to view a list of threads inside.

➔ User Story #3

As a user, I would like to start a new thread on the forum, so that I can ask questions or post issues I have that other Purdue students may be able to help me with.

Task #	Task Description	Estimated Time	Owner
1	Secure the API endpoint (token verification)	1	Terry
2	Expose a backend endpoint that will interface with our database when we want to commit new thread entities (start new threads).	1	Terry
3	Write the controller logic that handles parsing POST requests to start new threads.	1	Terry
4	Write the controller logic that handles committing new thread entities under the corresponding ‘threads’ table	3	Terry
5	Write unit tests to check proper response output from the given endpoint.	1	Terry
6	Create a Create New Thread view	2	Selin
7	Send POST request containing the new thread	3	Selin
8	Display user the Forum they posted the Thread on	2	Selin
9	Manual Testing Document	1	Selin

Acceptance Criteria

- ➔ Given that the controller logic to start new threads is implemented correctly, when a POST request is sent to the appropriate endpoint, a new thread entity will be committed to our database.
- ➔ Given that token verification is implemented successfully, when a user tries to start a new thread with an invalid token, they will be met with an error (and asked to login again on the client-side).
- ➔ Given that the `New Thread` view is successfully created, user will be able access a page where they can create a new thread.
- ➔ Given that the response from the backend is successfully received and managed, the user will be displayed the Forum they posted their thread on.

User Story #4

As a user, I would like to view and comment on individual threads, so that I can provide my expertise in an area if it calls for it.

Task #	Task Description	Estimated Time	Owner
1	Secure the API endpoint (token verification)	1	Terry
2	Expose a backend endpoint that will serve to interface with our database when we want to commit new ‘post’ entities that will act as comments to a thread	1	Terry
3	Expose a backend endpoint that will interface with our database when we want to retrieve all comments inside a particular thread.	1	Terry
4	Write the controller logic that handles parsing GET requests when attempting to retrieve all comments under a specific thread.	1	Terry
5	Write the controller logic that handles parsing POST requests to add a comment on an individual thread.	1	Terry
6	Write the controller logic that handles commiting new ‘post’ entities to the corresponding ‘thread_posts’ table	3	Terry
7	Write the controller logic that handles retrieving and returning comments from our database that are tagged with the relevant thread id.	2	Terry
8	Write unit tests to check proper response output from the given endpoint.	1	Terry
9	Create Thread view for individual threads	1	Selin
10	Make a GET request to the backend and receive thread content	2	Selin
11	Create Post view for individual replies	2	Selin
12	Display replies using the Post view	2	Selin

13	Create a reply box	2	Selin
14	Make GET request to the backend if a new reply is submitted by a user	3	Selin
15	Manual Testing Document	1	Selin

Acceptance Criteria

- ➔ Given that the controller logic to post replies is implemented correctly, when a POST request is sent to the appropriate endpoint, a new post entity will be committed to our database.
- ➔ Given that the controller logic to retrieve replies in a thread is implemented correctly, when a GET request is sent to the appropriate endpoint, a list of post entities will be returned for a particular thread.
- ➔ Given that token verification is implemented successfully, when a user tries to view or post in a new thread with an invalid token, they will be met with an error (and asked to login again on the client-side).
- ➔ Given that the Thread view is successfully implemented, users will be able to view their own or others' threads.
- ➔ Given that the Post view and displaying them is successfully implemented, the replies under each thread will be displayed.
- ➔ Given that the reply box is successfully implemented, users will be able to create a reply to a thread and submit it, and view their reply under the said thread.

User Story #5

As a user, I would like to search through threads in a particular forum, so I can look for threads that are relevant to me.

Task #	Task Description	Estimated Time	Owner
1	Create search component for frontend	2	Jacob
2	Update thread results as user types	2	Jacob
3	Manual Testing of Frontend	1	Jacob

Acceptance Criteria

- ➔ Given that the search component is implemented correctly, when a user enters in a query in the search box, the displayed threads should update to only show what is matched by the current query
- ➔ Given that the search component is implemented correctly, when a user enters a query that does not match any threads, none will be displayed.
- ➔ Given that the search component is implemented correctly, when a user clicks on a search result, he/she should be taken the the page that shows that thread.

User Story #6

As a user, I would like to have advanced filters in addition to regular search such as year standing of students or their majors, so I can make my search as narrow or as wide as I want.

Task #	Task Description	Estimated Time	Owner
1	Extend Scout to have advanced filters filters that include “Graduation Year” and “Major”	4	Jacob
2	Manual Testing of Frontend	1	Jacob
2	Refactor the current search method in the backend to accept “Graduation Year” and “Major” as additional search parameters	3	Hardy
3	Write unit tests to check proper response output from the given endpoint., both for the new requests added, and the old	2	Hardy

Acceptance Criteria

- ➔ Given that the Scout is extended to handle more specific queries from the user correctly, the method should be able to interact with more areas of the database, and return users accordingly.
- ➔ Given that the Scout is extended to handle more types of data from the user correctly, there will be a field drop down where the user can select the graduation year and a drop down where the user can select the major to search.
- ➔ Given that the Scout is extended to handle more types of data from the user correctly, when the user searches with advanced options selected, users will be displayed that match the advanced criteria.
- ➔ Given that the Scout is extended to handle more types of data from the user correctly, when a user enters in a query with the graduation year and major, and there is no results, the user will be **notified that there are no results** and no results will be displayed.

User Story #7

As a user, I would like to update my password through Settings.

Task #	Task Description	Estimated Time	Owner
1	Secure the API endpoint (token verification)	1	Hardy
2	Implement a secure method that allows the user to update their password.	3	Hardy
3	Update current reset password method to a safer alternative	4	Hardy
4	Write unit tests to check proper response output from the given update endpoint	3	Hardy
5	Make POST request to update password	2	Jacob
6	Visually inform the user if updating password was successful	2	Jacob
7	Manual Testing of Frontend	1	Jacob

Acceptance Criteria

- ➔ Given that the update password feature is implemented successfully, users will need to input their current password, as well as to confirm the new one before any value changes in the database.

- ➔ Given that the update password feature is implemented successfully, users will be able to enter a new password and submit it.
- ➔ Given that the POST request is implemented successfully, user's password will be changed in the database.
- ➔ Given that the visual features of update password are implemented successfully, user will be informed whether their password was successfully updated.

User Story #8

As a user, I would like to manage my further account information, such as name, graduation year, and major.

Task #	Task Description	Estimated Time	Owner
1	Secure the API endpoint (token verification)	1	Hardy
2	Write controller logic to parse a POST and interact with the requested fields	3	Hardy
3	Write controller logic to update only the requested fields	3	Hardy
4	Write unit tests to check proper response output from the given endpoint	2	Hardy
5	Write frontend logic to track changed fields and send POST request with only those fields	3	Selin
6	Add fields to Update Profile for name, graduation year and major	2	Selin
7	Edit the POST request for updating profile to include the new fields	3	Selin
8	Manual Testing Document	1	Selin

Acceptance Criteria

- ➔ Given that the backend logic to update the profile is correctly implemented, the user should have to provide his current password in order to change any of the settings (name, graduation year, password)
- ➔ Given that the new options to update profile feature is implemented successfully, users will be able to change their name, graduation year and major in addition to the already existing options.
- ➔ Given that the POST request to update profile information is implemented correctly, user's changes will be sent to the backend application to be handled.

User Story #9

As a user, I would like to have an inbox and outbox where I can send and receive messages.

Task #	Task Description	Estimated Time	Owner
1	Secure the API endpoint (token verification)	2	Baris
2	Return messages from a specific user as a response by creating a controller using GET request.	5	Baris
3	Write unit tests to check proper response output from the given endpoint.	3	Baris
3	Create Inbox View.	2.5	Jacob
4	GET all message threads and display them.	3.5	Jacob
5	Display the message received from a given user when a result it clicked	5	Jacob
6	Add a box that allows the logged in user to a send a message to the other user that the given message is from	3	Jacob
7	Create Outbox	2	Jacob
8	Display threads in outbox, show full message when clicked	2	Jacob

9	Manual Testing of Frontend	1	Jacob

Acceptance Criteria

- ➔ Given that the messaging inbox/outbox is implemented correctly, a user should be able to view all of his/her messages from the inbox/outbox view.
- ➔ Given that the messaging outbox is implemented correctly, when a user clicks on a message when viewing all of the sent messages, that message will be displayed.
- ➔ Given that the messaging inbox is implemented correctly, when the logged in user is viewing a message from another user, a reply box will be present and will send the message the other user when filled out and submitted.

User Story #10

As a user, I would like to have a smooth and intuitive user experience.

Task #	Task Description	Estimated Time	Owner
1	Give visual feedback on forms on the validity of their password entry	3	Selin
2	Give visual feedback on forms on the validity of their email entry	3	Selin
3	Redesign page format to have clear boxes and background	5	Selin
4	Rethink and redesign color scheme of web application	5	Selin
5	Manual Testing Doc	2	Selin

Acceptance Criteria

- ➔ Given that the visual feedback regarding password input in already existing pages is correctly implemented, users should have visual feedback informing them of the validity of their password.
- ➔ Given that the visual feedback regarding email input is correctly implemented, users should have visual feedback informing them of the validity of their email.
- ➔ Given that the formats of some views are changed, appropriate views will have clear background and boxes.

User Story #11

As a user, I would like to search for messages in my inbox.

Task#	Task Description	Estimated Time	Owner
1	Create search messages component	2	Jacob
2	Display messages as the user types	2	Jacob
1	Manual Testing of Frontend	1	Jacob

Acceptance Criteria

- ➔ Given that searching a keyword in the inbox implemented successfully, users should be able to see the messages containing the keyword be returned in his inbox.
- ➔ Given that searching a keyword in the inbox implemented successfully, users should get a message indicating that no message matches the keyword, if this were to be the case.
- ➔ Given that the search features of the inbox is implemented successfully, when a user searches for a keyword, any messages containing the keyword should be displayed.
- ➔ Given that the search features of the inbox is implemented successfully, when a user searches for a keyword that doesn't exists, he/she will be informed that the search query is not found and nothing will be displayed.

User Story #12

As a user, I would like to sort my messages according to time.

Task#	Task Description	Estimated Time	Owner
1	Secure the API endpoint (token verification)	2	Baris
2	Write the controller logic that handles parsing GET requests to return messages from oldest to newest or vice versa.	5	Baris
3	Write unit tests to	3	Baris

	check proper response output from the given endpoint.		
4	Create sort messages component.	2	Jacob
5	Display messages after sending a GET request of the sort.	2	Jacob
6	Manual Testing of Frontend	1	Jacob

- ➔ Given that the sorting messages according to time is successful, if user wants to sort its messages from newest to oldest, messages will be displayed from newest to oldest message.
- ➔ Given that the sorting messages according to time is successful, if user wants to sort its messages from oldest to newest, messages will be sorted from oldest to newest.
- ➔ Given that the GET request after a search has been made is implemented successfully, the user will be displayed with the appropriate search results in the inbox.

User Story #13

As a user, I would like to log out of my current BoilerScout session, so I can protect my account.

Task #	Task Description	Estimated Time	Owner
1	Secure the API endpoint (token validation)	1	Terry
2	Write controller logic that wipes a user's token upon "log out"	1	Terry
4	Add logout button to navbar	1	Jacob
5	Remove user id and token from local storage; Redirect to home	1	Jacob
6	Manual Testing of Frontend	1	Jacob

Acceptance Criteria

- ➔ Given that the controller logic to search threads is implemented correctly, when a GET request is sent to the appropriate endpoint, a list of thread entities will be returned from

our database based on the user's query.

- ➔ Given that the logout component is implemented correctly, a user should be redirected to home once the logout button is clicked
- ➔ Given that the logout component is implemented correctly, when a logged out user tries to access a page when he/she is not logged in, they will be kicked back to home.