BoilerScout Sprint 2 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 integrate between the frontend and the backend. More specifically, we will be doing a lot of work testing that clients will be able to hit and interact with our API endpoints. The frontend team will be creating more views, specifically ones that deal with searching and user profiles. Our backend will create functionality for updating user profiles, viewing profiles, and email verification. By the end of this sprint, users should be able to sign up and create an account with our application, verify their accounts via their Purdue email addresses, and search/view other profiles.

Scrum Master

Jacob Miecznikowski

Meeting Schedule

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

Risks/Challenges

Moving forward onto sprint two, there are a few challenges that our team will face. The main one being that most of our backend members still are not very familiar with how the current code functions, as they have not written any formal code yet. It will be a challenge to not only keep up the pace with our sprint plan, but also to continue teaching our members and make sure we're all on the same page. Communication between members has been getting better as we have moved onto using Slack, but we will need to do our best to make sure that it does not break down again.

Sprint Detail

User #1

As a user, I would like to create and update a profile that contains a short biography, classes taken and my personal skills, so that other users can see if I meet their needs.

Task #	Task Description	Estimated Time	Owner
1	Write logic to parse POST requests that contain initial profile information the user inputs	3	Terry
2	Write logic to assign unique identifiers and insert to database tables for situations where brand new (unique) skills and courses are added by a user.	5	Terry
3	Write query logic to insert into database table the initial profile information from the user	3	Terry
4	Write query logic to remove any skills, courses, and biographical information from the respective database columns.	3	Terry
5	Implement guards to prevent the additions of duplicate courses or skills for an individual user.	3	Terry

6	Secure the API with error handling (token verification for every call)	1	Terry
7	Create Profile View	2	Selin
8	Route view with application and make POST request with data when profile is saved	5	Selin

Acceptance Criteria

- ➔ Given that profile creation logic is implemented successfully, when a person tries to edit their profile for the first time, they will be able to add new technical skills, add enrolled courses, and a new biography.
- ➔ Given that profile creation logic is implemented successfully, when a person tries to edit their profile, they will be able to remove any information that is not needed anymore, so they can keep their profile up to date.
- ➔ Given that token validation is implemented successfully, when a user tries to create their profile with a stale token, they will be met with an error (and asked to login again on the client-side).
- ➔ Given that request parsing is implemented successfully, when a user creates their profile, a proper POST request will be sent to our backend services containing the profile data in a request body (client-side).
- ➔ Given that input validation is implemented successfully from a backend perspective, if a user tries to add the same skill or course twice, it will not be duplicated in our database.

Testing

To test this module, we will be calling our API via Postman with different inputs in our requests. Proper and malformed input will test to see if our profile creation flow is robust enough, and ensures that our error handling is correct. Results can simply be examined through entries in our database as well as the response object. We will also be testing token validation with stale tokens when calling our API.

User #2

As a user, I would like to be able to complete a basic search for a name, skill, or course and have a list of users that match be returned.

Task #	Task Description	Estimated Time	Owner
1	Write logic to parse GET requests for users/skills/courses	2	Terry
2	Write query logic to search database and return users based on a query on their name	3	Terry
3	Write logic to link user_skills and skills table in the database and return users when querying for a specific skill	3	Terry
4	Write logic to link user_courses and courses tables in database and return users when querying for a specific course or course type.	3	Terry
6	Secure the API with error handling (token verification on each call to search)	1	Terry
7	Get results from backend and display user results in the Scout page	7	Jacob

Acceptance Criteria

- ➔ Given that basic search is implemented successfully, when a person enters a name, skill, or course into their search, the backend service will return a list of users in the response.
- ➔ Given that a GET request is parsed with a query that doesn't return anything from our database, an empty `query` list will be returned in the response body.
- ➔ Given that basic search is implemented successfully, when a person enters a blank search input on the client, no users will be displayed.
- ➔ Given that token validation is implemented successfully, when a user tries to enter a search with a stale token, they will be met with an error (and asked to login again on the client-side).

Testing

To test this module, we will be calling our API via Postman with different inputs in our requests. Proper and malformed input will test to see if our search logic captures input correctly, and ensures that our error handling is correct. Results can simply be examined through entries in our database as well as the response object. We will also be testing token validation with stale tokens when calling our API.

User Story #3

As a user I would like to sign up with my Purdue email and a unique password, so that I can have a set of credentials to use the application.

Task #	Task Description	Estimated Time	Owner
1	Send POST Request containing user's username and password from Sign Up component	3	Selin
3	Display page if sign up was successful	1	Selin

Acceptance Criteria

- ➔ Given that sign up is implemented correctly, when a user enters an email and password into the sign up form, the backend service will return an OK status.
- ➔ Given that sign up is implemented successfully, when a user enters a email that already exists, the backend will report a bad request.
- ➔ Given that sign up is implemented successfully, when a user enters an incorrect password, the backend will report a bad request.
- ➔ Given that sign up is implemented successfully, if a user logs in successfully, they will be displayed a page indicating success and have the option to redirect to Login.
- ➔ Given that the view is implemented successfully, if a user isn't able to log in, they will be displayed with a message that lets them know there is an error.

Testing

To test this module, we will supply the input with valid and invalid emails and usernames. Invalid emails are ones that are already created. When an invalid email is inputted, the user should be notified of this. When valid input is supplied, then the user should be redirected to the login page to login with their newly created account.

User Story #4

As a user, I would like to login with my Purdue email the and password that I created, so that I can access and use the application.

Task #	Task Description	Estimated Time	Owner
1	Send POST Request containing user's username and password from Login Component	4	Jacob
2	Get a User ID and token from server and store locally	1	Jacob
3	Redirect user to Home	1	Jacob

Acceptance Criteria

- ➔ Given that the login is implemented correctly, when a user enters in an email that exists and the corresponding password is valid, the user will be redirected to the home and the user id and token will be stored locally.
- ➔ Given that the login is implemented correctly, when the user enters an email that doesn't exists, he/she will be notified that the email is incorrect.
- ➔ Given that the login is implemented correctly, when the user enters an email that exists but with a wrong password, he/she will be notified that there is an error.

Testing

To test this module, we will manually enter valid and invalid emails and and passwords as input to test if our logic captures the input as it should. This will also show that our error handling is correct. These results will be shown on the screen. The user should be redirected to to the homepage on a successful log in.

User Story #5

As a user, I would like to click on other users' profiles and view them.

Task #	Task Description	Estimated Time	Owner
1	Secure API endpoint (token and user validation)	1	Hardy
2	Write logic to retrieve all the data related to a user's from the database into an object.	3	Hardy
3	Make GETrequest including the selected user's id	3	Jacob
4	Create User View	3	Selin
5	GET user profile information and display it	5	Jacob

Acceptance Criteria

- ➔ Given that clicking a user's profile is implemented successfully, a person will have the ability to click on a profile and be taken to a page to display all that user's information.
- ➔ Given that the POST request to backend application was implemented successfully, inputs will be sent to background application.
- ➔ Given that the response to backend application is made successfully, the view will be updated depending on if password was updated successfully or not.
- ➔ Given that the logic to retrieve a user's data is successfully implemented, once the profile of a user is requested, the backend should forward a map or JSON object containing the requested user's information so it can be parsed and displayed by the client.

Testing

To test this module, we will see if the user is taken to another person's profile when they click on the said user. We also need to test, either manually or by unit tests, that the information returned is actually correct, and ensure that a bad input (invalid user or nonexistent one) returns the appropriate error.

User Story #6

As a user, I would like to be able to resend the confirmation email, so that I can verify my account in case the email is lost.

Task #	Task Description	Estimated Time	Owner
1	Write logic to send emails to users	2	Hardy
2	Write query logic to change the ‘verified’ field corresponding to the user to a random value, and compare it with one received via an endpoint	3	Hardy
3	Create an endpoint for the url to be sent via email, which needs to contain the randomly generated value and compare it to the one in the ‘verified’ field for the user	4	Hardy
4	Write logic to change the ‘verified’ field in the database to 1 if the user successfully verified via the above method	1	Hardy
5	Write a method so a user can request a new verification email, invalidating the previous one	3	Hardy
6	Send POST request with the email that was entered in Resend Confirmation	2	Jacob

Acceptance Criteria

- ➔ Given that the method to generate a random number and the query logic are implemented successfully, the ‘verified’ field in the database should not ever be equal to 1 unless the user has verified, and never change its value after it becomes 1
- ➔ Given that email verification method and the query logic are implemented successfully, a user will be considered verified, permanently, after just one successful verification, and won’t have any restrictions to access the application.
- ➔ Given that method to send a new verification email is implemented successfully, the first email should be useless for any purpose, and the user should be able to successfully verify with the new one only.
- ➔ Given that the resend confirmation email is implemented correctly, the user should be notified that the confirmation email has been sent when he/she clicks the submit button.

Testing

To test this module, we will be creating dummy accounts and then verifying them via Postman. Their verification status will be different and should also return different codes (errors or not) upon attempting a normal login. Requesting a new email will also be tested manually, as well as trying to verify with an old email, which should return an error as well.

User Story #7

As a user, I would like to be reset my password via email if I forget it, so that I can keep my account and information in case of such an event.

Task #	Task Description	Estimated Time	Owner
1	Write logic to send a POST request with the user's email	2	Hardy
2	Write query logic to change the user's password to a randomly generated one	2	Hardy
3	Write logic to send the user an email containing the new generated password, so they can login and then change it to their taste via the in-app method	2	Hardy
4	Send POST request with the email that was entered in Forgot Password	3	Jacob

Acceptance Criteria

- ➔ Given that the POST method is successfully implemented, inputting an invalid email or unexisting user will return an error instead of sending an email.
- ➔ Given that the method to generate random passwords is successfully implemented, these should be very safe and also somewhat undesirable (too hard to remember) by the user, so that they will want to change it to one of their choice once they are able to login.
- ➔ Given that the method to send an email with the new password is successfully implemented, requesting a new email should send a new one and make the previous one useless, and invalidate the password it contained.

Testing

To test this module, we will be calling our API via Postman with different emails and inputs in our requests. Invalid emails, or ones that do not correspond to an existing user will return an error. Results can simply be examined through entries in our database as well as the response object. Testing the new password will be done manually via Postman as well, and will be done several times to ensure that the method is consistent and safe.

User Story #8

As a user, I would like to be able to navigate the web pages with a top navigation bar.

Task #	Task Description	Estimated Time	Owner
1	Organize hierarchical organization of components	2	Selin
2	Change view so logo is hidden when logged in	1	Selin
3	Change view so top navigation bar is hidden before user has logged in	1	Selin
4	Style and fix top navigation bar	2	Selin

Acceptance Criteria

- ➔ Given that the hierarchical is implemented correctly, the page will have a different structure depending on whether a user is logged in or not.
- ➔ Given that the logo feature is fixed, the user will have the logo hidden after they have logged in.
- ➔ Given that the top navigation bar feature is fixed, the user will be displayed a navigation bar at the top of each view.

Testing

To test this user story we will make sure that all the links in top navigation bar route to the correct views, that the logo is hidden after logging in, and that navigation bar only appears after user has logged in.

User Story #9

As a user, I would like to privately message other users. (backend)

Task #	Task Description	Estimated Time	Owner
1	Create datatable for each individual message.	5	Baris
2	Get the message.	5	Baris
3	Get the destination(Message receiver user) and check if it exists.	2	Baris
4	If message is valid put it into Messages datatable.	5	Baris

Acceptance Criteria

- ➔ Given that private messaging system is implemented correctly, when a user sends a private message, It should take the information and save it in the data.
- ➔ Given that user puts the correct information for the receiver, they should be able to sent their message
- ➔ Given that the user inputs incorrect information into the receiver, then he/she should receive an error.

Testing

To test this user story, a user will sent a private message to other user, if the other user (destination) exists, it will allow the message to be sent, otherwise user will get a error message.

User Story #10

As a user, I would like to have an inbox where I can receive messages. (backend)

Task #	Task Description	Estimated Time	Owner
1	Create a data receiver for each individual message.	3	Baris
2	Connect send and receive according to text message, user and destination	3	Baris
3	Forward messages from sender to receivers data.	5	Baris

Acceptance Criteria

- ➔ Given that user receives a message from another user, user should get a text in its “inbox” data whenever it receives a message.
- ➔ Given that the inbox is implemented correctly, the user should get a message in his/her inbox in real time.
- ➔ Given that the inbox is implemented correctly, the message should be sent to the correct user

Testing

To test this user story, we need to be done with user story #11, User sends a private message to other user. If the messages are saved, it means that this user story is working.