Code changes / implementation for group “a-sep-up” milestone 2:

Code changes are organized by user stories needed to implement them.

Graphical user interface, text, application

Description automatically generated

**As an unregistered website user, I want to register as a regular user so that I can have an account.**

The implementation of this code concerns 3 pages of code, the user controller in the backend and the register.js and securityActions js scripts in the front end.

What was initially was given to us was a partial implementation of the register, in the form of the prebuilt back and front end, an initial sign-up page with an implemented register form partially linked to a local h2 database.

Graphical user interface, text, application, email

Description automatically generated

An initial code changes the immediately prevented the database from showing and therefore working was a missing dash file which was changed in the application and security properties. This was step 1 in implementing/fixing the register code.

Shape, rectangle

Description automatically generated

A previous issue in the registering process, is that the registering was never called or used in the javascript file, resulting in an end stup with no POST result after registering. The backend was verified using POSTman to ensure that once posted, would appear on the backend. The implementation made ensured the export used and connected the state so that once registered, it would send the appropriate POST signal to the backend.

Cross origin was added to both register and login methods in the controller in the back end to prevent the authentication error from occurring, however the placement of these permissions may change on the route of development depending on how the group handles deployment.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Please note here that a user can successfully register with the given details.

Text

Description automatically generated

**As a registered user I want to log into the website so that I can have a personalized experience.**

From this user story, things to keep in mind was that we had believed at the time we were retrieving the details of the database and verifying it with this. Change commands from the login form. However this did not end up being the case.

A picture containing text

Description automatically generated

Passwords were encrypted with the use of the swttoken libraries. Meaning the initial plan of verification of comparing a on submit password to an unencrypted password value was no longer possible. Thus our poker estimation was not accurate to the difficulty of the task shown on jira. In hindsight in my opinion, I believe that such a task would have been rated a 9th with the knowledge I have now after implementing.

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Homy provided a baseline to implement the login, however no verification or user feedback was given after a login event occurred, we also noted the lack of jwttoken that was missing from a codebase push. From testing we were able to find that jwttoken hosted a lot of the local data we needed and would use to retrieve for events like login, and potentially later, user status/book profiles/financial logistics. What was needed to be implemented was indication that a user was infact logged in and would need to be reflected so visually in some way. Our solution for this came with the need for operant rendering, specifically that for whether a user was currently logged in as demonstrated by the login panel. Graphical user interface, application

Description automatically generated



As shown here, after using the registered user done in user story mp–11, the login page accepts the registered user and displays their login username on the top right of the corner as well as a implemented but ugly logout prompt.

The code that allows us to display the code comes from:

Graphical user interface, text

Description automatically generated

This code declares user as a const which accesses the data from the store and receives it via the getState().

Timeline

Description automatically generated with medium confidence

Isloggedin is a declared state in the model that dictates on the Boolean basis whether a user is logged in (1) or not (0), with this it provides a foundational base potentially to include other statuses that can be grabbed from the model database, such as user account type, which ostensibly leads us to the next collection of user stories.

Graphical user interface, text, application, chat or text message

Description automatically generated

**As a publisher user I want to log into the website so that I can see the publisher dashboard.**

**As an admin I want to log into the website so that I can keep it running smoothy.**

The focus of these stories was operant rendering under the assumption of the user type, of the ones available there were 3 prepared: user, admin, and publisher. To account for this, we needed to make sure the database schema USER had a separate column of which is always filled. This is achieved by modifying the model class USER in the back end as on sprint bootup the schema is automatically filled.

Text

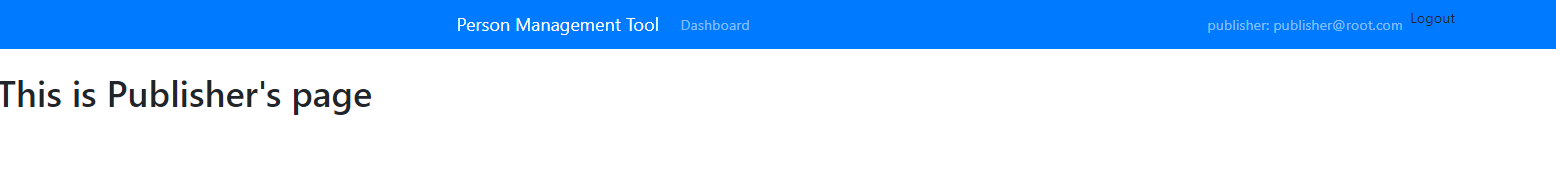
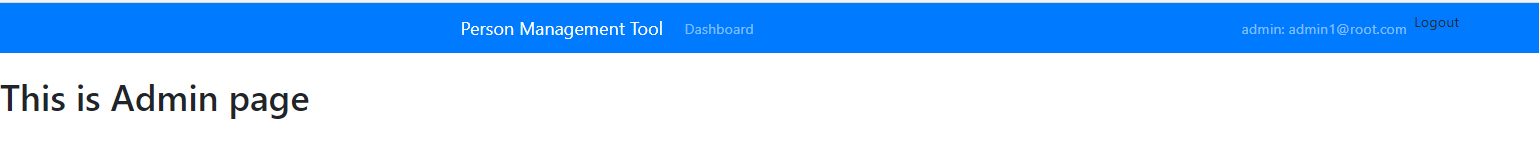
Description automatically generated

As we’re still currently still using h2 console as our database, the results are always rewritten on compiling, thus we currently use a dropdown menu to dictate our what type of user a userstatus is assigned to per signup event.

Graphical user interface, text, application, email

Description automatically generated

As the duties of the admin and publisher are strictly out of scope of this sprint, we implemented placeholder pages to indicate the differing pages that would be offered to a publisher or admin as shown below.

  
  
this is made possible with operand rendering that was made in milestone mp-14, thus the user story estimation in this case is accurate but with the assumption of the previous implementation, the estimation of the difficulty is accurate.

Graphical user interface, text, application, chat or text message

Description automatically generated

MP-12 and MP-13 are currently user stories that are in limbo, confirmation to ensure mp-12 works is dependent on the display of mp-13, the groundwork to getting there includes modifying a current user which has not yet been done in our progress yet, as creating new users are based on the code that was given to us in the code base.

Graphical user interface, text, application, Word

Description automatically generatedThe idea is for a registered user to be able to register himself as a publisher, the request would go to a database where an admin who is able to view the database would be able to change the type of the user through the front end.

The form has been made, which sends a request to the back end which sets a Boolean “usertopublisher” which hopefully will let the admin user view all users with the Boolean set to 1.

From there the admin should be able to change the user to a publisher type.

Of the code that’s currently being implemented for mp-13 can be seen here:

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Of the current obstacles overcome was the printing of users in a list, the next step is to display user’s by a certain status and then by changing the status.