BOOKEROO Sprint Planning Notes

Team: 3 (Friday 10:30am Tutorial)

Sprint: 1

Date: 09/08/21

Attended: Jeffrey, Ross, Allister, Mohammad

Scrum Master: Jeffrey

Product Owner: Dipto (Tutor)

Development team: Ross, Allister, Mohammad

1. Goal

Our team would like to create a shippable product for BOOKEROO.

This is defined by:

(1) A Spring Boot REST API back-end to manage *Book* and *User* models and repositories as well as support authorised API calls for CRUD of Books and Users.

(2) A React web application with user interface components which run API calls and subsequently display returned data.

Together, (1) and (2) will provide functionalities relating to:

* + User Authentication and Login
  + Displaying Books

1. Duration of the sprint

*2 weeks*

1. What is the team’s vision for this sprint?

Dipto (product owner) spoke to Jeffrey (scrum master) and said that we need to prioritise user stories which relates to the listing and searching of books. We also require a secure application (encryption/authorisation).

Any user story which relates to listing books, and user registration/login will be added to the sprint backlog.

A potentially shippable product will look like:

* Login and registration forms
* Homepage with list of books
* Search and sort functions for listing books
* Info and purchase page for individual books
* A dashboard for all three types users (public, admin, shop-owner/publisher)

And have features:

* REST API for users
* REST API for books
* Public users can create accounts
* When business users create accounts, admins must approve.

1. Estimation in story points

|  |  |  |
| --- | --- | --- |
| Story | Points | Justification |
| As an unregistered user, I want to sign-up with my details, so I can create a public or business account with BOOKEROO. | 13 | Creating the React form will not be too hard. There are many tutorials online which can help with styling.  As for the backend, a spring microservice can be made for users.  Entails creating an API call to create a user in the database.  For now, we don’t have to make the application perfectly secure. |
| As a registered user, I want to enter my login information, so I can be directed to my application dashboard. | 3 | Just a matter of verifying username and password and showing the user the correct dashboard based on their user type.  Needs to be secure.  Once we have the spring microservice setup from the registration user story, we can setup our microservice to authenticate user credentials as well. |
| As an admin user, I want to approve/reject business accounts via my portal, so I can verify the legitimacy of a shop owner/publisher. | 21 | Requesting/Denying pending business accounts through a dedicated admin page requires the creation of the admin dashboard.  The frontend for the dashboard will need to support viewing of pending requests as well as the ability to approve/reject requests. All this with decent styling.  Dashboard will also need to setup frontend skeleton for other admin features. May take a considerable amount of time experimenting with frontend design.  As for backend, this would be a GET request with a query for those accounts that are ‘pending’. Then we need a PATCH request to update the approval status. |
| As an admin user, I want power to add/remove/edit users, so I can control account access on the platform. | 8 | Using the same frontend template of the admin portal, we can create a separate page with a GET request for all users.  Then we also have to setup buttons for backend requests POST/PATCH/DELETE that admins have access to.   Will require authorization in the API request header for security. |
| As an admin user, I want to add/remove/edit books, so I can control what information or listings are available to the public. | 13 | Before we begin, we need to create a model + repository + REST-service for books. Also, we need to populate our database with books from an ISBN database somewhere online. This may take some time.  Using the same template of the admin portal, we can create a separate page with a GET request for all books.  Then we also have to setup buttons for backend requests POST/PATCH/DELETE that admins have access to.  Will require authorization in the API request header for security. |
| As a customer, I want to see a book cover image preview with a table of contents, so that I can make an informed decision before buying an item. | 5 | React page render with data fetched from microservice. Some reasonable styling, but not too overboard. Add purchasing button, but no functionality for purchasing for now. |
| As public user, I want to search for books by name, author, ISBN and category, so I can view listings for books based on my interest. | 5 | Ross says that he is confident with implementing search functionality.  Using a SQL based database, this should be accomplished without too much headache.” |
| As any user, I want to see a list of all available books on the homepage, so I can browse the website's collection. | 5 | This is the main priority of the Sprint. Hence, we will be ensuring that the front-end looks pleasing to the client and that the book listing functionality works as promised. |
| As a public user, I want to contact the administrator so that I can contact about them and send a message to them | 3 | This can be a static page. Mohammed mentioned that he would like to use a library to produce an in-page email client to make the website seem more impressive. |
| As a public user, I want to view the about page, to know more about the application and its owners | 1 | This is just a static page with our names and emails. |
| As public user, I want to sort book lists by price + other parameters, so I can easily find the books I am looking for. | 3 | Ross says that he is confident with implementing sort functionality.  Using a SQL based database, this should be accomplished without too much headache. |

Story Point Budget = 4 \* 20 = 80.

**Estimated Velocity = 80**

We have 3 story points to spare. So, if all goes to plan, we should be able to finish all added user stories within this Sprint.