Mixbook

Mixology Application

Sprint 2 Retrospective

https://github.com/alexcoll/mixbook

Team 3:

Alexander Coll (acoll@purdue.edu), Matthew Fouts (foutsm@purdue.edu),

John Tyler Preston (prestoj@purdue.edu), Nicholas Zenobi (nzenobi@purdue.edu)

What went well?

Overall we were able to complete all of the user stories as well as set up what is needed for going forward into sprint 3.

User stories:

- As a user, I would like to choose from pre-populated recipes as well as user-added recipes
  - Created 100 pre-populated recipes
  - Made script to create them from database creation all under one unified pre-populated account
  - Ensured professionalism in recipes by checking grammar
  - Tested and ensured robustness of all-things-recipes, checking common corner cases and the likes
- As a developer, I would like for users to be able to lock their accounts
  - Created frontend UI for locking accounts
  - Created backend UI for requesting to unlock an account
  - Created backend UI to unlock an account
  - Created backend handling of unlocking an account
  - Tested and ensured robustness of locking/unlocking accounts, checking common corner cases and the likes
- As a user, I would like to have pagination of results
  - All list views of content include pagination
  - Pagination is seamless without freezing the UI
  - Pagination has added made the frontend more performant
  - UI is intuitive with regards to pagination
  - Tested and ensured robustness of pagination, checking common corner cases and the likes
- As a user, I would like to have access to even more ingredients from which to choose
  - Added over 25 more new ingredients to a growing list of over 800 ingredients
  - Added creation of the 25 new ingredients to the SQL database creation script
  - Ensured proper grammar of the new ingredients
  - Tested and ensured robustness of the new ingredients, checking common corner cases and the likes
- As a user, I would like to say if reviews are helpful or not
  - Created an intuitive frontend UI for marking and viewing ratings of reviews

    - Created backend handling of rating reviews
    - Backend processing of rating reviews all occurs in less than a second
    - Tested and ensured robustness of rating reviews, checking common corner cases and the likes
- As a user, I would like to filter the reviews based on rating
    - Created frontend UI for filtering reviews
    - Frontend UI filters reviews seamlessly without freezing
    - Filtering of reviews can be done by either helpful or not helpful
    - Tested and ensured robustness of filtering reviews, checking common corner cases and the likes
- As a user, I would like to have a profile rating that reflected the ratings of the recipes I have created
    - Created frontend UI to view profile ratings
    - Created backend handling of profile ratings
    - Processing of profile ratings all occurs in sub-second time
    - Tested and ensured robustness of handling profile ratings, checking common corner cases and the likes
- As a user, I would like to be able to find and sort users by their profile ratings
    - Created frontend handling of finding and sorting users by their profile ratings
    - Finding and sorting users by their profile ratings is seamless without freezing the UI
    - Backend loads down appropriate profile ratings in less than a second
    - Tested and ensured robustness of finding and sorting users by their profile ratings, checking common corner cases and the likes
- As a user, I would like to see recipes that I have made
    - Created frontend UI to view recipes created by user
    - Created backend handling of viewing recipes created by user
    - Backend loads recipes created by user in less than a second
    - Tested and ensured robustness of loading recipes created by user, checking common corner cases and the likes

What did not go well?

- Not realizing that certain acceptance criteria weren’t met until the last minute
    - For several user stories it was only discovered within the last week of development that certain acceptance criteria were not met or were not satisfactory

    - Other times acceptance criteria were misinterpreted and as such development needed to focus on correcting the misinterpreted acceptance criteria
- Aspects of the application that dealt with old functionality disappeared from the frontend and had to be re-implemented
    - Aspects that dealt with guest users (i.e. non-registered users) on the frontend seemingly disappeared from the frontend codebase and had to be reimplemented
    - Reimplementing these aspects posed the risk of introducing new bugs into the application and as such gave us much less runway to vet the app before the demo
- Had to slightly modify user stories as new issues presented themselves
    - Account unlocking happens completely outside the app, only locking an account occurs inside the app
    - Certain design decisions made some acceptance criteria need to be changed to a more nuanced view
- Miscommunication over expected behavior of certain functionalities of the app
    - Certain functionalities were implemented based on a misunderstanding of the expected behavior
    - Sometimes it was due to miscommunication and other times it was due to the lack of communication over the issue

How can you improve?

- **Continuous feedback on implementation of functions - Nick**
    - Continuously and immediately evaluate the newly implemented features
    - Communicate any issues found when evaluating the newly implemented features
- **Reviewing the acceptance criteria more often - Matt**
    - Always review the acceptance criteria and expected behavior as new features are implemented and enhanced
- **Peer review of code - Tyler**
    - Consistently review pull requests and commits
    - Offer code review prior to issuance of pull requests and commits
- **Establishing a more paced and consistent work pattern - Alex**
    - Spread out the workload as opposed to being more heavy-loaded on one end of the sprint
    - Ensure others are also following suit

Did we improve from last sprint?

- **More Standup Meetings/Consistency of Member Attendance**
    - Attendance and consistency of member attendance at standup meetings improved markedly this sprint
- **Running the App**
    - Build issues were virtually not an issue this sprint
- **Communication of issues**
    - Group members communicated a lot more frequently when they ran into issues this sprint
- **Buddy coding/coding sessions**
    - Buddy coding/coding sessions for several frontend components dramatically helped in reducing the number of last minute issues this sprint that had occurred in the previous sprint