Mixbook

Mixology Application

Sprint 3 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 required user stories and meet all non-functional requirements as well as set up what is needed for going forward into the future should we continue to develop this mobile application (which we plan on doing).

User stories:

- As a user, I would like to be able to recommend recipes to other users
    - Created frontend UI to handle recommending recipes as well as loading the recommendations that a user has received
    - Created frontend UI to delete recommendations that a user has received as well as the ability to search and filter the recommendations by various criterion
    - Created backend handling of recommending recipes, loading the recommendations that a user has received, as well as deleting recommendations that a user has received
    - Tested and ensured robustness of all-things-recommendations
- As a developer, I would like to overhaul the current error handling platform
    - Implemented app-wide error handling platform on the frontend
    - Standardized on the frontend the alert/error messages
    - Standardized and implemented app-wide, centralized logging of errors as well as ability to send error log to developers
    - Tested and ensured robustness of the new error handling platform
- As a developer, I would like to overhaul the interchange format of data to a more standardized version of JSON
    - Modified frontend to handle new standardized JSON format of responses and requests
    - Modified backend to make standardized JSON formatted responses as well as modified API endpoints to handle receiving requests that include JSON in a more standardized format
    - Modified backend to make any data marshalling that involved going to/from JSON occur in a streamlined and efficient manner
    - Tested and ensured robustness of the handling of the new interchange format of JSON data
- As a developer, I would like to remove the use of deprecated functions as much as possible
    - Removed/replaced any deprecated libraries/functions on the frontend

    - Modified logic on frontend where an equivalent function/library to the deprecated function/library was not found
    - Removed/replaced any deprecated libraries/functions on the backend
    - Modified logic on backend where an equivalent function/library to the deprecated function/library was not found
    - Tested and ensured robustness of the existing functionality with the deprecated functions entirely removed
- As a developer, I would like to overhaul the request handling to improve and make request handling more standardized
    - Modified frontend to make more RESTful requests as well as handle more RESTful responses
    - Modified frontend to handle standardized use of HTTP response codes and action verbs
    - Modified backend to make the API endpoints more RESTful
    - Modified backend to utilize proper, standardized HTTP response codes and action verbs
    - Tested and ensured robustness of the existing functionality with the new RESTful/HTTP specification standardization
- As a developer, I would like to redesign the add recipe page to be more user-friendly
    - Revamped frontend UI for adding a recipe to be more user friendly by removing a lot of the annoying popups
    - Revamped frontend UI for adding a recipe by making an ingredient list show so that a user can see what ingredients they have already added
    - Revamped frontend UI to make the recipe difficulty picker be a list of difficulties that make sense in English as opposed to showing a number between one and five
    - Revamped any page that views recipes to use the new difficulty picker as well as made the interaction options with recipes more intuitive and easier to navigate
    - Tested and ensured robustness of the new UI for all-things-recipes

What did not go well?

- As a user, I would like to receive push notifications for a variety of events (if time allows)
    - Local notifications were implemented however there was simply not enough time to implement push notifications

    - Implementing push notifications would have entailed a great many changes that could have had many confounding effects
    - Not enough research was done to figure out exactly how to implement push notifications and the best practices associated with push notifications
    - Failed to brainstorm enough events that would warrant push notifications to make it worthwhile
- As a user, I would like to see images of cocktails (if time allows)
    - There simply was not enough time to even start to handle viewing images on the frontend
    - The UI/UX design of the frontend did not allow for viewing images of cocktails without having to completely redesign the frontend
    - The backend would have had to undergo a significant number of changes that were not foreseen to handle the transmission of the images
- As a user, I would like to add and remove images of cocktails (if time allows)
    - There simply was not enough time to even start to handle adding and removing images of cocktails
    - A knowledge gap of how to implement this functionality would have made it very difficult to implement this properly
    - A major overhaul of the backend would have been required to handle this functionality which was not foreseen
- Not able to squash some minor bugs that we discovered in time
    - Some last minute bugs were discovered and were not sufficiently patched
    - Other bugs were simply too elusive to fix, almost like a Heisenbug
    - Bugs that existed in the libraries used on the frontend made patching some of these bugs next to impossible
- Build issues caused by some miscommunication over new frontend modules
    - Newly imported frontend libraries caused several group members to not be able to build the mobile application
    - It took a great amount of time to realize that a simple installation could have prevented the build error in the first place, thus wasting valuable development time
- At times blindly applying changes without thoroughly testing on frontend
    - Code on frontend was pushed without much if any testing at times and that caused an undue number of bugs to be introduced

    - Code that was pushed also indirectly caused other bugs and as such made it hard to trace the source of the issues
- Bugs were reported that really did not exist due to either missing changes due to not pulling new changes from the GitHub repository or simple environment issues
    - Environment issues caused odd bugs that were not actually application bugs and as such many hours were spent trying to track down nonexistent bugs
    - Not pulling down changes from the GitHub repository also caused duplicate fixes to bugs and as such slowed down the development process needlessly

How can you improve?

- **More consistent pushing and pulling of code - Alex**
    - Consistent pushing of new features and code fixes will allow for all users to maintain an up-to-date local repository and avoid merge conflicts
    - Consistent pulling of new code will ensure users are not reporting bugs that are already fixed as well as help to avoid merge conflicts
- **Peer review of code on a more consistent basis - Tyler**
    - Peer review code as soon as it gets pushed to ensure high quality code and eliminate bugs
    - Re-review code often as part of consistent code reviews to ensure that there is no oversight of bad code
- **More thorough testing of frontend code before committing to production repository - Matt**
    - Thorough testing of frontend code before committing to production will greatly cut down on the bugs that are in the new feature/fix or are caused by the new feature/fix
    - Thorough testing of frontend code before committing to production will also cut down on development time trying to track down bugs that could have been stopped from ever entering production
- **More communication over new libraries that are being used on frontend that involve environment changes in order to build the new frontend application - Nick**
    - Improved communication about changes to the codebase that require changes to the build environment will help to cut down on time spent figuring out why the mobile application will not build

    - Improved communication about changes to the codebase that require changes to the build environment will also help to ensure that everyone is aware of the ramifications of the changes themselves to the end product

Did we improve from last sprint?

- **Continuous feedback on implementation of functions - Nick**
    - Provided feedback on the quality of work helped to greatly ensure highly usable functionality
- **Reviewing the acceptance criteria more often - Matt**
    - Close inspection of the sprint planning document helped to significantly keep functionality within the realm of the acceptance criteria and helped to reduce development time as less redo's were necessary
- **Peer review of code - Tyler**
    - A great number of frontend and a few backend bugs were found and squashed through peer review of existing code as well as new code
- **Establishing a more paced and consistent work pattern - Alex**
    - A more steady pace of work allowed for a lot more time to test and debug the application, as well as ensure a quality product was available a couple days before the end of the sprint

Reflection on all sprints

- **The good:**
    - Successfully implemented all the required user stories and non-functional requirements that were planned across all three sprints
    - Improved upon every issue that occurred in each sprint
    - Worked well as a team with virtually no conflict/tension
    - Produced a well-tested and releasable product
    - Followed many development best-practices which will enable future development on the product
- **The bad:**
    - Did not manage to successfully implement the three optional (if time allows) stories
    - Had issues with communication among the group at various times
    - Introduced minor frontend bugs into production code at times due to poor testing practices

    - Had difficulty maintaining a consistent stand-up meeting schedule to due various last minute issues that arose among various group members
    - Ran into some build issues among various members on frontend due to a variety of reasons that slowed down development
- **The ugly:**
    - Clumsy copy/paste of existing frontend code to similar parts of the frontend created an undue number of bugs as well as obfuscated the codebase
    - Introduced major frontend bugs into production code at times due to poor testing practices
    - Introduced a couple major backend bugs into production code at times due to failing to test after doing something as simple as updating a library to a newer version
    - Somewhat sparse documentation on the frontend made it hard for other frontend developers to understand others code
    - Last minute realizations of bugs caused undue stress and burden at the worst possible times as a result of forgetting to do regression testing