Mixbook

Mixology Application

Sprint 3 Planning Document

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)

Sprint Overview: For the third sprint we will be working on recommendations of recipes, push notifications, and recipe images. By the end of the sprint users will be able to recommend a recipe. He or she will be able to view images of recipes (if time allows). Users will also receive push notifications (if time allows). Developers will also document the codebase as well as standardize/professionalize much of the codebase such as removing the use of deprecated functions, standardizing interfacing of frontend and backend, optimizing error handling, and more. Users will also experience a more user friendly add recipes page.

Scrum Master: John Tyler Preston

Meeting Schedule: Tue 4:30-5:30

Risks/Challenges: Designing an more user friendly user interface to provide a better user experience for adding new recipes will be challenging as none of the group members have much experience with UI/UX development. The bulk of this sprint will be revolving around standardizing and professionalizing the codebase, both on the frontend and the backend. Another difficulty will be attempting the stretch goal user stories such as push notification as those are technologies of which none of us have experience in implementing. Another difficulty that we will face will be adapting to changes to compensate for what did not go well in the second sprint. Overall, the third sprint will be a combination of overcoming our difficulties from the second sprint as well as standardizing/professionalizing our codebase while adding in a few new features.

User Story 1: **As a user, I would like to receive push notifications for a variety of events (if time allows).**

#	Task Description	Estimated Time	Owner
1	Implement handling push notifications on backend	10 hours	Tyler
2	Implement handling push notifications on frontend	15 hours (5 hours per team member)	Nick, Alex, Matt
3	Unit and usability testing	2 hours	Alex

Acceptance Criteria:

- ❏ Given that a recipe exists, when the user recommends a recipe to another user, then the user will receive a push notification about the recommended recipe.
- ❏ Given that a push notification has been sent, when the user opens a push notification, then the push notification will be cleared.
- ❏ Given that an action that warrants a push notification has been performed, once the action has been finished, then a push notification will be sent immediately to be processed by Google FCM.
- ❏ Given that a recipe exists, when the user recommends a recipe to another user, then the user will be able to open the recommended recipe from the push notification.
- ❏ Given that a push notification has been sent, when the user opens the app, then the notification will appear in a local notification list.

The frontend for this is mostly handling receiving push notifications and updating the display as needed. The backend mostly deals with creating, persisting, and handling the distribution of push notifications.

User Story 2: As a user, I would like to be able to recommend recipes to other users.

#	Task Description	Estimated Time	Owner
1	Implement the ability to recommend a recipe to another	10 hours	Nick

	user on the frontend		
2	Implement handling recommendations of recipes on backend	10 hours	Tyler
3	Unit and usability testing	5 hours	Nick

Acceptance Criteria:

- ❏ Given that a recipe exists, when a user recommends a recipe to another user, then they will receive a local notification.
- ❏ Given that a recipe exists, when a user opens up a recommended recipe, then the recommended recipe will load.
- ❏ Given that a recipe exists, when a user recommends a recipe to another user, then the other user will receive the recommended recipe in less than a second.
- ❏ Given a user recommended a recipe, when a user loads the recommended recipe, then the recipe will load in less than a second.
- ❏ Given a user does not exist, when a user attempts to recommend to a non-existent user a recipe, then they will be prompted that the user does not exist.

The frontend for this is mostly displaying a UI to recommend a recipe to another user as well as handling viewing the notification/recommended recipe. The backend is mostly responsible for persisting the recommendation as well as handling the request to make the recommendation.

User Story 3: **As a user, I would like to see images of cocktails (if time allows).**

#	Task Description	Estimated Time	Owner
1	Edit the recipe page to allow an image of the cocktail to be shown	8 hours	Matt
2	Retrieve the image from the server and display on the page	7 hours	Matt
3	Implement handling images on backend	5 hours	Tyler

4	Unit and usability testing	10 hours	Nick

Acceptance Criteria:

- ❏ Given that a recipe page is loaded, when the image is shown, then it must the correct image for that drink.
- ❏ Given that an image is loaded, when the image is shown, then it must fit the space that is allocated for it.
- ❏ Given that an image is unavailable, when the page is loaded, then a placeholder image will be shown.
- ❏ Given that an image is retrieved from the server, when it reviewed, then it should refer to the correct recipe.
- ❏ Given that a recipe page is loaded, when the image is shown, then load times for the page should not be affected in a meaningful way.

The image will be displayed on the frontend when the users loads a specific recipe page. The image will fill a set size and if no image is available then there will be a stock photo that is shown in place of it. The image will be stored on the backend and associated with the recipe to be loaded when the recipe is viewed.

User Story 4: **As a user, I would like to add and remove images of cocktails (if time allows).**

#	Task Description	Estimated Time	Owner
1	Implement ability to upload an image of a cocktail	7 hours	Alex
2	Implement the ability to edit/remove an image of a cocktail	6 hours	Alex
3	Handle on backend the adding and removing of images of cocktails	5 hours	Tyler
4	Unit and usability testing	5 hours	Nick

Acceptance Criteria:

- ❏ Given a cocktail doesn't have an image, when a user uploads an image, then the image will be displayed with the cocktail.

- ❏ Given a cocktail doesn't have an image, when a user uploads an image, then the image will be stored in the database.
- ❏ Given a cocktail has an image, when a user clicks remove image, then the image will no longer be displayed with the cocktail.
- ❏ Given a cocktail has an image, when a user clicks remove image, then the image will no longer be stored in the database.
- ❏ Given a cocktail has an image, when a user uploads a new image, then the new image will replace the existing image for the given cocktail.

This story involves associating a user-uploaded picture for each cocktail. This picture will be displayed on the recipe details page. It will have an option for the recipe owner to upload a new picture, replace an existing picture, and remove an existing picture.

User Story 5: As a developer, I would like to overhaul the current error handling platform.

#	Task Description	Estimated Time	Owner
1	Implement app-wide error handling platform on the frontend	7 hours	Alex
2	Standardize error alerts throughout the frontend application	7 hours	Alex
3	Implement central, standardized logging of errors on the frontend	7 hours	Alex
4	Unit and usability testing	6 hours	Alex

Acceptance Criteria:

- ❏ Given an error occurs, when it is of very low severity, then do not display an error dialog to the user.
- ❏ Given an error occurs, when it is of a higher severity, then display the error to the user in a standardized alert dialog.
- ❏ Given possible errors, when any error occurs, then record the details of the error in the error log.
- ❏ Given an error log exists, when the user requests support, then a support email will be created with the error log attached.

- ❏ Given an error log exists, when it is sent to the developers, then it can be used to debug the application.

This user story involves standardizing error messages throughout the front end application. This will assign a severity to errors, and triage them accordingly, displaying alerts to the user of high severity errors, and handling low severity errors without disturbing the user. All errors that occur will have its details recorded in a central log file. This log file will be sharable to the developers via a contact support button that will create an email with the error log attached.

User Story 6: As a developer, I would like to overhaul the interchange format of data to a more standardized version of JSON.

#	Task Description	Estimated Time	Owner
1	Handle new standardized responses from the server	5 hours	Matt
2	Unit and usability testing	5 hours	Matt
3	Implement more standardized interchange of JSON on backend to extent possible	10 hours	Tyler

Acceptance Criteria:

- ❏ Given that data is tied to a user or recipe, when it is connected, then a key:value pair should be used.
- ❏ Given that variables have been named without camelcasing, when they are found, then the naming must changed to be camel cased.
- ❏ Given that requests are formatted inconsistently, when they are found, then a standard format should be used.
- ❏ Given that variables have been poorly named, when they are found, then the naming should be changed to be informative.
- ❏ Given that all inconsistencies have been fixed, when the app is the run, then app should function normally.

The biggest part of this user story will be spent going through our code and checking for inconsistencies in naming, requests, and relations between objects. The only way to do so will be to go through the code line by line and check for issues. When an issue is found we will fix and document the changes.

User Story 7: As a developer, I would like to remove the use of deprecated functions as much as possible.

#	Task Description	Estimated Time	Owner
1	Go through list of functions and determine which ones, if any, are deprecated	10 hours	Matt
2	Find new alternatives that will accomplish the same task that are not deprecated	5 hours	Matt
3	Replace deprecated functions with the alternatives found	10 hours	Matt
4	Refactor and review backend for use of deprecated functions	10 hours	Tyler
5	Unit testing and usability testing	5 hours	Matt

Acceptance Criteria:

- ❏ Given that there are deprecated functions on the front end, when one is found, then an alternative must be researched.
- ❏ Given that an alternative is found, when it is implemented, then it should do the same or similar task to function its replacing.
- ❏ Given that an alternative has been implemented, when it is tested, then it should have the same result as the deprecated function.
- ❏ Given that there deprecated function on the back end, when one is found, then an alternative must be researched.
- ❏ Given that an alternative is found on the back end, when it is implemented, then it should do the same or similar task to function its replacing.

The bulk of this work on the front end will be going through the package.json file and seeing which dependencies are deprecated. Once found alternatives will need to be found and adapted to eliminate the deprecated calls.

User Story 8: As a developer, I would like overhaul the request handling to improve and make request handling more standardized.

#	Task Description	Estimated Time	Owner

1	Implement requests to and from the server in a standardized RESTful/HTTP fashion	7 hours	Alex
2	Overhaul backend to use more standardized requests	10 hours	Tyler
3	Unit and usability testing	6 hours	Alex

Acceptance Criteria:

- ❏ Given a request need to be made, when the request is made, then the HTTP verb of the request will conform to industry best practices and standards.
- ❏ Given a request is made, when the server returns a successful response, then the response will contain a HTTP response code in the 2XX range that conforms to industry best practices and standards.
- ❏ Given a request is made, when the server encounters an error caused by the server itself, then the server response will contain a HTTP response code in the 5XX range that conforms to industry best practices and HTTP standards.
- ❏ Given a request is made, when the server encounters an error caused by the client sending the request, then the server response will contain a HTTP response code in the 4XX range that conforms to industry best practices and HTTP standards.
- ❏ Given a request is made, when the server returns a response, then the response headers will conform to industry best practices and HTTP standards.
- ❏ Given a request is to be made, when the request is made, then the request headers will conform to industry best practices and HTTP standards.

This story will involve making sure all request to and from the server will conform to HTTP standards and industry best practices. This includes sending the correct HTTP verb, returning a correct HTTP response code, and having the headers of both the request and the response be valid.

User Story 9: As a developer, I would like redesign the add recipe page to be more user-friendly.

#	Task Description	Estimated Time	Owner
1	Redesign Add Recipe Page	5 hours	Nick

2	Implement New Add Recipe Design	15 hours	Nick
3	Unit and usability testing of new page	5 hours	Nick

Acceptance Criteria:

- ❏ Given the user is adding a recipe, when they open the page, then it must be more user friendly.
- ❏ Given the user is adding a recipe, when they add ingredients, they should be able to see what ingredients they have already added.
- ❏ Given the user is adding a recipe, when they rate the difficulty of a recipe, then they should be given more feedback on what rating they are given.
- ❏ Given the user is adding a recipe, when they add a recipe, then the performance of the page should be better.
- ❏ Given the user is adding a recipe, when the user adds a recipe, then the page should be streamlined and less confusing.

The add recipes page has long been confusing, slow and just not easy to use. This user story should clean it up, improve performance and make it easier to use. It should also display all of the relevant data to the user such as the ingredients that have been added which is not currently happening.

Remaining Product Backlog:

Functional

- None (all have been completed)

Non-Functional

- None (all have been completed)

Time Allocation of Work Per Group Member + Total:

Alex's Hours	Matt's Hours	Tyler's Hours	Nick's Hours	Total Hours
40 (60 if including "if time allows" user stories)	40 (60 if including "if time allows" user stories)	40 (60 if including "if time allows" user stories)	40 (60 if including "if time allows" user stories)	160 (240 if including "if time allows" user stories)