Team 8 Sprint 1 Retrospective Document: Kadence

Jackson Rosenberg, Avery Schaefer, Nathan Simon, Colston Streit, Raymond Xie
rosenbe7@purdue.edu, schaef35@purdue.edu, simon70@purdue.edu,
cstreit@purdue.edu, xie328@purdue.edu

What Went Well:

We believe that overall, this sprint was very successful. The main objective of this sprint was to lay a solid foundation for the project that could be built upon in sprints 2 and 3. To do this, we decided to focus on the fundamental technologies used in the project (user stories 1-4, 10, 11), user account creation and setup (user stories 5, 7-9) and linking music platform accounts (user story 6). We believe that this went extremely well. We were able to accomplish everything we wanted to with these user stories. After this sprint, users are now able to create, log into and update their account, as well as pair a spotify and/or apple music account. This sets us up very well for sprint 2.

User Story #1:

As a developer, I would like a MongoDB database connected to the backend.

#	Description	Estimated Time	Owner
1	Read MongoDB Query Documentation	3 hours	Avery
2	Create and format a MongoDB database	2 hours	Avery
3	Test CRUD operations on database using mock data and Postman	3 hours	Avery
4	Link MongoDB database to Node/Express backend	3 hours	Avery
Total User Story Time		11 Hours	

Completed:

A MongoDB database was created for the project, as well as four collections within the database for different types of objects that will be stored for the project. The database is able to accept CRUD requests from postman, as well as the Kadence app itself.

User Story #3: KAD-69

As a developer, I would like a functional Node/Express backend.

#	Description	Estimated Time	Owner
1	Research and familiarize ourselves with how to use Node and Express.	2 Hours	Avery, Raymond, Jack
2	Create a runnable Node/Express backend server with an organized file structure.	6 Hours	Raymond
3	Sketch out a list of all the API endpoints that we want our backend to expose to our frontend.	3 Hours	Avery
4	Implement a skeleton version of each of these endpoints that returns some placeholder data (to be filled in later when functionality is added).	8 Hours	Jack
Total User Story Time		23 Hours	

Completed:

A functional node.js backend has been created for the project, and is able to send/receive functional HTTP requests, as well as parse information it receives. A thorough API documentation README has been created to track request and response parameters for the endpoints we created.

User Story #4: KAD-71

As a developer, I would like to integrate the frontend and backend.

#	Description	Estimated Time	Owner
1	Add functionality to the relevant UI components to make requests to each of the backend API endpoints.	9 Hours	Colston
2	On the frontend, display the data returned by the backend for each API endpoint in some way.	5 Hours	Colston
3	Implement preliminary error handling for each of the backend endpoints, as well as for each of the requests made from the frontend.	4 Hours	Colston
4	Test that all of the backend API endpoints are reachable from the frontend and return the expected placeholder data.	6 Hours	Colston
Total User Story Time		24 Hours	

Completed:

The frontend and backend have been successfully integrated in our project, and the frontend can now successfully call our backend API endpoints, as well as send information to it and request information from it. Additionally, basic error handling has been implemented on both the frontend and backend for each API call, ensuring that only valid requests are accepted by the backend API.

User Story #5: KAD-5, 6, 7, 8

As a user, I would like to register an account for Kadence, log-in, and delete my account if desired.

#	Description	Estimated Time	Owner
1	Create tables in the database for storing user login information and implement security features / hashing algorithm for properly storing passwords	6 hours	Jack
2	Create algorithm to receive login information from database and check entered password	8 hours	Avery
3	Develop forgotten password retrieval functionality	5 hours	Jack
4	Develop password strength checking algorithm	2 hours	Jack
5	Create unit tests for checking password strength	2 hours	Jack
6	Design relevant UI components to assist registering for new accounts	3 hours	Colston
7	Design relevant UI components to assist logging in and logging out of already made and stored accounts	3 hours	Colston
8	Add querying in order to remove accounts from the database on request	2 hours	Jack
9	Create manual testing sequences for account management	2 hours	Jack
Total User Story Time		33 Hours	

Completed:

Users are now able to create an account, which persists in the MongoDB database, as well as log in to an already existing account. They are also able to delete their account from the database if they choose to, and will not be able to log in to the deleted account.

User Story #6: KAD-12, 13, 14, 15

As a user, I would like to link, update, or remove my music platform of choice within the app.

#	Description	Estimated Time	Owner
1	Use the Spotify and Apple Music APIs to connect and retrieve data about user accounts (with a focus on third party account authentication)	11 hours	Jack
2	Design relevant UI components to connect a new platform upon registering	4 hours	Nathan
3	Design relevant UI components with proper formatting to display the current connected account	4 hours	Nathan
4	Allow users to remove current connected platform	1 hour	Avery
5	Allow users to update current connected platform	1 hour	Avery
6	Allow users to leave Kadence and access their specified music platform	2 hours	Raymond
Total User Story Time		23 Hours	

Completed:

Users are able to navigate to a frontend page that allows them to link their spotify or apple music accounts to their Kadence account. Additionally, users are able to update and remove a connected platform if they choose.

User Story #7 (UPDATED 2/22)): KAD-39, 40, 41, 42, 43, 44, 45

As a user, I would like to specify my preferences for the different playlist generation modes.

#	Description	Estimated Time	Owner
1	Design layout of page used to update playlist generation preferences	6 Hours	Nathan
2	Create backend API endpoint for updating preferences documents	4 Hours	Avery
3	Design relevant UI components to the “update playlist preferences” form	5 Hours	Nathan
4	Create functionality to allow users to update preferences	3 Hours	Nathan
5	Test the “preference update” form and functionality	3 Hours	Raymond
Total User Story Time		21 Hours	

Completed:

Users are now able to navigate to a page - once logged in to the app - that allows them to update the preferences for their account. These changes persist in the database, modifying the “preferences” document associated with their user account.

User Story #8: KAD-10, 11, 65, 66, 67

As a user, I would like to have and be able to update a profile.

#	Description	Estimated Time	Owner
1	Design layout of profile page.	6 Hours	Nathan
2	Create a form to request user information upon signing up for the app.	4 Hours	Nathan
3	Create a backend request to get user info.	4 Hours	Nathan
4	Link form with the backend request	3 Hours	Nathan
5	Allow users to update information on their profile.	3 Hours	Nathan
6	Create manual tests to ensure profile updating is properly tested	4 Hours	Avery
Total User Story Time		24 Hours	

Completed:

Users are now able to create a profile upon creating an account, setting up all the required documents on the database. Also, they are able to view their profile information on a dedicated frontend page, and update this information in a way that persists on the backend and in the database.

User Story #9: KAD-17

As a user, I would like to be able to update my default music and device preferences.

#	Description	Estimated Time	Owner
1	Expand upon design of user settings page from mockup	3 Hours	Colston
2	Create form UI for default music preferences page	2 Hours	Colston
3	Create form UI for default device preferences page	2 Hours	Colston
4	Implement storage of default music preferences in database	3 Hours	Colston
Total User Story Time		10 Hours	

Completed:

Backend endpoints have been created that allow music platform and device information documents within the database to be updated, as well as frontend pages that gather required data for these documents and send that information to the backend endpoints.

User Story #10: KAD-73

As a developer, I would like the repository to have a Continuous Integration pipeline.

#	Description	Estimated Time	Owner
1	Research Docker and containerize the back-end app with Docker	6 Hours	Raymond
2	Set up ESLint to check for code quality	2 Hours	Raymond
3	Create a Github Action workflow to automatically build the app upon PR creation	4 Hours	Raymond
4	Add a step to the above workflow that automatically runs the test suite upon PR creation	6 Hours	Raymond
Total User Story Time		18 Hours	

Completed:

A Continuous Integration pipeline has been created using GitHub Actions that builds the app on multiple versions of Node, completes Docker containerization, runs ESLint and runs a Jest test suite on the creation of a pull request within GitHub.

User Story #11: KAD-73

As a developer, I would like a way to automatically deploy the back-end API with a Continuous Deployment pipeline.

#	Description	Estimated Time	Owner
1	Create a Github Action workflow that pushes the app image to a container registry upon merging a branch	4 Hours	Raymond
2	Test the manual creation of an Azure Container Instance running our API	2 Hour	Raymond
3	Set up a continuous deployment pipeline in Azure	3 Hours	Raymond
Total User Story Time		9 Hours	

Completed:

An Azure Container Instance has been created for our project, and has been implemented within our CICD pipeline that automatically pushes an updated docker image to the Azure registry upon a pull request with the “main” branch.

What Did Not Go Well:

Overall, the main thing that we struggled with during this sprint was fitness device connection. It took us a great deal of time to purchase and subsequently acquire the smartwatch we chose for the project, and by the time we had the watch in our possession, there was not enough time left in the sprint to accomplish everything allocated for this sprint with the device. Thus, the associated user stories with it have been pushed back to sprint 2. Additionally, we faced some issues with XCode, and subsequently building the app on iOS.

User Story #2:

As a developer, I would like a functional Next.js frontend.

#	Description	Estimated Time	Owner
1	Research and familiarize with Next.js framework	4 hours	Avery
2	Create a blank Next.js frontend	2 hour	Avery
3	Connect Capacitor to blank Next.js frontend	2 hour	Avery
Total User Story Time		8 Hours	

Not Completed:

A Next.js frontend has been created for the project. We also have been able to connect Capacitor to this frontend, so the project can be compiled on Android devices. However, Apple requires using XCode, an IDE that is only available on Mac computers, to build an iOS-compatible image of the app. Only one member of the group - Avery - has a Mac, however his computer was facing issues getting XCode to open the project repository, and thus was unable to fully test if iOS compilation works. However, the support for it has been added to the project, it just cannot be tested at this time.

User Story #7 (FROM ORIGINAL SPRINT PLANNING DOCUMENT): KAD-18, 20, 21, 22, 23, 68

As a user, I would like to link, update, and delete my fitness watch in the app.

#	Description	Estimated Time	Owner
1	Use bluetooth to identify nearby devices.	4 Hours	Raymond
2	Use the Huawei API to connect to and retrieve data from the device.	6 Hours	Avery
3	Create a default function that will select the preferred device if two are detected.	2 Hours	Avery
4	Design relevant UI components to connect a new device registering	5 Hours	Nathan
5	Design relevant UI components with proper formatting to display the current connected devices	4 Hours	Nathan
6	Create functionality to allow users to update and delete devices	3 Hours	Nathan
7	Test device connection and functionality	3 Hours	Raymond
Total User Story Time		27 Hours	

Not Completed:

As described earlier, we were not able to get the smartwatch required to complete this user story until the last week of the sprint, at which time we decided there was not enough time left to complete the user story. Because of that, we decided to push this user story back to sprint 2.

How We Will Improve:

For this sprint, we will be able to fix the primary issue of not having the smartwatch needed to complete the original user story 7. With the device in our possession and more time available to us, we will be able to complete that user story during the upcoming sprint. Additionally, we have more familiarity with the frameworks we are using for this project now than we did at the beginning of the sprint, so we will be able to spend less time learning them and troubleshooting issues throughout the upcoming sprint, allowing us to spend more time working on user stories. Finally, we know more about each other's development styles, as well as our strengths and weaknesses. This way, during planning for sprint 2, we can assign everyone tasks that best align with their strengths and let everyone work on what they're best at.