Sprint 2 Retrospective

Team 18

Sai Edupulapati, Akshaya Kumar, Shreya Sanapala, Visa Thongdee, Vidya Vuppala

What Went Well

After having set up the preliminary UI in Sprint 1, we were able to focus on some of the core functionalities of Wander including travel features, like entering a starting location and destination for a user's trip. This information, coupled with a newly restructured quiz, allowed for us to implement a playlist generation algorithm. In addition to these features, we implemented ways for the user to view their past quiz selections and past trips. We also majorly overhauled the UI and navigation of the website to create a better user experience.

User Story 1: As a user, I would like to be able to delete my profile picture.

#	Description	Estimated Time	Owner
1	Update UI to 1) add button for deleting picture and 2) display necessary prompts upon success	1 Hr	Akshaya
2	Implement appropriate logic for deletion of profile picture in Firebase and Firestore	1 Hr	Akshaya

3	Debug delete functionality and test with unit testing	2 Hr	Akshaya

Completed:

After creating a profile, the user is able to navigate to the 'Edit Profile' page. Within that page, they are given two button options regarding their profile picture: 'Edit Profile Picture' and 'Delete Profile Picture'. If the user wants to delete their profile picture, they can click on that button. Upon doing so, they are shown a popup that asks them to confirm their decision. If the user decides not to delete their profile picture after viewing the popup, they can click the 'X' button or on the blurred background behind the popup. No changes are made to the user's profile picture. However, if the user does decide to delete their profile picture, the popup goes away and the UI is immediately updated to show the user's profile picture as the default, null image. The Firebase backend is also updated; the user's profile picture information is set to null and the image file is deleted from Firebase storage.

User Story 2: As a user, I would like to be able to edit my account information (ex. demographic, music preference, etc.) as needed.

#	Description	Estimated Time	Owner
1	Modify Settings/Edit Profile UI to include 'Edit' buttons and textfields to edit account information	2 Hrs	Visa
2	Implement appropriate logic for updating account information	3 Hrs	Visa
3	Update UI to reflect edited account information	2 Hrs	Visa
4	Testing and Debugging	2 Hrs	Visa

Completed:

When the user would like to edit their account information, they are able to go to the Edit Profile page, which can be accessed from their profiles, in order to alter their account information which includes demographic information. Having added two new fields for demographic information

(e.g. age and gender) which will be prompted for in the profile set up, it can also be edited in the Edit Profile page. When the user changes their information and clicks submit, their changes will be reflected in the website database in the backend.

User Story 3: As a user, I would like to be able to view my listening data (total minutes, genres, artists).

#	Description	Estimated Time	Owner
1	Set up appropriate front end format for displaying the data	4 Hr (because he has no experience with frontend)	Sai
2	Gather the data mentioned above from Spotify Web API	2 Hrs	Sai
3	Set up necessary structure in Firebase backend to store data in an organized way	2 Hrs	Sai
4	Recall data from backend and display onto the appropriate page on the website	1 Hr	Sai
5	Debug and Test (connecting/disconnecting, etc.)	1 Hr	Sai

Completed:

When the user grants spotify access now, their authorization is persistent until they click disconnect, even after leaving the page. Using this access token, the website is able to display information such as their top tracks and top artists. With the use of the access tokens, it reduced overhead in storage and number of reads, as simply storing the access tokens made it possible for us to recall this information through XML/HTTP class instead.

User Story 4: As a user, I would like to be able to retake the quiz for different results.

#	Description	Estimated Time	Owner

1	Change structure of database to implement time based entries and/or quiz number to allow for quiz history	4 Hrs	Shreya
2	Change quiz UI structure so that there are Next and Previous button with quiz selections rather than automatically going to the next question upon answer selection	4 Hrs	Shreya
3	Change UI for quiz so that user's past quiz selections are populated in the quiz when they click the previous/start over buttons	4 Hrs	Shreya
4	Debugging and Testing	3 Hrs	Shreya

Completed:

When the user gets to the quiz portion of the Trip creation process, they are greeted with a restructured UI that remembers their answer selection while navigating between questions. If the user attempts to go on to the next question without answering, they receive an alert that ensures that they pick an answer. When the user submits their quiz selections, their answers are successfully stored in Firebase.

User Story 5: As a user, I would like to be able to see my past music preferences (i.e. past answers to quizzes, past results).

#	Description	Estimated Time	Owner
1	Modify profile UI to include past quiz history button	1 Hr	Vidya
2	Create a new page and skeleton UI for past quiz history page	1 Hr	Vidya
3	Connect quiz answers to newly structured database (Refer to User Story #4, Task 1)	1 Hr	Shreya
4	Connect quiz answers database to profile page UI	3 Hrs	Shreya
5	Debugging and Testing	3 Hrs	Shreya

Completed:

When the user navigates to the profile page, they can click a button to view a page with all of their past quiz selections. Here, all of their quiz history is displayed in chronological order. If there are no quizzes associated with the user in the database, the page will indicate that the user has no past quizzes.

User Story 6: As a user, I would like to be able to reset my quiz history.

#	Description	Estimated Time	Owner
1	Design an algorithm that locates all of the quiz entries that are made by a certain user and deletes them	4 Hrs	Shreya
2	Create aesthetic button on past quiz history page to delete past quiz history	0.5 Hrs	Vidya
3	Connect the delete past quiz history button to the backend	1 Hr	Shreya
4	Update the UI of the past quiz history page to reflect the deleted quizzes	2 Hrs	Shreya
5	Debugging and Testing	3 Hrs	Shreya

Completed:

If the user wishes to erase their quiz history from the database, they are able to navigate to the past quiz history page in the profile page and select the Delete History button. After this button is selected, none of the user's past quizzes will be populated on the screen and they are erased from the database. If the user has no past quizzes, or has already deleted their quiz history, the Delete History button will be disabled.

User Story 7: As a user, I would like to be able to toggle on preferences for a playlist (ex. seasonal, genre, mood, location).

#	Description	Estimated Time	Owner

1	Create UI for playlist for toggles	1.5 Hrs	Vidya
2	Connect backend to store user preference	2.5 Hrs (because she's currently inexperienced in Firebase)	Vidya
3	Edit and format settings page to accommodate toggle	2 Hr	Vidya
4	Design UI for toggle and updated setting page	1 Hr	Vidya
5	Testing and Debugging	2 Hrs	Vidya

Completed:

When the user wants to set their preferences (dark vs light mode, wanting explicit content in songs), they can go to the settings page and see a toggle UI that lets them set their preferences. The user can press on these toggles to state their preferences. Then, this information is stored in our firebase database and is updated as the user makes changes to their preference.

User Story 8: As a user, I would like to be able to enter a starting location.

#	Description	Estimated Time	Owner
1	Create UI to make user trip	3 Hr	Vidya
2	Add textfield to enter starting location	0.5 Hrs	Vidya
3	Connect UI to backend to store user input	1 Hr	Akshaya
4	Complete in-depth research on how to incorporate autocomplete tool to give user suggestions and implement tool	6 Hr	Akshaya
5	Testing and Debugging	2 Hrs	Akshaya

Completed:

When a user wants to create a new trip, they can navigate to the correct page by clicking on the 'Create New Trip' button on the Home page. In that page, they are shown a Textfield where they

can enter their desired starting location. When the user begins typing in that Textfield, a dropdown of 5 suggestions appears below the Textfield. The suggestions are all autocomplete locations based on the user's initial input. The user can select one of the suggestions or continue typing to receive more suggestions. When the user clicks the submit button at the bottom of the page, the autocomplete starting location suggestion is stored in the Firebase database and is linked to the user. Upon successful trip creation, the user is navigated to the first Quiz page.

User Story 9: As a user, I would like to be able to edit a starting location.

#	Description	Estimated Time	Owner
1	Create UI to be able to edit journey with editable text fields	3 Hrs	Visa
2	Add a text field for editing the starting location	0.5 Hrs	Visa
3	Connect UI to backend to update starting location	1 Hr	Akshaya
4	Incorporate autocomplete to suggest locations	1 Hr	Akshaya
5	Add button to be able to navigate back to trip view	0.5 Hrs	Visa
6	Testing and Debugging	1 Hr	Akshaya

Completed:

When a user wants to edit their starting location of their current trip, they can click on a card to visit a page called "Edit Trip". This lets the user change their starting location. As the user types in letters to indicate their location, they can see autocomplete suggestions to help complete this form. This new starting location is stored in our database and the user can navigate back to the trip view by clicking on the "Edit Changes" button. This button stores the information in the database and navigates back to the home page

User Story 10: As a user, I would like to be able to enter my destination.

#	Description	Estimated Time	Owner
1	Create button for new trip in the homepage	1 Hr	Vidya
2	Create a quiz UI page to enter destination location	1 Hr	Vidya
3	Connect text field to an autocomplete API with maps location	1 Hr	Akshaya
4	Save user input/location in their account's Firebase backend	1 Hr	Akshaya
5	Testing and Debugging	1 Hr	Akshaya

Completed:

The functionality for this User Story is virtually identical to User Story 8. When a user wants to create a new trip, they can navigate to the correct page by clicking on the 'Create New Trip' button on the Home page. In that page, they are shown a Textfield where they can enter their desired destination location. When the user begins typing in that Textfield, a dropdown of 5 suggestions appears below the Textfield. The suggestions are all autocomplete locations based on the user's initial input. When the user clicks the submit button at the bottom of the page, the autocomplete destination location suggestion is stored in the Firebase database and is linked to the user. Upon successful trip creation, the user is navigated to the first Quiz page.

User Story 11: As a user, I would like to be able to edit my destination.

#	Description	Estimated Time	Owner
1	Create UI to be able to edit destination location with editable text fields	3 Hrs	Visa
2	Connect UI to backend to update destination location	1 Hr	Akshaya
3	Incorporate autocomplete to suggest locations	1 Hr	Akshaya

4	Add button to be able to navigate back to trip view	0.5 Hrs	Visa
5	Testing and Debugging	1 HR	Akshaya

Completed:

When a user wants to edit their ending location of their current trip, they can click on a card to visit a page called “Edit Trip”. This lets the user change their starting location. As the user types in letters to indicate their location, they can see autocomplete suggestions to help complete this form. This new ending location is stored in our database and the user can navigate back to the trip view by clicking on the “Edit Changes” button. This button stores the information in the database and navigates back to the home page

User Story 12: As a user, I would like to be able to set and edit my travel preference (ex. car, plane) for each trip.

#	Description	Estimated Time	Owner
1	Create a UI with text field to set travel preferences	1 Hr	Vidya
2	Create a separate UI with editable text field to allow changes to travel preferences	1 Hr	Visa
3	Store travel preferences in Firebase backend	1 Hr	Akshaya
4	Update changes in travel preferences in Firebase backend	1 Hr	Akshaya
5	Testing and Debugging	1 Hr	Akshaya

Completed:

When a user wants to edit their travel preferences of their current trip, they can click on a card to visit a page called “Edit Trip”. This lets the user change their preferences. The user can choose an option of car vs plane on this page. This new preference is stored in our database and the user

can navigate back to the trip view by clicking on the “Edit Changes” button. This button stores the information in the database and navigates back to the home page.

User Story 13: As a user, I would like to be able to navigate through the website and access every appropriate page.

#	Description	Estimated Time	Owner
1	Create page for *every* feature with base features, color scheme, and corresponding aesthetic	7 Hrs	Vidya, Visa
2	Add the appropriate links/buttons to *every* page for navigation	2 Hrs (each) = 6 hours	Vidya, Visa, Akshaya
3	Add routing for every page	4 Hrs	Akshaya
4	Debug and test routing with unit tests	3 Hrs	Akshaya

Completed:

There now exists a page for every feature we plan to implement in the final sprint with the appropriate buttons/links that navigate every page to one another. This includes an alternate homepage for new users, a past quiz history page, a past trips page, a quiz confirmation page, a loading page, an expanded trip view page (with source location, destination, and trip playlist), and a page to edit source and destination locations for trips. Along with these pages, we also added appropriate navigation and routing to connect and unify the whole website.

User Story 14: As a user, I would like to be able to see my past curated playlists.

#	Description	Estimated Time	Owner
1	Create skeleton UI to display past playlists	2 Hrs	Vidya, Visa
2	Set up Firebase backend to handle storage of playlists by user	2 Hrs	Shreya, Sai
3	Create buttons to delete and clear history	0.5 Hrs	Vidya, Shreya

4	Set up Firebase logic to remove associated user history	2 Hrs	Shreya, Sai
5	Set up Firebase to retrieve history based on time intervals (12M ago, 6M ago, 3M ago)	3.5 Hrs	Shreya, Sai
6	Testing and Debugging	2 Hrs	Shreya, Sai

Completed:

The past trips page displays all the trips that the user has had in the past on separate cards, showing that trip's date, source location, destination, and embedded playlist with links to the expanded view of each component which is pulled from the user's Firebase backend. Additionally, there is a drop down menu that can filter the past trips within certain time intervals such as one year, 6 months, and 3 months. When a user would like to delete or clear their history, there is also a button that allows them to do so which will also remove the associated user history in the Firebase backend.

User Story 15 (Sprint 1 User Story 12): As a user, I would like to get a music playlist for my journey.

#	Description	Estimated Time	Owner
2	Create playlist in the server using information from the database and Spotify API	2 Hr	Shreya
3	Store current playlist in the database	3 Hr	Shreya
6	Debug and test algorithm with unit testing	2 Hrs (each)	Shreya, Vidya

Completed:

After granting access to their Spotify Account, the user is able to see a "Generate Custom Playlist" button on the results page after completing the quiz. Upon clicking this button, they are made a random playlist based on the preferences they specified in the quiz (currently based on favorite genre, top artists, and travel time). Additionally, similar to authentication, simply storing

the playlist ID in Firebase has been enough for us to access it for future use/records/to display it to the “Past Trips” page.

What Did Not Go Well

Based on our explicit acceptance criteria, we were able to achieve all of our Sprint 2 goals. The aspects of this sprint that did not go well had more to do with our process and approach to this sprint (i.e. dealing with a learning curve, time management when it came to debugging, etc). Those difficulties didn't necessarily manifest as failed User Stories or acceptance criteria. That being said, one user story that we feel we could have gone above and beyond with was User Story 12, which deals with custom playlist generation.

User Story 15 (Sprint 1 User Story 12): As a user, I would like to get a music playlist for my journey.

#	Description	Estimated Time	Owner
2	Design algorithm to implement custom playlist generation based on user preferences	10 Hrs	Shreya, Sai, Vidya

Not Completed:

Though we did design an algorithm that created a custom playlist, we could optimize the algorithm further by weighing the user's quiz answers and preferences more heavily in the final song selection. More time and research is necessary to create the most robust algorithm possible.

How We Should Improve

This sprint happened to fall during the busiest time of the semester for most of our team members. Balancing homework for other classes, preparing for finals/midterms, and making time for our social commitments on top of prioritizing the work required for this class was difficult for some of us. This issue is one that can be fixed with better time management and making more of an effort to check in with each other and hold each other accountable for making incremental progress. Now that the majority of our midterms are over, this time management issue should be resolved or at the very least, easier to deal with.

Another obstacle we faced was properly dividing tasks. There would be tasks with multiple people working on the same file. This eventually caused merge conflicts, confusion about who was doing what and general dysfunction. In the future, it would be better to give a single user story to one to two people, so there isn't much confusion. It would also be beneficial to group user stories together and give people user stories that are most similar to reduce the probability of merge conflicts and other issues.

One of the challenges we faced this Sprint was having to manage the amount of reads that our application was making from Firebase. Due to the nature of the Spotify API and Firebase, it was difficult to create an efficient algorithm that did not use up the daily free quota of reads from Firebase. Because we were not aware of how many reads we were generating, we accidentally used up all of our daily quota and could not work on the project or even view the database for around half a day, halting our progress for some time. In the future, we will be more aware of the daily Firebase quota and try to design our algorithms to be as efficient as possible. We will also research and examine ways to improve our usage and determine the scope of the APIs we incorporate in our application.

Lastly, an issue that we encountered was regarding the new requirement for a testing document this sprint. Since there was no clear format, set of specifications, or example that was available to us until only a couple days before it was due, our group had trouble starting and creating a formal testing document. Therefore, we had issues with figuring out the requirements, formatting, and due date of the testing document which delayed our progress in our project and

did not have time to complete automated testing causing us to only complete manual testing for each user story instead.