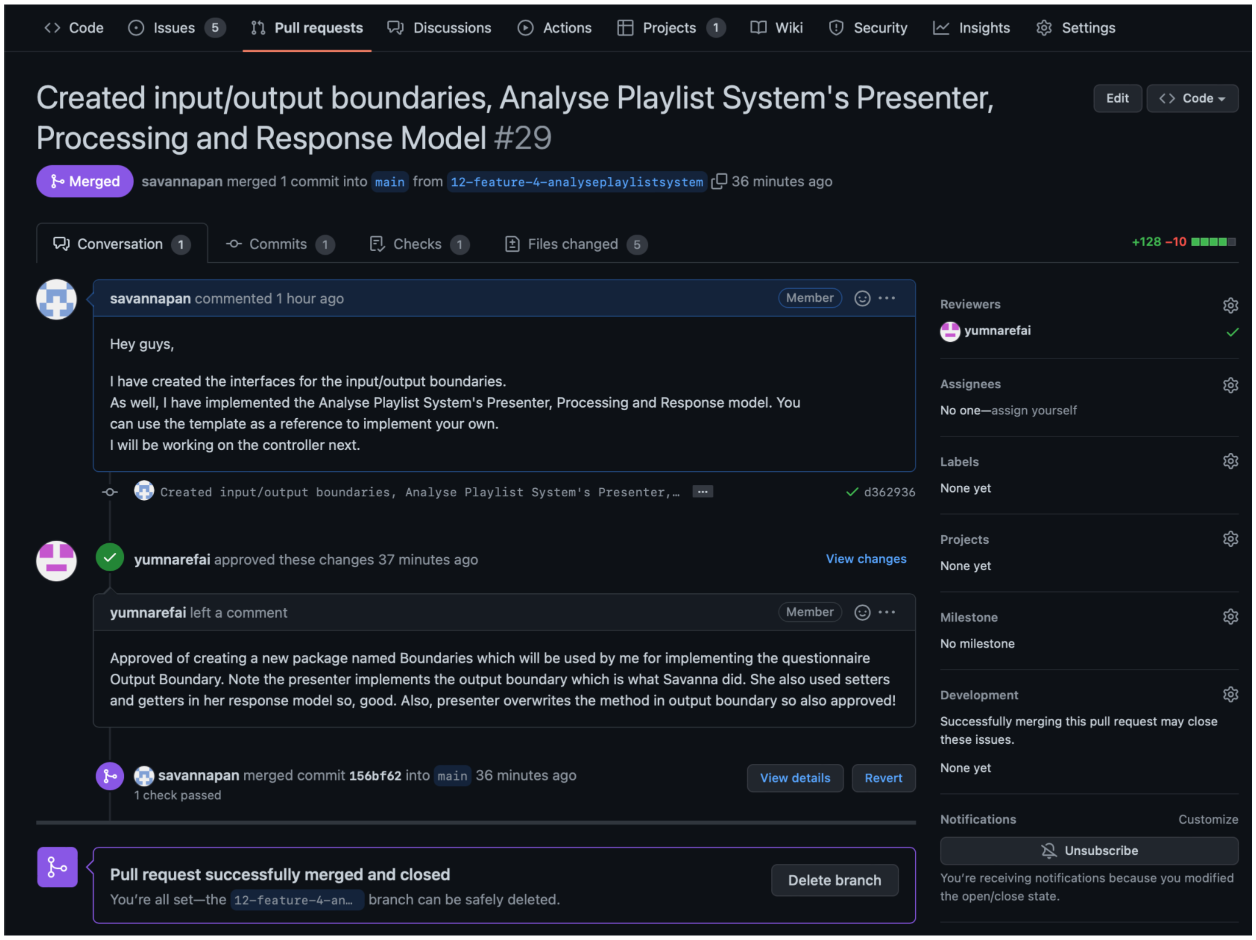
Milestone 4

Group 120

Savanna’s Pull Requests

Code Contribution:



Created input/output boundaries, Analyse Playlist System's Presenter, Processing and Response Model #29

Merged savannapan merged 1 commit into `main` from `12-feature-4-analyseplaylistsystem` 36 minutes ago

Conversation 1 · Commits 1 · Checks 1 · Files changed 5 · +128 −10

savannapan commented 1 hour ago · Member

Hey guys,

I have created the interfaces for the input/output boundaries.
As well, I have implemented the Analyse Playlist System's Presenter, Processing and Response model. You can use the template as a reference to implement your own.
I will be working on the controller next.

`Created input/output boundaries, Analyse Playlist System's Presenter,…` ✓ d362936

yumnarefai approved these changes 37 minutes ago · View changes

yumnarefai left a comment · Member

Approved of creating a new package named Boundaries which will be used by me for implementing the questionnaire Output Boundary. Note the presenter implements the output boundary which is what Savanna did. She also used setters and getters in her response model so, good. Also, presenter overwrites the method in output boundary so also approved!

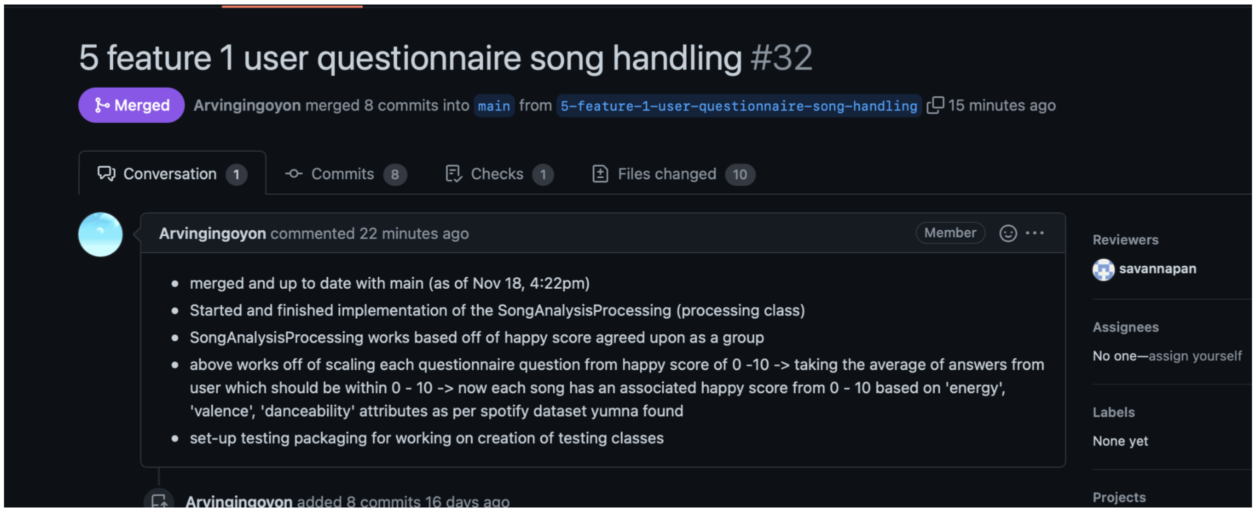
savannapan merged commit `156bf62` into `main` 36 minutes ago · 1 check passed · View details · Revert

Pull request successfully merged and closed
You're all set—the `12-feature-4-an…` branch can be safely deleted. · Delete branch

Reviewers: yumnarefai ✓
Assignees: No one—assign yourself
Labels: None yet
Projects: None yet
Milestone: No milestone
Development: Successfully merging this pull request may close these issues. None yet
Notifications: Customize · Unsubscribe · You're receiving notifications because you modified the open/close state.

- I chose this pull request since it contains the completed processing, output boundary, presenter and response model for my analyse playlist system. This would provide an overall structure of how my system would communicate through the different layers. My processor will communicate with the History Entity and does all the processing/calculations in this layer. So far I have set up multiple methods to do the calculations, but I can work on how I may be able to combine or reuse the methods. The processing then interact with the presenter and uses the output boundary to transfer the results back to the UI, which will be implemented at a later date. Throughout the coding process, I realized that I may not need a controller since I am not going to take any user input. I will be working on the details later this week and start writing tests to test the classes in my system.

Code Review:



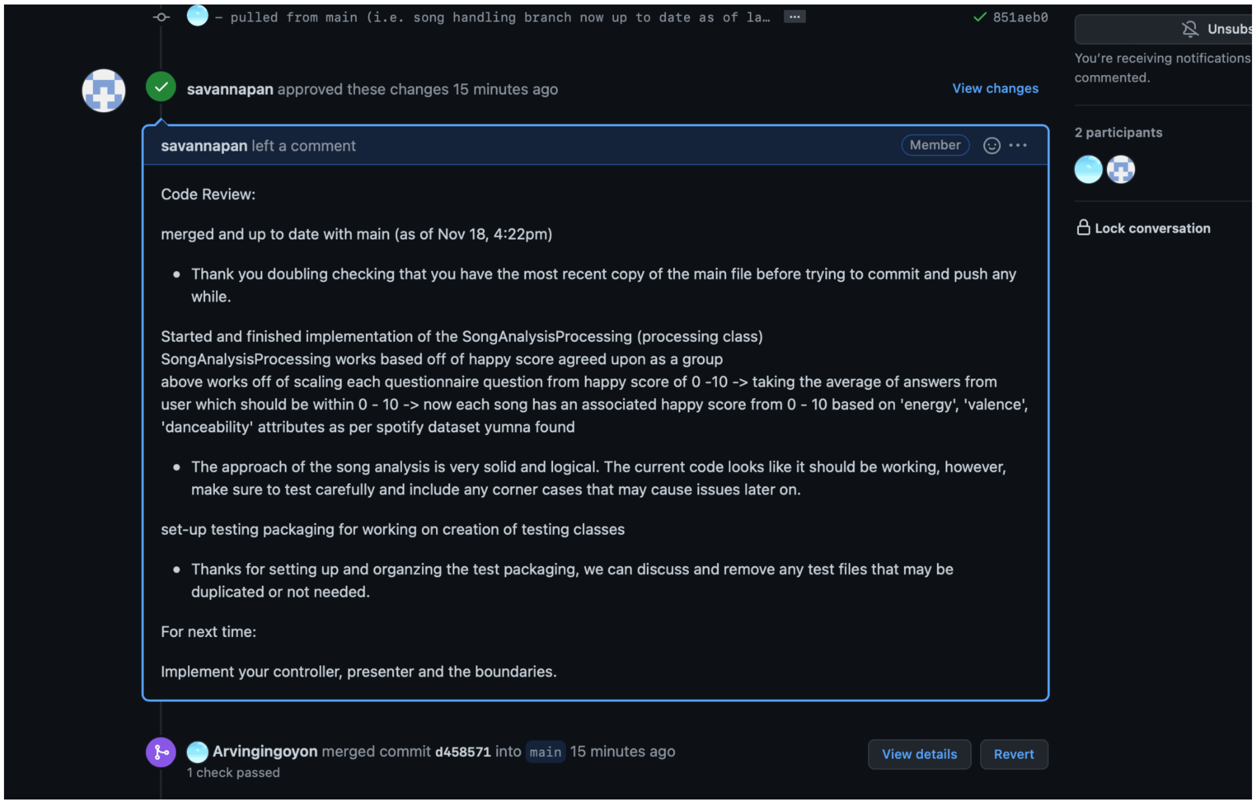
5 feature 1 user questionnaire song handling #32

Merged Arvingingoyon merged 8 commits into `main` from `5-feature-1-user-questionnaire-song-handling` 15 minutes ago

Arvingingoyon commented 22 minutes ago Member

- merged and up to date with main (as of Nov 18, 4:22pm)
- Started and finished implementation of the SongAnalysisProcessing (processing class)
- SongAnalysisProcessing works based off of happy score agreed upon as a group
- above works off of scaling each questionnaire question from happy score of 0 -10 -> taking the average of answers from user which should be within 0 - 10 -> now each song has an associated happy score from 0 - 10 based on 'energy', 'valence', 'danceability' attributes as per spotify dataset yumna found
- set-up testing packaging for working on creation of testing classes

Arvingingoyon added 8 commits 16 days ago



- pulled from main (i.e. song handling branch now up to date as of la… 851aeb0

savannapan approved these changes 15 minutes ago View changes

savannapan left a comment Member

Code Review:

merged and up to date with main (as of Nov 18, 4:22pm)

- Thank you doubling checking that you have the most recent copy of the main file before trying to commit and push any while.

Started and finished implementation of the SongAnalysisProcessing (processing class)
SongAnalysisProcessing works based off of happy score agreed upon as a group
above works off of scaling each questionnaire question from happy score of 0 -10 -> taking the average of answers from user which should be within 0 - 10 -> now each song has an associated happy score from 0 - 10 based on 'energy', 'valence', 'danceability' attributes as per spotify dataset yumna found

- The approach of the song analysis is very solid and logical. The current code looks like it should be working, however, make sure to test carefully and include any corner cases that may cause issues later on.

set-up testing packaging for working on creation of testing classes

- Thanks for setting up and organzing the test packaging, we can discuss and remove any test files that may be duplicated or not needed.

For next time:

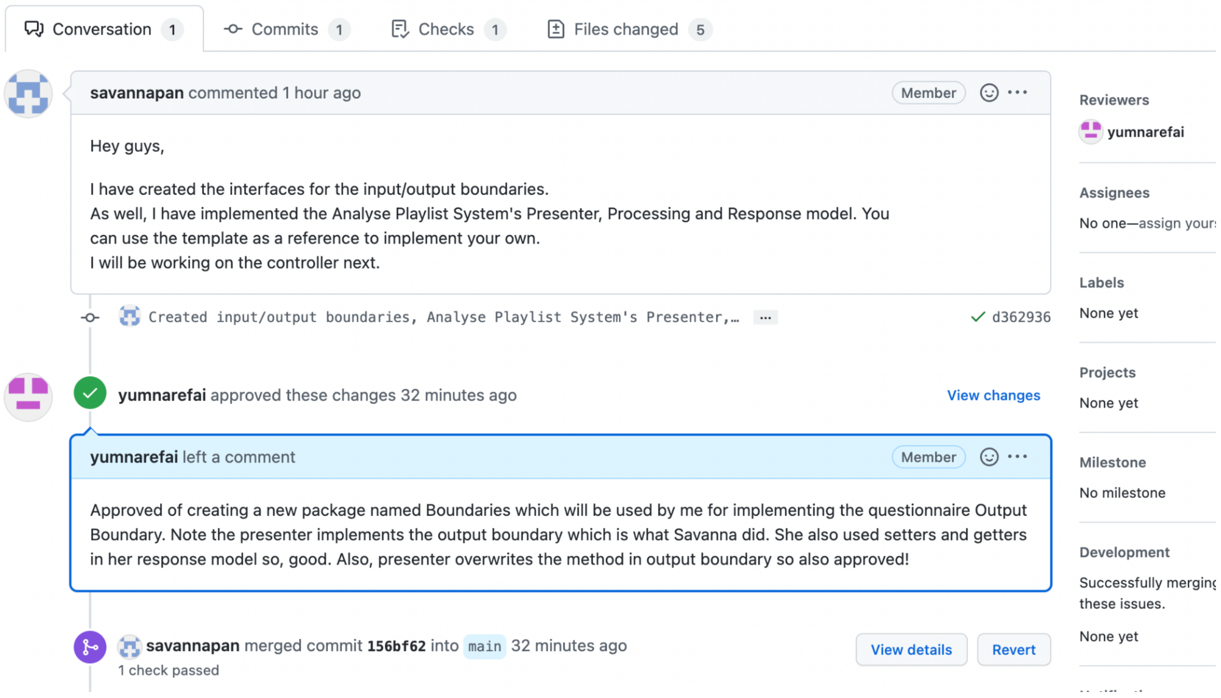
Implement your controller, presenter and the boundaries.

Arvingingoyon merged commit d458571 into `main` 15 minutes ago
1 check passed

- I chose this code review because I was able to provide the most feedback. And I had the opportunity to carefully read over Arvin’s code and understand how they are going to implement their questionnaire system.

Yumna's Pull Request

Code Contribution:
I chose this pull request because I did the most modifications and creations which are required for my questionnaire system.
I implemented the output boundary, questionnaire response model and presenter class for Questionnaire. I added an output boundary method for questionnaire which has the method generate. My Questionnaire Response Model uses setters and getters to set and get questions respectively. I implemented Presenter class to generate 5 random questions using the setter method and return these 5 questions. This can be done because Presenter implements output boundary which has the processing I made my presenter class abstract and this class implements output boundary so I overwrote only my method, generate from output boundary to update and return 5 random questions in my response model.



savannapan commented 1 hour ago · Member

Hey guys,

I have created the interfaces for the input/output boundaries.
As well, I have implemented the Analyse Playlist System's Presenter, Processing and Response model. You can use the template as a reference to implement your own.
I will be working on the controller next.

```
Created input/output boundaries, Analyse Playlist System's Presenter,… … ✓ d362936
```

yumnarefai approved these changes 32 minutes ago · View changes

yumnarefai left a comment · Member

Approved of creating a new package named Boundaries which will be used by me for implementing the questionnaire Output Boundary. Note the presenter implements the output boundary which is what Savanna did. She also used setters and getters in her response model so, good. Also, presenter overwrites the method in output boundary so also approved!

savannapan merged commit `156bf62` into `main` 32 minutes ago · View details · Revert
1 check passed

Code Review:

I chose this code review because I was able to provide concrete feedback on this. I looked over Savanna's code to double check if interfaces were correctly implemented and response models were made correctly.

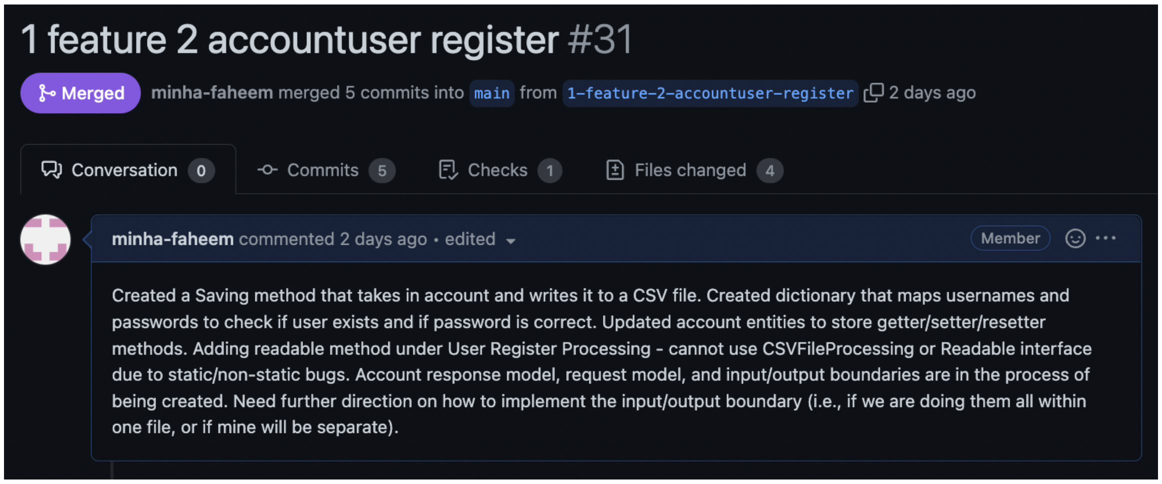
Minha's Pull Request

Minha's Code contribution:

I am responsible for the Account and User Register component of our program. This includes creating the Account entity, Account Checker Processing level, User Register Processing level, Account Response and Request Models, and input/output boundaries. The Account Checker Processing level consists of checking methods that assure the usernames and passwords being inputted (1) exist within the database, and (2) correct password has been inputted. The User Register processing level is responsible for assuring that any new User is written to a CSV File, and a method is being created to read that data as well. The response model for this has been created as well - that will return the login information to the controller (yet to be created). The request model also has yet to be created. Input/Output boundaries will be created in accordance with group decisions regarding implementation.

I have also allowed History/Favourites objects to be taken in as an attribute of Accounts, as each registered user will have an associated History/Favourites object to be accessed every time the User logins. New accounts will contain empty History and Favourites objects, that will be added to as the user proceeds to interact with the program. Tests will be created to ensure that the CSV database is correct in including every user's username, password, History, and Favourites, as well as testing checkers, getters, and setters.

Next steps: Input/Output boundaries, controllers, presenters, (potentially) a Gateway, and UI.


1 feature 2 accountuser register #31

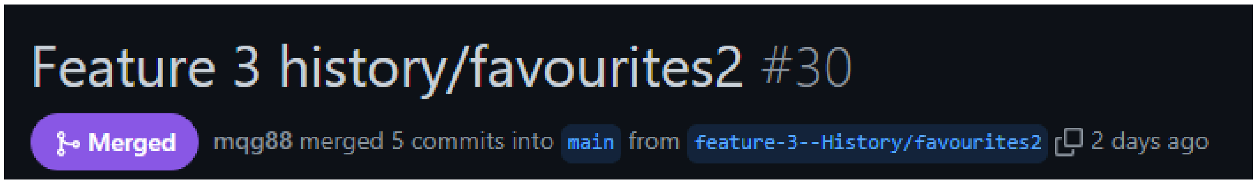
Merged minha-faheem merged 5 commits into `main` from `1-feature-2-accountuser-register` 2 days ago

Conversation 0 · Commits 5 · Checks 1 · Files changed 4

minha-faheem commented 2 days ago • edited · Member

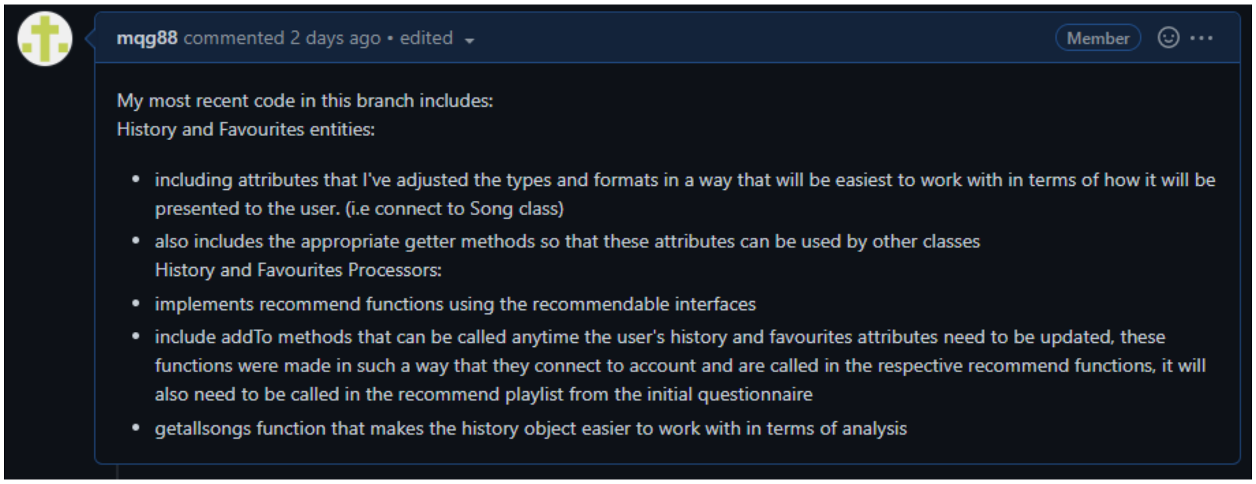
Created a Saving method that takes in account and writes it to a CSV file. Created dictionary that maps usernames and passwords to check if user exists and if password is correct. Updated account entities to store getter/setter/resetter methods. Adding readable method under User Register Processing - cannot use CSVFileProcessing or Readable interface due to static/non-static bugs. Account response model, request model, and input/output boundaries are in the process of being created. Need further direction on how to implement the input/output boundary (i.e., if we are doing them all within one file, or if mine will be separate).

Minha's Code Review:



Feature 3 history/favourites2 #30

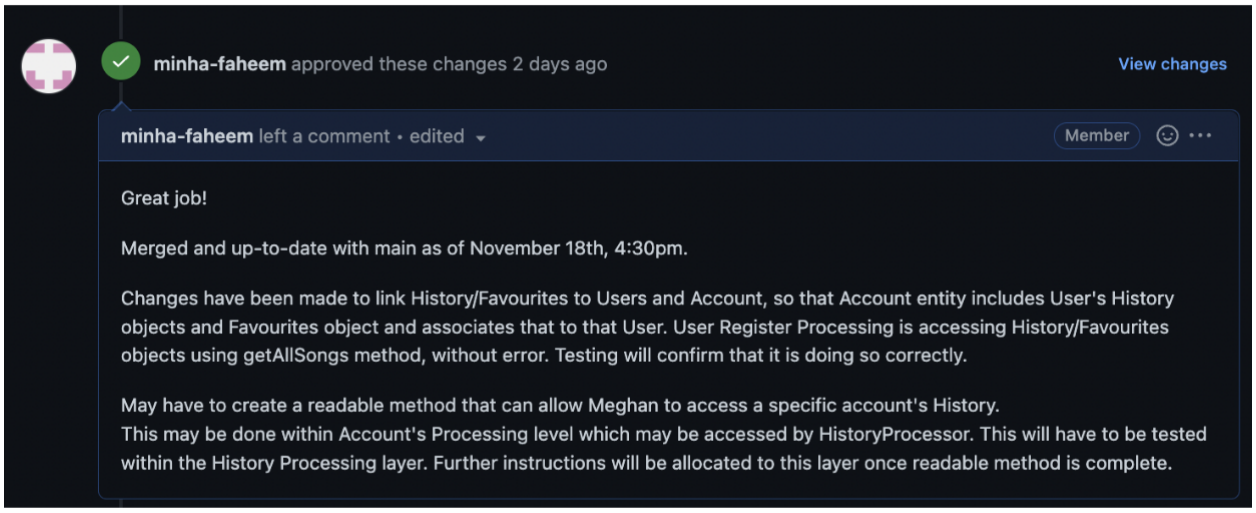
mqg88 merged 5 commits into `main` from `feature-3--History/favourites2` 2 days ago



mqg88 commented 2 days ago • edited

My most recent code in this branch includes:
History and Favourites entities:

- including attributes that I've adjusted the types and formats in a way that will be easiest to work with in terms of how it will be presented to the user. (i.e connect to Song class)
- also includes the appropriate getter methods so that these attributes can be used by other classes
  History and Favourites Processors:
- implements recommend functions using the recommendable interfaces
- include addTo methods that can be called anytime the user's history and favourites attributes need to be updated, these functions were made in such a way that they connect to account and are called in the respective recommend functions, it will also need to be called in the recommend playlist from the initial questionnaire
- getallsongs function that makes the history object easier to work with in terms of analysis



minha-faheem approved these changes 2 days ago

minha-faheem left a comment • edited

Great job!

Merged and up-to-date with main as of November 18th, 4:30pm.

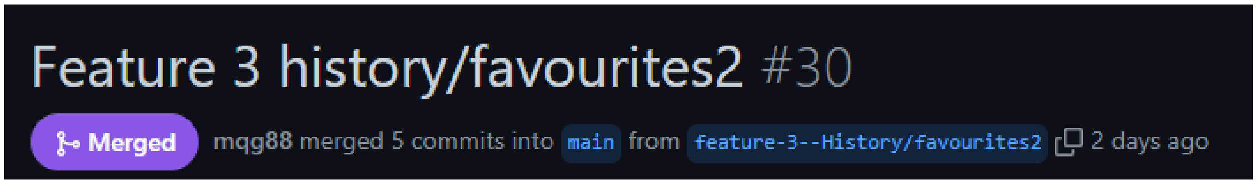
Changes have been made to link History/Favourites to Users and Account, so that Account entity includes User's History objects and Favourites object and associates that to that User. User Register Processing is accessing History/Favourites objects using getAllSongs method, without error. Testing will confirm that it is doing so correctly.

May have to create a readable method that can allow Meghan to access a specific account's History.
This may be done within Account's Processing level which may be accessed by HistoryProcessor. This will have to be tested within the History Processing layer. Further instructions will be allocated to this layer once readable method is complete.

Due to History/Favourites' interaction with User Registering and Account, Meghan and I had to work closely to ensure that each of our components were correctly interacting with one another without violating clean architecture. Methods from History/Favourites had to be used within User Registering, so I had to ensure that they were implemented accordingly. I also had to make note of potentially testing out if History can be read from the User database, for upcoming testing.
Next steps: Testing

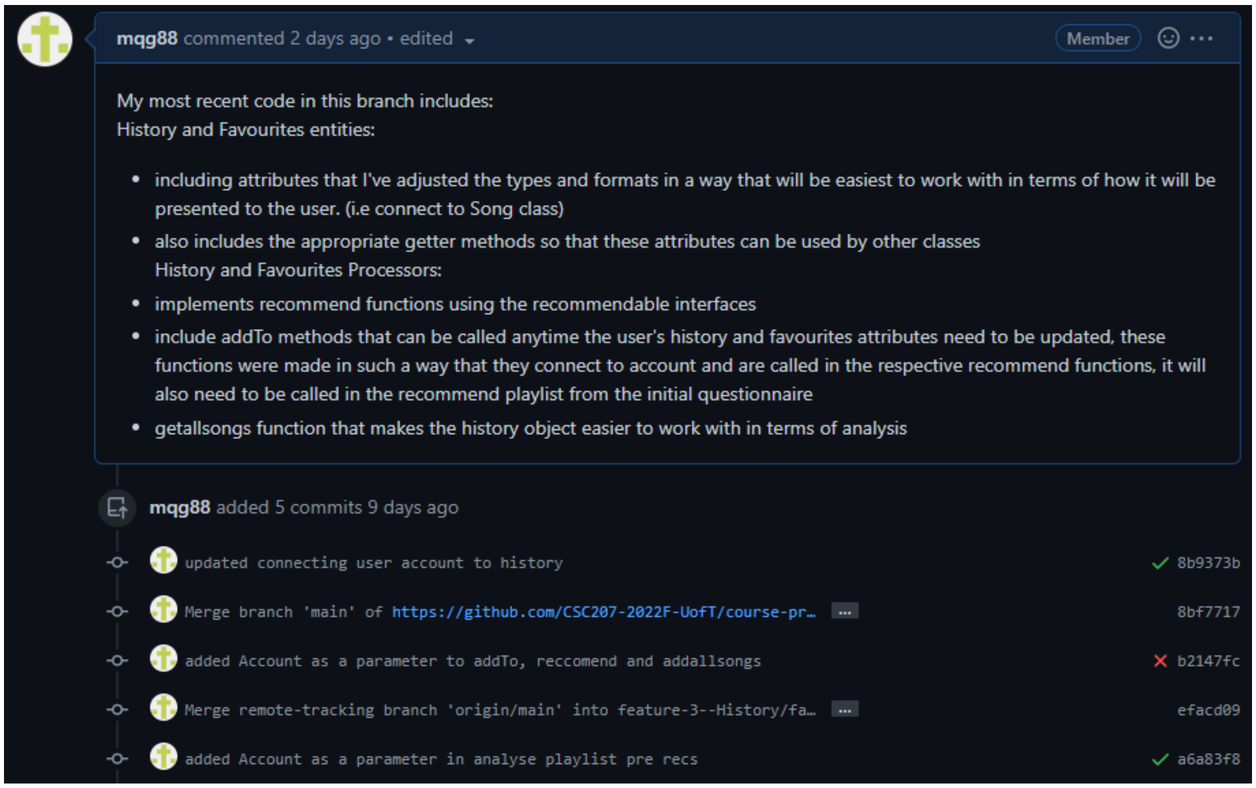
Meghan's Pull Request:

Code contribution:



Feature 3 history/favourites2 #30

Merged mqg88 merged 5 commits into `main` from `feature-3--History/favourites2` 2 days ago



mqg88 commented 2 days ago • edited

Member

My most recent code in this branch includes:
History and Favourites entities:

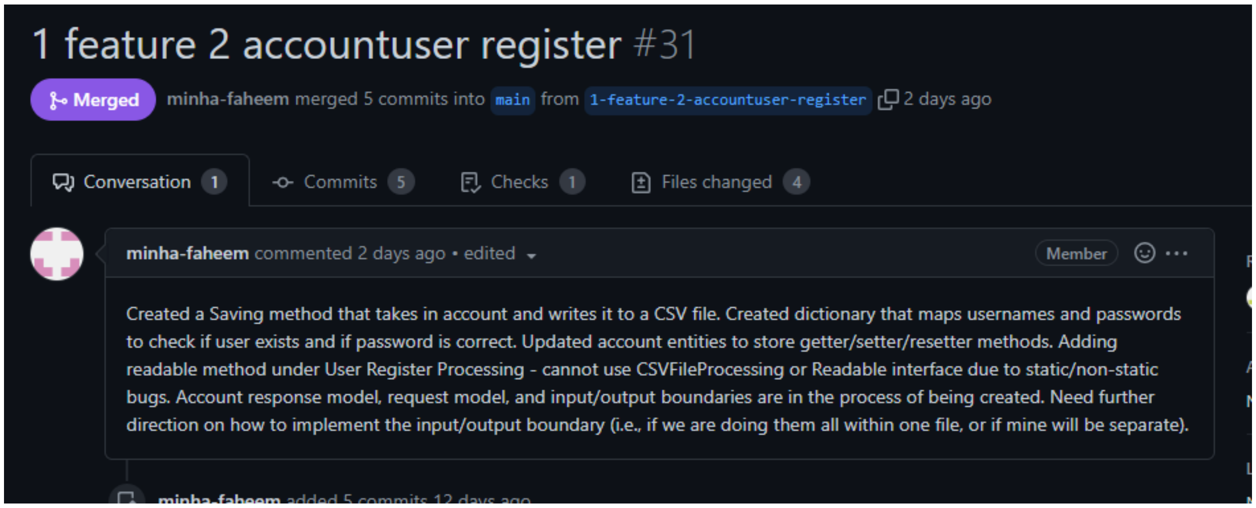
- including attributes that I've adjusted the types and formats in a way that will be easiest to work with in terms of how it will be presented to the user. (i.e connect to Song class)
- also includes the appropriate getter methods so that these attributes can be used by other classes
  History and Favourites Processors:
- implements recommend functions using the recommendable interfaces
- include addTo methods that can be called anytime the user's history and favourites attributes need to be updated, these functions were made in such a way that they connect to account and are called in the respective recommend functions, it will also need to be called in the recommend playlist from the initial questionnaire
- getallsongs function that makes the history object easier to work with in terms of analysis

mqg88 added 5 commits 9 days ago

```
updated connecting user account to history                                         ✓ 8b9373b
Merge branch 'main' of https://github.com/CSC207-2022F-UofT/course-pr… …              8bf7717
added Account as a parameter to addTo, reccomend and addallsongs                     ✗ b2147fc
Merge remote-tracking branch 'origin/main' into feature-3--History/fa… …              efacd09
added Account as a parameter in analyse playlist pre recs                           ✓ a6a83f8
```

I chose this pull request as it is my most recent contribution to the branch I've been working on concerning the History and Favourites features of our program. As written in the comment I've completed both the entities, as well as all of the functions thus far required in the processors. More specifically my recommend functions parse through a given history or favourites object and use the .random functions to return a playlist of 10 songs (accounting for no repeats) while my addTo functions will update the history and favourites objects belonging to the account anytime a new playlist is generated or when they add a song to favourites. While the bulk of the code was written early on, over the course of the projects I have been constantly adjusting the content and structure of the functions in order to work best with my teammates code, such as adjusting what inputs the functions take, rearranging which level of the code functions are on, adjusting interfaces to fit the needs of the functions implementing them. My next steps will be coding the controllers for both of these classes, as well as continuing to work with Minha on the readers and writers in account as they are closely associated with how the data in my entities will be saved and updated. Additionally I will work on writing tests for the functions in my processors.

Code Review



1 feature 2 accountuser register #31

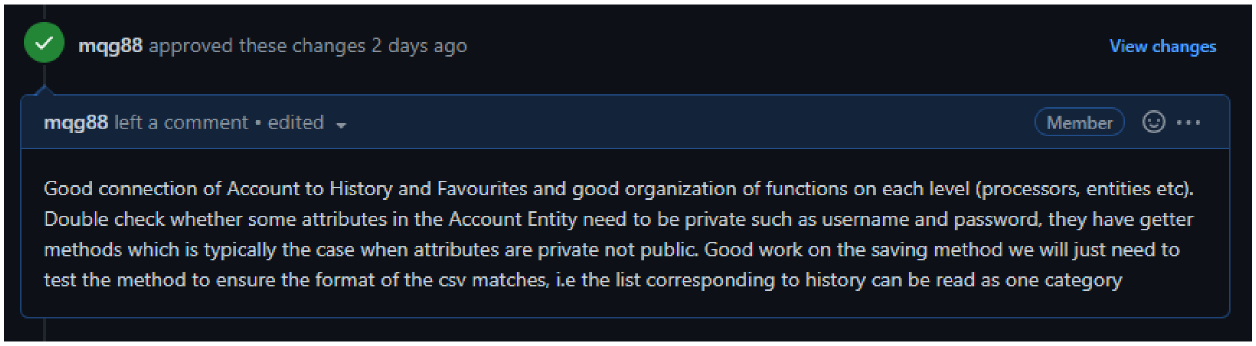
Merged minha-faheem merged 5 commits into `main` from `1-feature-2-accountuser-register` 2 days ago

Conversation 1 · Commits 5 · Checks 1 · Files changed 4

minha-faheem commented 2 days ago • edited · Member

Created a Saving method that takes in account and writes it to a CSV file. Created dictionary that maps usernames and passwords to check if user exists and if password is correct. Updated account entities to store getter/setter/resetter methods. Adding readable method under User Register Processing - cannot use CSVFileProcessing or Readable interface due to static/non-static bugs. Account response model, request model, and input/output boundaries are in the process of being created. Need further direction on how to implement the input/output boundary (i.e., if we are doing them all within one file, or if mine will be separate).

minha-faheem added 5 commits 12 days ago



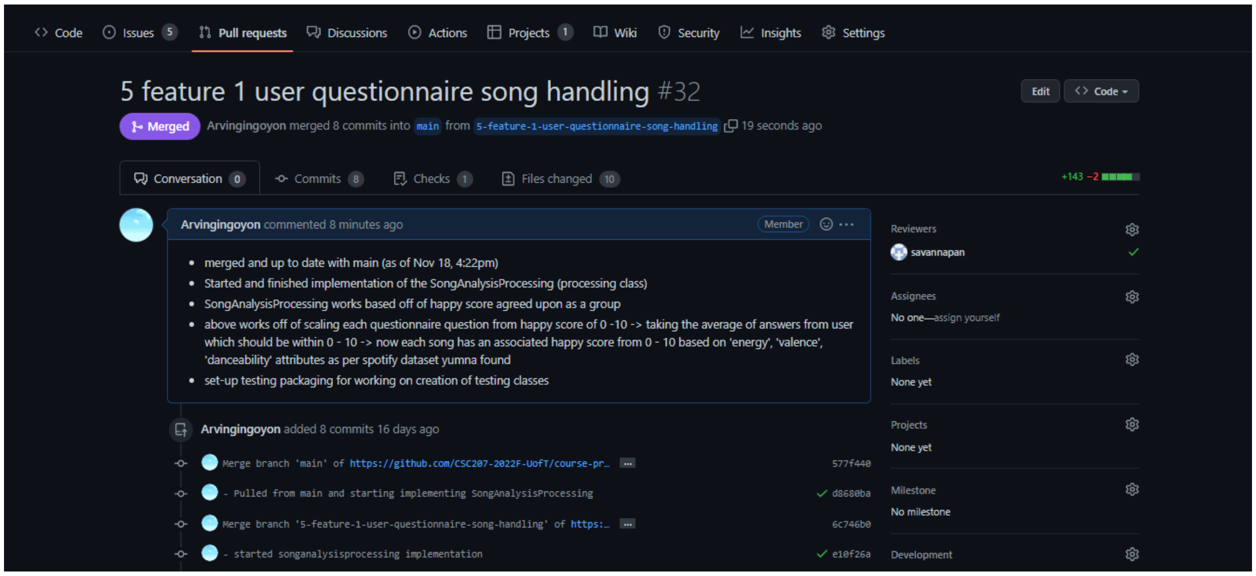
mqg88 approved these changes 2 days ago · View changes

mqg88 left a comment • edited · Member

Good connection of Account to History and Favourites and good organization of functions on each level (processors, entities etc). Double check whether some attributes in the Account Entity need to be private such as username and password, they have getter methods which is typically the case when attributes are private not public. Good work on the saving method we will just need to test the method to ensure the format of the csv matches, i.e the list corresponding to history can be read as one category

I provided a review for Minha's branch concerning account and user register. This worked well as much of the recent edits within both of our branches had to do with connecting our features so that History and Favourites were appropriately connected with a given account when we start the program. I reviewed the code and was able to consider the changes as well as any future things we may need to consider in relation to my own work (the commentary on whether the CSV will be read properly to create a History object).

Arvin’s Pull Request



5 feature 1 user questionnaire song handling #32

Merged Arvingingoyon merged 8 commits into `main` from `5-feature-1-user-questionnaire-song-handling` 19 seconds ago

Arvingingoyon commented 8 minutes ago

- merged and up to date with main (as of Nov 18, 4:22pm)
- Started and finished implementation of the SongAnalysisProcessing (processing class)
- SongAnalysisProcessing works based off of happy score agreed upon as a group
- above works off of scaling each questionnaire question from happy score of 0 -10 -> taking the average of answers from user which should be within 0 - 10 -> now each song has an associated happy score from 0 - 10 based on 'energy', 'valence', 'danceability' attributes as per spotify dataset yumna found
- set-up testing packaging for working on creation of testing classes

Arvingingoyon added 8 commits 16 days ago

```
Merge branch 'main' of https://github.com/CSC207-2022F-UofT/course-pr… …        577f440
- Pulled from main and starting implementing SongAnalysisProcessing        ✓ d8680ba
Merge branch '5-feature-1-user-questionnaire-song-handling' of https:… …  6c746b0
- started songanalysisprocessing implementation                           ✓ e10f26a
```

Reviewers
savannapan ✓

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

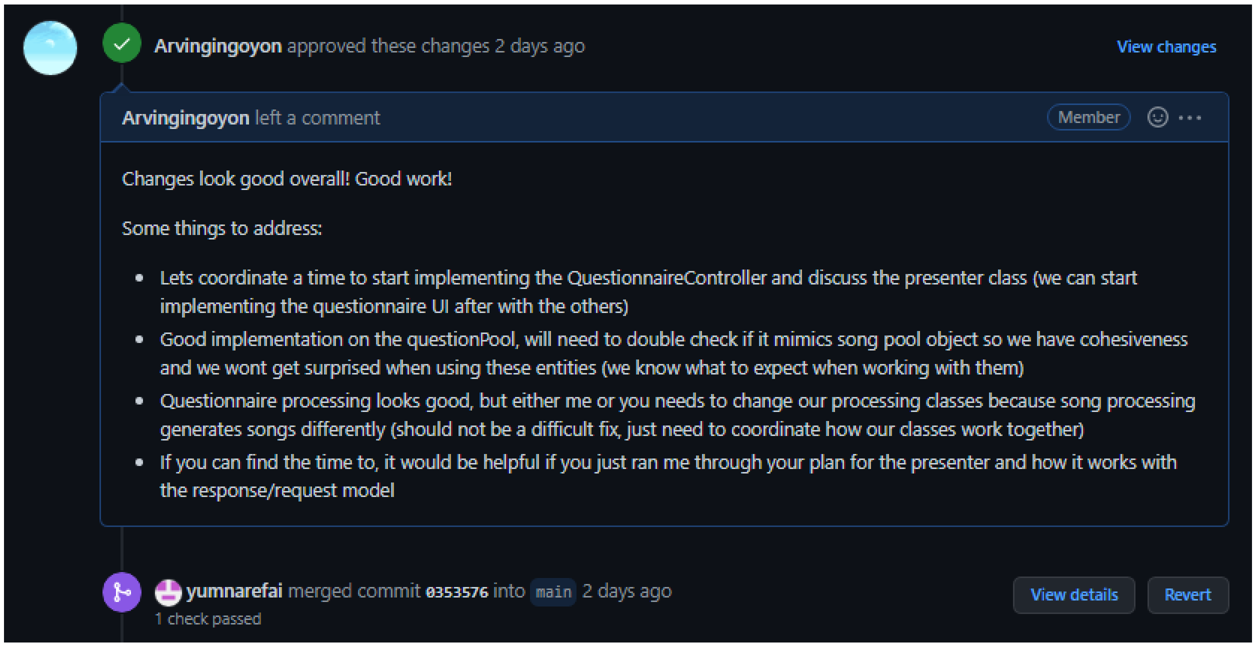
Milestone
No milestone

Development

I am responsible for implementing and working on song data handling in the questionnaire feature of our program. I chose this pull request because it not only represents my most recent contribution or change to the project, it also represents what my other pull requests we’re all working towards, which is a finished and functional implementation of the SongAnalysisProcessing class. With this, we will now be able to pull specific songs from our chosen song dataset, a crucial and necessary component for our program. Without this class, we would have no uniform way of passing songs and song data around to other classes. My previous significant contribution, implementing a data access interface to allow for the reading of data files, led to me being able to implement this class. Another reason why I choose this pull request is that it is able to clearly describe each significant change in the pull request. It is more up-to-par with how we will be expected to contribute later on in future group projects and work.

In short, SongAnalysisProcessing contains a method called getSong (and two other helper methods) which takes in a users ‘average happy score’, which is determined from user responses to questions in the questionnaire. It also takes in a generated song pool full of song objects generated from file processing classes using our readable file interface and our chosen data set of songs. As explained in my pull request, each questionnaire presented to the user works on a scale system with a scale of 0-10. Similarly, each song also has an associated score from 0-10. This class is able to return music based on the happy score of a user and each returned song should reflect that individual's mood, just as we planned for it to.

My chosen code review:


Arvingingoyon approved these changes 2 days ago

Arvingingoyon left a comment

Changes look good overall! Good work!

Some things to address:

- Lets coordinate a time to start implementing the QuestionnaireController and discuss the presenter class (we can start implementing the questionnaire UI after with the others)
- Good implementation on the questionPool, will need to double check if it mimics song pool object so we have cohesiveness and we wont get surprised when using these entities (we know what to expect when working with them)
- Questionnaire processing looks good, but either me or you needs to change our processing classes because song processing generates songs differently (should not be a difficult fix, just need to coordinate how our classes work together)
- If you can find the time to, it would be helpful if you just ran me through your plan for the presenter and how it works with the response/request model

yumnarefai merged commit `0353576` into `main` 2 days ago

1 check passed

In this code review, I highlighted some points of interest in my group member's code. Yumna and I are working together in implementing the questionnaire feature of our program and so I wanted to be sure we were both on the same page. Both me and Yumna are responsible for implementing the questionnaire controller and presenter classes, and I made sure to point out potential areas of concern between our implemented classes.