Sprint 8

Sprint Planning for Sprint 8 - 04/26/24:

- Interval from 04/26/24 to 05/10/24

Individual Capacities:

- Unit measurement: 1 point to 1 hour

Team member:	Capacity by hours
Julie	40
Jason	35
Kihambo	35
Diego	35
Parth	40
Khuong	50
An	50
Total	285

Work from Backlog:

Work Item	Effort Points (Hours)	Owner
Fixing ui interface	5	Julie
Update backend tests	20	Julie
Connect features that need to be connected	8	Julie
Analysis of DashBoard database and backend	30	Jason
Scale Display Complexity Updates	10	Kihambo
Collab Feature	25	Kihambo
UI frontend updates	30	Diego
Price Range Sorting	10	Parth

Item Creation/Listing/Offering	37	An
Seller Dashboard: Product Editing, Product Deletion, Inventory Stock List, and Financial Progress Report	50	Khuong
End-to-end testing	10	An
Analysis dashboard frontend	30	Parth
End-to-end testing	10	Khuong

Task Breakdown:

Julie's Work Item: (Internal Due Date): 05/05

Fixing UI: Task of fixing ui given to diego so i only have subtask for it.

1. Checking diego's branch to see the current ui fixes in css,js and html(he is separating every feature into its own folder. (1 hour)
2. Route pages accordingly (4 hours)

Update Backend Tests:

1. Update and create new DataAccessLayer test files (cs) to pass and cover all code. (3 hours)
2. Update Service layer test files (cs) to pass and cover all code. (2 hours)
3. Create new Service layer test files (cs) to pass and remaining code. (3 hours)
4. Update Securitylayer test files (cs) to pass and cover all code. (2 hours)
5. Create new Securitylayer test files (cs) to pass and remaining code. (3 hours)
6. Update and create new Logging layer test files (cs) to pass and cover all code. (2 hours)
7. Create Controller test files to pass and cover all code (5 hours)

Connecting features that need to be connected:

1. Reading and understanding kihambos collab feature html and js file so i can properly insert my connection later (2

2. Adding artist portfolio view to kambos collab feature when he is done. This is done by adding a button in his collab feature html for every user that comes up on the search to route to my display portfolio function in my feature js. He should have buttons ready to go. (4 hours)

Estimate = 31 hours

Kihambo's Work Item Collab Feature: (Internal Due Date): 5/03

1. Fix the Collab Feature DAL and Controller so that it stores the usernames that have been sent collabs into the database. (5 hours)
2. Fix the CollabFeature.js file so that it retrieves those usernames who have been sent collabs and displays them to the currently logged- in user when they click on "See Collabs." (4 hours)
3. Fix the Collab Feature.js file so that it doesn't display a "Error Loading View. Please Try Again" message in place of the name of the feature in the main menu when it successfully loads the view anyway. (4 hour)
4. Clean up the User Interface for Collab Feature so that it looks more presentable
    a. Add color as well as separate the buttons on the feature so that they occupy more of the page and aren't so close together (4 hours)
    b. Display the available usernames that the user can collaborate with in a mini- window fashion (4 hours)
5. Make the automated tests for every test case written in the BRD (4 hours)

Estimate = 25

Kihambo's Work Item Scale Display Complexity Updates: (Internal Due Date): 5/04

1. Create backend for Scale Display feature (10 hours total)
    a. Make a controller for scale display (5 hours)
    b. Make a DAL layer to access a data store for images of all the scale notes and their types (5 hours)
        i. Screenshots of the images from MuseScore

Estimate = 10

Diego's Work Item: Frontend UI Updates & BingoBoard fixes

1. Add new login page on launch

    a. Display register/login/recovery options
2. Add page to navigate between MusiCali and CraftVerify after login
    a. Let users navigate to either musicali or craftverify views
    b. Should also let users navigate to profile
3. Add nav bar for MusiCali home page to switch between features
    a. Displays different buttons to switch to different musicali feature views(bingo board, profile, display, etc.)
4. Bingo board fixes
    a. Update controllers, service, and DAL code to reflect Vong feedback
    b. Add pagination numbers to frontend

Estimation = 35 hrs

Jason's Work Item: (Internal Due Date): 4/13

Analysis Dashboard Database and Backend:

1. Backend for Key Performance Indicator (KPI) of the number of successful and failed login attempts per month over selected time span (4 hours total)
    a. Create the unit test for receiving the number of successful and failed login attempts per month over selected time span (1 hour).
    b. Implement the Database Access Layer to retrieve the number of successful and failed login attempts from the database (1 hour).
    c. Implement Service Layer to add logging when a user sees/receives the trend chart of the number of successful and failed login attempts from the database (1 hour).
    d. Implement Controller Layer to retrieve the number of months in the timespan from frontend and send back the number of successful and failed login attempts per month to the frontend to be created as a trend chart (1 hour)
2. Backend for Key Performance Indicator (KPI) of the number of success and failed registrations per month over selected time span (4 hours total)
    a. Create the unit test for receiving the number of success and failed registrations per month over selected time span (1 hour).
    b. Implement the Database Access Layer to retrieve the number of success and failed registrations per month over selected time span (1 hour).
    c. Implement Service Layer to add logging when a user sees/receives the trend chart of the number of success and failed registrations per month over selected time span from the database (1 hour).
    d. Implement Controller Layer to retrieve the number of months in the timespan from frontend and send back the number of success and failed registrations per month over selected time span to the frontend to be created as a trend chart (1 hour)
3. Backend for Key Performance Indicator (KPI) of the 3 longest page visits in seconds (5 hours)

    a. Create the unit test for receiving the 3 longest page visits in seconds (1 hour).
    b. Implement the Database Access Layer to retrieve the 3 longest page visits (on average) over selected time span (1 hour)
    c. Implement the logging of the length of time spent on a page visit to be saved to the database (1 hour)
    d. Implement the Service Layer to add logging when a user sees/receives the 3 longest page visits in seconds from the database (1 hour).
    e. Implement Controller Layer to retrieve the number of months in the timespan from frontend and send back the 3 longest page visits over selected time span to the frontend (1 hour)
4. Backend for Key Performance Indicator (KPI) of the 3 most used features (4 hours total)
    a. Create the unit test for receiving the 3 most used features (1 hour).
    b. Implement the Database Access Layer to retrieve the 3 most used features over selected time span (1 hour)
    c. Implement the Service Layer to add logging when a user sees/receives3 most used features from the database (1 hour).
    d. Implement Controller Layer to retrieve the number of months in the timespan from frontend and send back 3 most used features over selected time span to the frontend (1 hour)
5. Backend for Key Performance Indicator of the number of gigs created per month (displayed as a trend chart) (4 hours total)
    a. Create the unit test for receiving the number of gigs created per month over selected time span (1 hour).
    b. Implement the Database Access Layer to retrieve the number of gigs created from the database (1 hour).
    c. Implement Service Layer to add logging when a user sees/receives the trend chart of the number of gigs created from the database (1 hour).
    d. Implement Controller Layer to retrieve the number of months in the timespan from frontend and send back the number of gigs created per month to the frontend to be created as a trend chart (1 hour)
6. Backend for Key Performance Indicator of the number of collabs within a selected timespan (4 hours total)
    a. Create the unit test for receiving the number of collabs per month over selected time span (1 hour).
    b. Implement the Database Access Layer to retrieve the number of collabs from the database (1 hour).
    c. Implement Service Layer to add logging when a user sees/receives the trend chart of the number of collabs from the database (1 hour).
    d. Implement Controller Layer to retrieve the number of months in the timespan from frontend and send back the number of collabs per month to the frontend to be created as a trend chart (1 hour)

Estimation = 4 + 4 + 5 + 4 + 4 = 21 hours

An's Work Item: (Internal Due Date): 05/08

Item Creation, Listing, and Offering:

1. Item Creation:
    a. Front end: 7h40min
        i. Create CraftVerify button on home page to change view to CraftVerify view: 10mins
        ii. Create CratVerify.js and css to handle the view change function and set the format decoration of the view, button, color, padding: 30 mins
        iii. Create CraftVerify view with buttons for Price range sorting, Buyer History, Seller Dashboard: 20mins
        iv. Making js function to change the view of CraftVerify to Seller Dashboard view: 20 mins
        v. Create Seller Dashboard view with buttons for Item Creation, Item Listing, Item Editting/Deleting, Inventory Stock, Financial Progress Report: 20 mins
        vi. Making js function to change the view of Seller Dashboard to Item Creation: 10 mins
        vii. Create Item Creation view with all attributes input and images, videos select button: 20 mins
        viii. Making js function to validate the input data to match BRD: 30 mins
        ix. Making js function to Ajax fetch call 2 APIs, 1 for upload the pics and videos to backend sandbox/userhash folder, 1 for create an item and upload files to s3, and insert item to database: 2 hours
        x. Front end testing, unit testing, automated test will be in each tasks but I will combine them together here, total of: 3 hours.

    b. Backend 17h10min
        i. Create an s3 bucket aws: 10 mins
        ii. Set up user, user group, permission: 20 mins
        iii. Install aws s3 sdk on window, config account key, secret key: 40mins
        iv. Making s3 object model, which have file name, presigned URL 10mins
        v. Making UploadS3Controller, configuration: 20 mins
        vi. Making API endpoint to upload all files from a front end to a backend sandbox/userhash folder: 2 hours

vii. Making API endpoint to upload all file from sandbox/userhash to aws s3 /sku folder: 2 hours.
viii. Making API endpoint to get all file from aws s3 /sku folder: 30mins
ix. Making API endpoint to preview, get presignedURL of a file from aws s3 /sku folder: 20 mins
x. Making item model: 20mins
xi. Making ItemCreationDAO, which insert item to database table, check the sku duplicate: 1 hour
xii. Making ItemCreationService, which will validate all item attributes to adhere to BRD, generate sku, check pic and videos file, create an item: 3 hours
xiii. Making ItemCreationController, config: 20mins
xiv. Making API endpoint to create item from json form: 1 hour
xv. Backend testing, unit testing, automated test will be in each tasks but I will combine them together here, total of: 5 hours.

c. LLD Fixing, adding more fail cases of left over files in sandbox not uploaded to s3 due to error happen in itemCreation process: 30mins

2. Item Listing:
   a. Front end 6h30mins
      i. Create Item Listing view : 20 mins
      ii. Making js to change view from Seller Dashboard to item Listing view: 10 mins
      iii. Making Pagination view to show the seller their all items in list, with buttons to choose listing the item or off list item: 2 hours
      iv. Making js to Ajax call API to get all seller item with their sku and list attribute: 1 hour
      v. Making js to Ajax fetch call API to modify the change of the items: 1hour
      vi. Front end testing, unit testing, automated test will be in each tasks but I will combine them together here, total of: 2 hours.

   b. Back end 8h
      i. Create ItemListingDao to get the list of item based on username, and insert/modify the change of listing status of items in the database: 2 hours
      ii. Create ItemListingController, config it: 30 mins
      iii. Create ItemListing api endpoint to get all the item from a username: 30mins
      iv. Create itemListing api endpoint to modify the listing status of an item: 1 hour

v. Create ItemListingService to handle pagination: 1 hour
vi. Backend testing, unit testing, automated test will be in each tasks but I will combine them together here, total of: 3 hours.

c. Fixing LLD to reflect the change in pagination failure: 1hour

3. Item Offering:
    a. Front end 6h30mins
        i. Create Item Offering view : 20 mins
        ii. Making js to change view from Seller Dashboard to Item Offeringview: 10 mins
        iii. Making Pagination view to show the all items with offer attribute is true, and a buttons to choose to offer a price for an item in Craft Receipt table: 2 hours
        iv. Making js to Ajax call API to get all item with their attribute: 1 hour
        v. Making js to Ajax fetch call API to get the seller contact to show buyer, and send both email to seller and buyer about the offering: 1hour
        vi. Front end testing, unit testing, automated test will be in each tasks but I will combine them together here, total of: 2 hours.

    b. Back end 7h
        i. Create ItemOfferingDao to get the list of items, and insert/modify the change the offer price status of items in the database: 2 hours
        ii. Create ItemOfferingController, config it: 30 mins
        iii. Create ItemOffering api endpoint to get all the item in databse with offer attribute is true: 30mins
        iv. Create itemListing api endpoint to modify the offer price status of an item in Craft Receipt table: 1 hour
        v. Create ItemListingService to handle pagination, send email to both buyer ans seller about the price offering: 1 hour
        vi. Backend testing, unit testing, automated test will be in each tasks but I will combine them together here, total of: 3 hours.
    c. Fixing LLD on success case, and failcase of pagination: 1 hour

Estimate =

1+7+17+0.5+6.5+8+6.5+1+7+1 = 55.5 hour

Parth's Work Item: (Internal Due Date): 04/15

Price Range Sorting:

Frontend Development (4 hours):

1. HTML Layout and Integration (2 hours)
    - File: PriceRangeSorting.html
    - Section: Create a form structure including two input fields for topPrice and bottomPrice, a submit button, and a results display section.
    - Why and How: This form allows the user to input their desired price range. The structure will be straightforward, facilitating ease of use and ensuring accessibility standards are met. The form will be integrated into the existing page layout to maintain a consistent user experience.
2. CSS Styling (1 hour)
    - File: PriceRangeSorting.css
    - Section: Style the new form, focusing on the input fields, button, and result area. Implement responsive design for various screen sizes.
    - Why and How: CSS will enhance the visual appeal and ensure that the form is functional across devices. This includes setting appropriate margins, padding, colors, and fonts that align with the existing design guidelines of the application.
3. JavaScript Functionality (2 hours)
    - File: PriceRangeSorting.js
    - Section: Script setup to handle AJAX requests for form submission, input validation, and DOM manipulation for displaying results.
    - Why and How: JavaScript will manage the form's dynamic behavior, including client-side validation to prevent erroneous data submission. The AJAX setup will communicate with the backend without needing a full page reload, improving the user experience by providing faster response times.

Backend Development (5 hours):

1. Controller Setup (1 hour)
    - File: PriceRangeSortingController.cs

- Section: Implement GetPagedFilteredItems method to handle HTTP requests, extract parameters, and call the service layer.
- Why and How: The controller will parse incoming requests, ensuring that all required parameters are valid, and then pass these parameters to the service layer. It acts as a gatekeeper for the data flow into the application.

2. Service Layer Implementation (2 hours)
    - File: PriceRangeSortingService.cs
    - Section: Code the logic to request data from the DAO based on user input and apply any necessary transformations or additional business rules.
    - Why and How: The service layer will implement the core logic for processing data, ensuring that the business rules, like filtering and pagination, are correctly applied. It serves as a bridge between the controller and the DAO, encapsulating the business logic of the application.
3. Data Access Layer (DAL) Coding (2 hours)
    - File: PriceRangeSortingDAO.cs
    - Section: Implement FetchPagedItems to execute the SQL query and return the data.
    - Why and How: The DAL will interact directly with the database to retrieve data based on the given parameters. Efficient SQL queries will be written to minimize response time and resource consumption, focusing on fetching only the necessary data.

Deployment (1 hour)

- Details: Roll out the implemented feature to the staging server, configure any required environment settings, and perform initial integration checks.
- Why and How: This step is essential for ensuring that the new feature integrates seamlessly with the existing platform and works as expected in a live-like environment.

Estimate = 10 hours

Analysis Dashboard Frontend:

HTML Layout (10 hours)

- File: dashboard.html, widgets.html, charts.html

- Section: Develop HTML for the main dashboard layout in dashboard.html (5 hours), interactive widgets in widgets.html (3 hours), and data visualization elements in charts.html (2 hours).
- Why and How: These HTML files structure different components of the dashboard, ensuring modular design and ease of maintenance. Each file is tailored to optimize user experience and facilitate efficient data interaction, designed to provide clear, intuitive navigation and presentation of information.

CSS Styling (10 hours)

- File: dashboard.css, widgets.css, charts.css
- Section: Apply cohesive layout styles in dashboard.css (4 hours), visually engage styles for widgets in widgets.css (3 hours), and enhance chart readability and appeal in charts.css (3 hours).
- Why and How: The CSS files will create a unified and responsive design that not only matches the visual theme of the entire application but also ensures that the user interface is accessible and easy to navigate. The focus is on improving interaction through visual cues and consistent design language, which supports the functionality and aesthetic appeal of the dashboard.

JavaScript Development (10 hours)

- File: dashboard.js, widgets.js, charts.js
- Section: Implement user interaction handling in dashboard.js (4 hours), manage widget functionality in widgets.js (3 hours), and enable dynamic data visualization in charts.js (3 hours).
- Why and How: JavaScript files will be used to enhance the dashboard's interactivity, allowing users to dynamically manipulate data visualizations, filter results, and navigate different dashboard sections. This setup will prioritize performance to manage large datasets efficiently and update the user interface in real time, ensuring a seamless and responsive user experience.

Estimate = 30 hours

Khuong’s Work Item: (Internal Due Date): 05/08

Product Editing, Product Deletion, Inventory Stock List, and Financial Progress Report :

Product Editing: 9h

Front end 5h

i. Create Item Editing view: 20 mins
ii. Making js to change view from Seller Dashboard to Item Editing view: 10 mins

iii. Making Pagination view to show the all items of a seller, with buttons to modify item attributes: 1 hour
iv. Making js to Ajax call API to get all items with their attribute: 1 hour
v. Create item form view to input data: 20mins
vi. Making js to Ajax fetch call API to upload the new pic and video to sandbox/userhash: 20 mins
vii. Making js to Ajax fetch call API to upload files to s3 and update the item to database: 20 mins
viii. Frontend testing, unit testing, and automated testing will be in each task but I will combine them here, a total of: 1.5 hours.

d. Back end 4h
   i. Create ItemEditingDao to get the list of items, and insert/modify the change the items in the database: 1 hour
   ii. Create ItemEditingController, config it: 30 mins
   iii. Create ItemEditing API endpoint to get all the items from a username: 30mins
   iv. Create ItemEditing API endpoint to modify the item in the database table: 1 hour
   v. Create ItemEditingSevice to handle pagination: 1 hour
   vi. Backend testing, unit testing, and automated testing will be in each task but I will combine them here, a total of 2 hours.

Estimate = 9 Hours

Product Deletion:

Front end 5h

vii. Create Item Deletion view: 20 mins
viii. Making js to change view from Seller Dashboard to Item Deletion view: 10 mins
ix. Create item deletion view to input SKU number: 20mins
x. Making js to AJAX call API to get all items with their attribute: 1 hour
xi. Making js to Ajax fetch call API to delete pic and video from sandbox/userhash: 20 mins
xii. Making js to Ajax fetch call API to delete from to s3 and delete the item from the database: 20 mins
xiii. Front end testing, unit testing, and automated testing will be in each task but I will combine them here, a total of: 1.5 hours.

e. Back end 4h

i. Create ItemDeletionDAO to remove the items in the database: 1 hour
ii. Create ItemDeletionController, config it: 30 mins
iii. Create ItemDeletion API endpoint to delete items using SKU: 30mins
iv. Create ItemEditing API endpoint to delete the item in the database table: 1 hour
v. Create ItemDeletionService to delete the item using ItemDeletionDAO: 1 hour
vi. Backend testing, unit testing, and automated testing will be in each task but I will combine them here, a total of 2 hours.

Estimate = 9 Hours

Inventory Stock List:

Backend Development (9 hours):

1. Controller Setup (3 hours)
    - File: InventoryStockController.cs
    - Section: Implement the GetInventoryStock method.
    - Why and How: The InventoryStockController is used to listen to the fetch request coming from the Frontend, call InventoryStockService to get all of the Items belonging to the authenticated user with the InventoryStock Model return by the RequestInventoryStockList method, and pass the list of items back as a HashSet of InventoryStockModel type.
2. Service Layer Implementation (3 hours)
    - File: InventoryStockService.cs
    - Section: Implement the RequestInventoryStockList method.
    - Why and How: The InventoryStockService calls the RecoverUserDAO to retrieve the userhash with the username of the authenticated user and pass it to InventoryStockDAO through the parameter and get the HashSet of InventoryStockModel type from the GetStockList method. I will also implement automated tests for InventoryStockService.
3. Data Access Layer (DAL) Coding (3 hours)
    - File: PriceRangeSortingDAO.cs
    - Section: Implement the GetStockList method.
    - Why and How: The InventoryStockDAO is the data access object that allows the InventoryStockService to search and retrieve a HashSet of InventoryStockModel type from the database using the query specifically retrieve every item that the user created with this information: Name, SKU

number, Price, Stock, and Creation Date. The InventoryStockDAO then sends the HashSet of InventoryStockModel type to the service calling the GetStockList reference. I will also implement automated tests for InventoryStockDAO.

Frontend Development (5 hours):

1. HTML Layout and Integration (2 hours)
    - File: InventoryStock.html
    - Section: Create a form structure including a results display section when the user chooses to see their inventory.
    - Why and How: This form allows the user to see a list of the items in their inventory including Item Name, Item SKU, Item Stock, Item Price, and the date the authenticated user registers that item to our database. The time includes debugging and creating tests.
2. CSS Styling (1 hour)
    - File: InventoryStock.css
    - Section: CSS is for stylizing the result list for user experience and friendly design.
    - Why and How: CSS allows me to include setting appropriate margins, padding, colors, and fonts that align with the existing design guidelines of the application. The time includes debugging and creating tests.
3. JavaScript Functionality (2 hours)
    - File: InventoryStock.js
    - Section: Script setup to handle AJAX requests for form submission, input validation, and DOM manipulation for displaying results.
    - Why and How: The AJAX setup and event listeners allow authenticated users to communicate with the backend showing the authenticated inventory list using the results being passed in from the backend, improving the user experience by providing faster response times. The time includes debugging and creating tests.

Estimate = 14 Hours

Financial Progress Report:

Backend Development (10 hours):

1. Controller Setup (3 hours)
    - File: FinancialProgressReportController.cs
    - Section: Implement the GetInventoryStock method.

- Why and How: The FinancialProgressReportController is used to listen to the fetch request coming from the Frontend, call FinancialProgressReportService to get all the financial data belonging to the authenticated user with the FinancialInfoModel return by the GetReport method, and pass the list of economic data back as a HashSet of FinancialInfoModel type.

2. Service Layer Implementation (3 hours)
    - File: FinancialProgressReportService.cs
    - Section: Implement the GetReport method.
    - Why and How: The FinancialProgressReportService first identifies the frequency that the authenticated user selects(i.e. Yearly, Quarterly, or Monthly), then RecoverUserDAO to retrieve the userhash associated with the username of the authenticated user and reference the appropriate FinancialProgressReportDAO method and pass the userhash through the parameter and get the HashSet of FinancialInfoModel type from the FetchYearlyReport or FetchQuarterlyReport or FetchMonthlyReport method. I will also implement automated tests for FinancialProgressReportService.
3. Data Access Layer (DAL) Coding (4 hours)
    - File: FinancialProgressReportDAO.cs
    - Section: Implement the FetchYearlyReport, FetchQuarterlyReport, and FetchMonthlyReport methods.
    - Why and How: The FinancialProgressReportDAO is the data access object that allows the FinancialProgressReportService to search and retrieve a HashSet of FinancialInfoModel type from the database using the query specifically retrieve the economic data using different methods that represent different frequencies the economic data requires. The FinancialProgressReportDAO then sends the HashSet of FinancialInfoModel type to the service calling either the FetchYearlyReport or FetchQuarterlyReport or FetchMonthlyReport method reference. I will also implement automated tests for FinancialProgressReportDAO.

Frontend Development (12 hours):

1. HTML Layout and Integration (3 hours)
    - File: FinancialProgressReport.html
    - Section: Create a form structure including 3 buttons: Yearly, Quarterly, and Monthly, and a results display section.
    - Why and How: This form allows the authenticated user to view their financial progress over time through a line chart. The form includes 3

buttons used for drawing the line chart using data collected from different frequencies. The time includes debugging and creating tests.

2. CSS Styling (3 hour)
    - File: FinancialProgressReport.css
    - Section: Style the new form, focusing on buttons, and the Financial Progress Report line charts.
    - Why and How: CSS will allow me to stylize the form by adding colors, padding, and line chart characteristics to help the chart look easier to understand. The time includes debugging and creating tests.
3. JavaScript Functionality (3 hours)
    - File: FinancialProgressReport.js
    - Section: Script setup to handle AJAX requests for form submission, result validation, and DOM manipulation for displaying Financial Progress Report Line Charts.
    - Why and How: The AJAX setup and event listeners allow authenticated users to communicate with the backend and draw the line charts using the results being passed in from the backend, improving the user experience by providing faster response times. The time includes debugging and creating tests.

Estimate = 22 Hours

Total Estimate = 307 hours from backlog

Individual Breakdown

Julie Reyes

Work Item	Task	Estimation in pts	Team Confirmation
Fixing ui	Checking diego's branch to see the current ui fixes in css,js and html(he is separating every	1 hour	AN, DG,KN,PN, JR, JJ, BM, SK

	feature into its own folder.)		
	Write js file to Route pages accordingly(make all features html page routes map correctly)	4 hours	AN, DG,KN,PN, JR, JJ, BM, SK
Update Backend test	Update and create new DataAccessLayer test files (cs) to pass and cover all code.	3 hours	AN, DG,KN,PN, JR, JJ, BM, SK
	Update Service layer test files (cs) to pass and cover all code.	2 hours	AN, DG,KN,PN, JR, JJ, BM, SK
	Create new Service layer test files (cs) to pass and remaining code.	3 hours	
	Update Securitylayer test files (cs) to pass and cover all code.	2 hours	AN, DG,KN,PN, JR, JJ, BM, SK
	Create new Securitylayer test files (cs) to pass and remaining code.	3 hours	AN, DG,KN,PN, JR, JJ, BM, SK

	Update and create new Logging layer test files (cs) to pass and cover all code.	2 hours	AN, DG,KN,PN, JR, JJ, BM, SK
	Create Controller test files to pass and cover all code	5 hours	AN, DG,KN,PN, JR, JJ, BM, SK
Connect all features to work together	Reading and understanding kihambos collab feature html and js file so i can properly insert my connection later	2 hours	AN, DG,KN,PN, JR, JJ, BM, SK
	Adding artist portfolio view to kambos collab feature when he is done. This is done by adding a button in his collab feature html for every user that comes up on the search to route to my display portfolio function in my feature js. He should have buttons ready to go.	4 hours	AN, DG,KN,PN, JR, JJ, BM, SK
	Total	31 hours	

1. Fixing ui - 5 hours
2. Backend tests - 20 hours
3. Connecting features - 6 hours

Total New Estimate = 29 hours

40hrs- 31hrs = 9 extra hrs. A lot of hours are left over but again as team lead I usually work over and since this is the last sprint I want to have an abundance of extra time to finish anything that may arise. There will likely be a last push the last few days til the due date so I expect to finish any left over work from the team then and reach 40 hours.

Jason Jitsiripol

Work Item	Task	Estimation in pts	Team Confirmation
Analysis Dashboard Database and Backend	Create the unit test for receiving the number of successful and failed login attempts per month over selected time span.	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement the Database Access Layer to retrieve the number of successful and failed login attempts from the database.	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement Service Layer to add logging when a user sees/receives the trend chart of the number of successful and failed login	1	AN, DG,KN,PN, JR, JJ, BM, SK

	attempts from the database.		
	Implement Controller Layer to retrieve the number of months in the timespan from frontend and send back the number of successful and failed login attempts per month to the frontend to be created as a trend chart	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Create the unit test for receiving the number of success and failed registrations per month over selected time span	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement the Database Access Layer to retrieve the number of success and failed registrations per month over selected time span	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement Service Layer to add logging when a user sees/receives the trend chart of the number of success and failed registrations per month over selected	1	AN, DG,KN,PN, JR, JJ, BM, SK

	time span from the database		
	Implement Controller Layer to retrieve the number of months in the timespan from frontend and send back the number of success and failed registrations per month over selected time span to the frontend to be created as a trend chart	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Create the unit test for receiving the 3 longest page visits in seconds.		AN, DG,KN,PN, JR, JJ, BM, SK
	Implement the Database Access Layer to retrieve the 3 longest page visits (on average) over selected time span	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement the logging of the length of time spent on a page visit to be saved to the database	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement the Service Layer to add logging when a user sees/receives the 3 longest page visits in	1	AN, DG,KN,PN, JR, JJ, BM, SK

	seconds from the database		
	Implement Controller Layer to retrieve the number of months in the timespan from frontend and send back the 3 longest page visits over selected time span to the frontend	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Create the unit test for receiving the 3 most used features	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement the Database Access Layer to retrieve the 3 most used features over selected time span	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement the Service Layer to add logging when a user sees/receives3 most used features from the database	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement Controller Layer to retrieve the number of months in the timespan from frontend and send back 3 most used features over selected time span to the frontend	1	AN, DG,KN,PN, JR, JJ, BM, SK

	Create the unit test for receiving the number of gigs created per month over selected time span	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement the Database Access Layer to retrieve the number of gigs created from the database	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement Service Layer to add logging when a user sees/receives the trend chart of the number of gigs created from the database	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement Controller Layer to retrieve the number of months in the timespan from frontend and send back the number of gigs created per month to the frontend to be created as a trend chart	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Create the unit test for receiving the number of collabs per month over selected time span.	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement the Database Access	1	AN, DG,KN,PN, JR, JJ, BM, SK

	Layer to retrieve the number of collabs from the database		
	Implement Service Layer to add logging when a user sees/receives the trend chart of the number of collabs from the database	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Implement Controller Layer to retrieve the number of months in the timespan from frontend and send back the number of collabs per month to the frontend to be created as a trend chart	1	AN, DG,KN,PN, JR, JJ, BM, SK
	Total	21	

1. Usage Analysis Board Frontend and Controllers = 21

Total New Estimate = 21

35 hrs - 21 hrs = 14 extra hrs. This is within the individual member's sprint capacity. The extra hours can be used to account for additional testing and bug fixing from unexpected errors or scope creep and/or assistance in establishing the frontend for the Analysis Dashboard

Kihambo

Work Item	Task	Estimation in pts	Team Confirmation
Collab Feature	1. Fix the Collab	5	AN, DG,KN,PN, JR, JJ, BM, SK

	Feature DAL and Controller so that it stores the usernames that have been sent collabs into the database.		
	2. Fix the CollabFeature.js file so that it retrieves those usernames who have been sent collabs and displays them to the currently logged- in user when they click on “See Collabs.”	4	AN, DG,KN,PN, JR, JJ, BM, SK
	3. Fix the Collab Feature.js file so that it doesn’t display a “Error	4	AN, DG,KN,PN, JR, JJ, BM, SK

	Loading View. Please Try Again" message in place of the name of the feature in the main menu when it successfully loads the view anyway.		
	4. Clean up the User Interface for Collab Feature so that it looks more presentable	8	AN, DG,KN,PN, JR, JJ, BM, SK
	5. Make the automated tests for every test case written in the BRD	4	AN, DG,KN,PN, JR, JJ, BM, SK
Scale Display Complexity Updates	1. Create backend for Scale	10	AN, DG,KN,PN, JR, JJ, BM, SK

	Display feature		
	Total	35	

1. Collab Feature
2. Scale Display Complexity Testing

Total New Estimate = 35 hrs

35 hrs - 35 hrs = 0 extra hrs. This is the estimated time for me to complete this work item.

Diego

Work Item	Task	Estimation in pts	Team Confirmation
Frontend UI updates	Add new login page	6	AN, DG,KN,PN, JR, JJ, BM, SK
	Add home page to choose between apps	7	AN, DG,KN,PN, JR, JJ, BM, SK
	Add navbar to musicali page	8	AN, DG,KN,PN, JR, JJ, BM, SK
	Clean up & reorganize previous homepage code	4	AN, DG,KN,PN, JR, JJ, BM, SK
Bingo Board fixes	Implement suggested fixes, add pagination numbers	10	AN, DG,KN,PN, JR, JJ, BM, SK
	Total	35	

1. Frontend UI updates = 25 hours
2. Bingo board fixes = 10 hours

Total New Estimate = 35

35 hrs - 35 hrs = 0 hrs. This is the expected work estimation based on the initial work estimate given.

An

Work Item	Task	Estimation in pts	Team Confirmation
Item Creation (I.C.)	Front end Make button to change view to craftVerify, itemCreation form, validate the input, call AJAX fetch to upload file to sandbox and s3, create item and insert to database	4h40min	AN, DG,KN,PN, JR, JJ, BM, SK
	Back end Config aws s3 , make s3 model, s3 Dao, s3 APIs to upload to sandbox, up load to s3 Make item model, service, DAO, API to create item	10h10min	AN, DG,KN,PN, JR, JJ, BM, SK
	LLD fixing	30min	AN, DG,KN,PN, JR, JJ, BM, SK
Item Listing (I.L.)	Front end Create listing view, button, pagination, function to change view, call AJAX API to get item, modify the listing status	4h30min	AN, DG,KN,PN, JR, JJ, BM, SK

	Back end Create Item Listing DAO, controller, Service to handel pagination, get all item from a seller, modify the listing status and save change to database	5h	AN, DG,KN,PN, JR, JJ, BM, SK
	LLD fix pagination fail case	1h	AN, DG,KN,PN, JR, JJ, BM, SK
Offering	Front end Create listing view, button, pagination, function to change view, call AJAX API to get item, modify the price status.	4h30	AN, DG,KN,PN, JR, JJ, BM, SK
	Back end Create Item OfferingDAO, controller, Service to handle pagination, get all item from that offerable, modify the price status and save change to Craft receipt database	4h	AN, DG,KN,PN, JR, JJ, BM, SK
	LLD fix success and fail case of pagination	1h	AN, DG,KN,PN, JR, JJ, BM, SK

End-to-end Tests	End-to-end Tests	18 hr	AN, DG,KN,PN, JR, JJ, BM, SK
	Total	55.5 hours	

Total New Estimate = 55.5hrs

50 hrs - 55.5 hrs = -5.5 hours. This is the expected work estimation based on the initial work estimate given, but it has exceeded my work hours so I will have work extra hours on this sprint.

Parth

Work Item	Task	Estimation in pts	Team Confirmation
Price Range Sorting	Frontend Development • HTML Layout and Integration • CSS Styling • JS Functionality	4	AN, DG,KN,PN, JR, JJ, BM, SK
	Backend Development • Controller Setup • Service Layer Implementation • DAL	5	AN, DG,KN,PN, JR, JJ, BM, SK
	Deployment	1	AN, DG,KN,PN, JR, JJ, BM, SK
Analysis Dashboard Frontend	HTML Layout	10	AN, DG,KN,PN, JR, JJ, BM, SK

	CSS Styling	10	AN, DG,KN,PN, JR, JJ, BM, SK
	JavaScript Development	10	AN, DG,KN,PN, JR, JJ, BM, SK
	Total	40	

Total New Estimate = 40 hrs

40hrs - 40hrs = 0 hrs. This is the expected work estimation based on the initial work estimate given.

Khuong

Work Item	Task	Estimation in pts	Team Confirmation
Item Modification	1. Backend: ItemModificationController	1	AN, DG,KN,PN, JR, JJ, BM, SK
	2. Backend: ItemModificationService	2	AN, DG,KN,PN, JR, JJ, BM, SK
	3. Backend: ItemModificationDAO	1	AN, DG,KN,PN, JR, JJ, BM, SK
	4. Frontend: ItemModification.html	2	AN, DG,KN,PN, JR, JJ, BM, SK
	5. Frontend: ItemModification.js	2	AN, DG,KN,PN, JR, JJ, BM, SK

	6. Frontend: ItemModification.css	1	AN, DG,KN,PN, JR, JJ, BM, SK
Item Deletion	7. Backend: Item DeletionController	2	AN, DG,KN,PN, JR, JJ, BM, SK
	8. Backend: Item DeletionService	2	AN, DG,KN,PN, JR, JJ, BM, SK
	9. Backend: Item DeletionDAO	2	AN, DG,KN,PN, JR, JJ, BM, SK
	10. Frontend: ItemDeletion.html	2	AN, DG,KN,PN, JR, JJ, BM, SK
	11. Frontend: Item Deletion.js	2	AN, DG,KN,PN, JR, JJ, BM, SK
	12. Frontend: ItemDeletion.css	1	AN, DG,KN,PN, JR, JJ, BM, SK
Inventory Stock List	13. Backend: InventoryStock Controller	2	AN, DG,KN,PN, JR, JJ, BM, SK
	14. Backend: InventoryStock Service	2	AN, DG,KN,PN, JR, JJ, BM, SK
	15. Backend: InventoryStock DAO	2	AN, DG,KN,PN, JR, JJ, BM, SK

	16. Frontend: InventoryStock.html	2	AN, DG,KN,PN, JR, JJ, BM, SK
	17. Backend: InventoryStock.js	1	AN, DG,KN,PN, JR, JJ, BM, SK
	18. Backend: InventoryStock.css	2	AN, DG,KN,PN, JR, JJ, BM, SK
FinancialProgressReport	19. Backend: FinancialProgressReportController	2	AN, DG,KN,PN, JR, JJ, BM, SK
	20. Backend: FinancialProgressReportService	2	AN, DG,KN,PN, JR, JJ, BM, SK
	21. Backend: FinancialProgressReportDAO	3	AN, DG,KN,PN, JR, JJ, BM, SK
	22. Frontend: FinancialProgressReport.html	2	AN, DG,KN,PN, JR, JJ, BM, SK
	23. Frontend: FinancialProgressReport.js	2	AN, DG,KN,PN, JR, JJ, BM, SK
	24. Frontend: FinancialProgressReport.css	2	AN, DG,KN,PN, JR, JJ, BM, SK

End-to-end Tests	1. End-to-end Tests	10	AN, DG,KN,PN, JR, JJ, BM, SK
	Total	54	

Total New Estimate = 54hrs

50 hrs - 54 hrs = - 4 hours. This is the expected work estimation based on the initial work estimate given, but it has exceeded my work hours so I will have work extra hours on this sprint.

Team individual estimate total = 241

Final Analysis:

Do we accept the **Core Tasks**?:

Julie	Yes
Jason	Yes
Kihambo	Yes
Diego	Yes
Parth	Yes
Khuong	Yes
An	Yes

7Yes/0 No: We will accept the current effort estimation.

Do we accept **Individual Features**?:

Julie	Yes
Jason	Yes
Kihambo	Yes
Diego	Yes

Parth	Yes
Khuong	Yes
An	Yes

7 Yes/0 No: We will accept the current effort estimation.

Is Team Phoenix within our sprint capacity?:

	Original Estimate	New Team Estimate
Total	271	241

Team capacity - estimated hours:

285 - 241= 44 hours

We are assigning 21 more hours over our estimated capacity. I think this means some of the team is over estimating for this sprint planning, but seeing as it's our last sprint we can use these extra hours of capacity to work on finishing whatever fixes we need in the end. Last Sprint so we must finish strong