TroveSpace

Sprint 2 Planning Document

Team 13 | https://github.com/baker323/TroveSpace
Adam Baker, Kareem Elhadidi, Rei Orikata, Pritesh Kadiwala

Contents

Sprint Overview

For this sprint, we will focus on adding search functionality and the ability to edit and approve edits on collectibles. We will have several different types of search filters as well as the ability to add images to collectibles, create folders and add items to them, see dates of when items were added to the user's folders and wishlist, and add details to collectibles within a specific trove.

Scrum Master: Kareem Elhadidi

Meeting Schedule: Mondays, Wednesdays, and Fridays at 3:30PM

Risks/Challenges: We will be using Algolia for our search functionality, which no one on our team has experience using. We will have to create UI for our voting system and figure out how to ensure users can only vote once. Another challenge will be figuring out how to store images in the database because we are unfamiliar with this functionality as well.

Current Sprint Detail

User Story #9

As a user, I would like to search for items that already exist in the TroveSpace database

#	Task Description	Estimated Time	Owner(s)
1	Implement UI for search results	10	Kareem
2	Implement Algolia search methods	15	Adam
3	Implement error handling for UI	3	Pritesh
4	Perform tests on search functionality	2	Rei

Acceptance Criteria:

- Given that the frontend is set up correctly, when a user selects the collectibles search category, then all collectibles that match their search should be displayed.
- Given that the frontend is set up correctly, when a user selects the troves search category, then all troves that match their search should be displayed.
- Given that the frontend is set up correctly, when a user searches for something that does not exist, then they should be notified that no search results were found.
- Given that the frontend is set up correctly, when a user attempts to search on an empty string, then the search should not execute.
- Given that the database is setup properly, when a user sends a search query for collectibles, then it should only return matching collectibles.
- Given that the database is setup properly, when a user sends a search query for troves, then it should only return matching troves.

User Story #14

As a user, I would like to edit items that already exist in the TroveSpace database

#	Task Description	Estimated Time	Owner(s)
1	Implement UI for editing items	10	Pritesh
2	Implement Firebase methods to update items	10	Rei
3	Implement error handling for UI	2	Kareem
4	Perform tests on edit functionality	3	Pritesh

Acceptance Criteria:

- Given that the frontend is set up correctly, when a user edits an item, then the changes should not be saved until the user clicks save.
- Given that the frontend is set up correctly, when a user edits an item, then they are unable to edit the title.
- Given that the frontend is set up correctly, when a user clicks the edit button, then the fields should become editable.
- Given that the frontend is set up correctly, when a user has not clicked the edit button, then the fields should not be editable.
- Given that the frontend is setup correctly, when a user is editing an item and clicks cancel, then the fields should return to their original values.
- Given that the database is setup correctly, when a user sends an update request, then the updated data is stored in the “pending” fields.

User Story #16

As a user, I would like to vote on whether or not changes made to an item are accurate

#	Task Description	Estimated Time	Owner(s)
1	Implement UI for voting for edits	1	Kareem
2	Implement UI for displaying vote counts for approve and reject	2	Kareem
3	Implement Firebase methods to fetch vote counts	2	Adam
4	Implement Firebase methods to increment vote counts	2	Adam
5	Implement error handling for UI	1	Pritesh
6	Perform tests on vote functionality	2	Rei

Acceptance Criteria:

- Given that the frontend is set up correctly, when a user edits the information of a collectible, then it should show as a pending edit.
- Given that the frontend is set up correctly, when a user clicks on the option to approve a pending edit, then the vote count to approve that edit should increase by one.
- Given that the frontend is set up correctly, when a user clicks on the option to reject a pending edit, then the vote count to reject that edit should increase by one.
- Given that the frontend is set up correctly, when a user who has already voted tries to vote again, then they should see an informative error message.
- Given that the frontend is set up correctly, when a pending edit gets the required amount of votes, then it should become accepted if approved or be discarded if rejected.
- Given that the backend is set up correctly, when a user sends vote data to be stored, then their vote and unique identification should appear in the database.

User Story #20

As a user, I would like to create a new folder for items in a specific category

#	Task Description	Estimated Time	Owner(s)
1	Implement UI for creating a new folder for items	3	Pritesh
2	Implement Firebase methods to create a new folder	5	Adam
3	Implement error handling for UI	1	Kareem
4	Perform tests on edit functionality	1	Rei

Acceptance Criteria:

- Given that the frontend is set up correctly, when a user visits their collection page, then they should see an option to create a new folder.
- Given that the frontend is set up correctly, when a user creates a new folder, then they should be given an option to choose a category for that folder.
- Given that the frontend is set up correctly, when a user creates a new folder, then it should appear in their collection.
- Given that the frontend is set up correctly, when a user tries to create a folder with a duplicate name, then they should see the appropriate error.
- Given that the database is set up correctly, when a user creates a folder, then it should be added to their list of folders in the database.

User Story #21

As a user, I would like to add an item to a folder in my collection

#	Task Description	Estimated Time	Owner(s)
1	Implement UI for adding items to the folder	3	Kareem
2	Implement Firebase methods for adding an item to a folder	5	Rei
3	Implement error handling for UI	1	Pritesh
4	Perform tests on the add functionality	1	Rei

Acceptance Criteria:

- Given that the frontend is setup correctly, when a user visits the page for a specific trove, then they should see the option to add each collectible to their collection.
- Given that the frontend is setup correctly, when a user adds an item to their collection, then they should see the option to select which folder it belongs to.
- Given that the frontend is setup correctly, when a user adds an item to a folder, then the item should be recorded as being in that folder.
- Given that the frontend is setup correctly, when a user views a folder, then they should see the options to remove each collectible from their folder.
- Given that the backend is setup correctly, when a user adds an item to their collection, then it should be recorded in the database.

User Story #22

As a user, I would like to search for items in a specific Trove (a Trove is a category for items, for example: baseball cards, stamps, coins, etc)

#	Task Description	Estimated Time	Owner(s)
1	Implement UI for search results	5	Kareem
2	Implement Algolia search methods	5	Adam
3	Implement error handling for UI	2	Pritesh
4	Perform tests on search functionality	3	Pritesh

Acceptance Criteria:

- Given that the frontend is set up correctly, when a user visits a specific trove page, then they should see the option to search for collectibles in that trove in the search bar.
- Given that the frontend is set up correctly, when a user selects the trove category, then all collectibles that match their search should be displayed.
- Given that the frontend is set up correctly, when a user searches for something that does not exist, then they should be notified that no search results were found.
- Given that the frontend is set up correctly, when a user attempts to search on an empty string, then the search should not execute.
- Given that the database is setup properly, when a user sends a search query for collectibles in a specific trove, then it should only return matching collectibles.

User Story #37

As a user, I would like to search for items within my personal collection

#	Task Description	Estimated Time	Owner(s)
1	Implement UI for search results	5	Pritesh
2	Implement Algolia search methods	5	Adam
3	Implement error handling for UI	2	Kareem
4	Perform tests on search functionality	3	Rei

Acceptance Criteria:

- Given that the frontend is set up correctly, when a user selects the search bar, then they should see an option to search for collectibles in their collection.
- Given that the frontend is set up correctly, when a user presses the search button, then they should be taken to a page that shows collectibles that match their search.
- Given that the frontend is set up correctly, when a user searches for something that does not exist, then they should be notified that no search results were found.
- Given that the frontend is set up correctly, when a user attempts to search on an empty string, then the search should not execute.
- Given that the database is setup properly, when a user sends a search query for collectibles, then it should only return matching collectibles in their collection.

User Story #38

As a user, I would like to see the date an item was added to my collection

#	Task Description	Estimated Time	Owner(s)
1	Implement UI for displaying time	1	Pritesh
2	Update Firebase method to return timestamp	1	Rei
3	Create Firebase method to store timestamp	2	Adam
4	Perform tests on timestamps	1	Rei

Acceptance Criteria:

- Given that the UI is set up correctly, when a user views an item in their collection, then they should be able to see when it was added.
- Given the the UI is set up correctly, when a user views the time an item was added, then the time displayed should match their local timezone.
- Given that the UI is setup correctly, when a user views the timestamp, then the time should be formatted properly (MM/DD/YYYY HH:MM) as this is not the format it is stored in the database.
- Given that the database is set up correctly, when a user adds an item to their collection, then the time should be recorded.
- Given that the database is setup correctly, when a user sends a request for an item, then the database should return the timestamp along with the other data.

User Story #39

As a user, I would like to see the date an item was added to my wishlist

#	Task Description	Estimated Time	Owner(s)
1	Implement UI for displaying time	1	Kareem
2	Update Firebase method to return timestamp	1	Rei
3	Create FIrebase method to store timestamp	2	Adam
4	Perform tests on timestamps	1	Rei
5	Implement UI for displaying time	1	Kareem

Acceptance Criteria:

- Given that the UI is set up correctly, when a user views an item in their wishlist, then they should be able to see when it was added.
- Given the the UI is set up correctly, when a user views the time an item was added, then the time displayed should match their local timezone.
- Given that the UI is setup correctly, when a user views the timestamp, then the time should be formatted properly (MM/DD/YYYY HH:MM) as this is not the format it is stored in the database.
- Given that the database is set up correctly, when a user adds an item to their wishlist, then the time should be recorded.
- Given that the database is setup correctly, when a user sends a request for an item, then the database should return the timestamp along with the other data.

User Story #40

As a user, I would like to upload an image when creating an item

#	Task Description	Estimated Time	Owner(s)
1	Implement UI for uploading an image	5	Pritesh
2	Implement Firebase method for uploading an image	10	Rei
3	Implement error handling for UI	3	Kareem
4	Perform tests on upload functionality	2	Rei

Acceptance Criteria:

- Given that the frontend is properly implemented, when a user uploads an image, then it should be displayed on the page.
- Given that the frontend is properly implemented, when a user uploads an unsupported format, then they should receive an error.
- Given that the frontend is properly implemented, when a user does not upload an image, then they should be able to submit the item.
- Given that the frontend is properly implemented, when a user edits an item, then they can upload a new image to replace the current one.
- Given that the database is setup properly, when a user uploads an image, then it should be stored in the database.

User Story #41

As a user, I would like to add details about my item based on the item's Trove

#	Task Description	Estimated Time	Owner(s)
1	Implement dynamic UI for item creation	10	Kareem
2	Implement Firebase method	3	Adam
3	Implement error handling for UI	1	Pritesh
4	Perform tests on the add functionality	1	Rei

Acceptance Criteria:

- Given that the frontend is implemented properly, when a user creates a new item, then they should have the option to select which trove it belongs to.
- Given that the frontend is implemented properly, when a user creates a new item, then the available fields should be consistent with the trove that the item belongs to.
- Given that the frontend is implemented properly, when a user does not enter a description, then they should receive an error.
- Given that the frontend is implemented properly, when a user does not enter a title, then they should receive an error.
- Given that the frontend is implemented properly, when a user enters a title that is already in use, then they should receive an error.
- Given that the database is properly implemented, when a user requests to create a new item in a specific trove, then the database should return the list of fields.

Total Sprint Workload

Developer's Name	**Estimated Time**
Adam Baker	41 Hours
Kareem Elhadidi	41 Hours
Rei Orikata	38 Hours
Pritesh Kadiwala	38 Hours
Total	**160 Hours**

Remaining Backlog

ID	Function Requirement	Hours
13	As a user, I would like to vote on whether or not an item should be marked as a duplicate	20
15	As a user, I would like to be notified when an item in my collection is edited by another user	15
17	As a user, I would like to be able to see the edit history of an item (time permitting)	30
18	As a user, I would like to see the number of people who have a specific item in their collection	15
19	As a user, I would like to see the number of people who have a specific item in their wishlist	10
25	As a user, I would like to view another user's collection	15
26	As a user, I would like to view another user's wishlist	10
27	As a user, I would like to rate items on a 5 star scale	15
28	As a user, I would like to comment on items	15
29	As a user, I would like to offer trades/purchases to other users (if time permits)	25
30	As a user, I would like to rate trades/purchases with other users (if time permits)	20
31	As a user, I would like to have items removed from my wishlist automatically when I add them to my collection	20
32	As a user, I would like to see a list of all users who own a specific item	15
33	As a user, I would like to follow a Trove to see when new items are added	20

34	As a user, I would like to add friends (if time permits)	20
35	As a user, I would like to be notified when my friends add a new item to their collection (if time permits)	15
36	As a user, I would like to be notified when my friends add a new item to their wishlist (if time permits)	10
42	As a user, I would like to request for a Trove to be removed	5
43	As a user, I would like to mark items in my collection as “for sale” (if time permits)	10
44	As a user, I would like to be able to see all “for sale” listings of a specific item (if time permits)	15

Non-Functional

- Application will be adequately responsive and easy to use.
- Application will be fully functional on Chrome (and Firefox if time permits).
- Application will be intuitive to use.
- Application will have a visually appealing UI.
- Application will store the user data safely and securely on Firebase.
- Application will securely transmit data to and from Firebase.
- Application will resize page content based on window size.
- Application will have cleanly organized and well-commented code.
- Application will be maintainable and testable.
- Application will produce useful error messages when required.