Sprint 3 Retrospective

Team 11: Ryan Huff, Sam Kravitz, Akash Lankala, Raziq Raif Ramli, Tyler Stanish, Blake Steel

What went well?

General:

This sprint saw a lot of "supplemental" features get added into the game, expanding on the basic game functionalities that were implemented in the last sprint. These features from this sprint make the application seem more well-rounded and more like an interactive game. While they aren't necessarily essential for gameplay, they still add to the experience and create a more immersive and competitive environment for players. We also managed to iron out some bugs that existed from prior sprints to make playing our game more seamless.

User Story Specific:

User Story # 13: Landing Page

As a player, I would like to see a landing page that includes the current prices of several cryptocurrencies.

#	Description	Estimated Time	Owner

5	Create tests to ensure that an arbitrary number of information components are displayed on the landing page	1 hr	Ryan
6	Create automated tests to ensure displayed data matches current data.	1 hr	Ryan
7	Update implementation to using a common Redux store between different pages	1 hr	Ryan

This story was relatively simple and was leftover from the previous sprint. Both automated tests and a manual testing document were created for this story and added to the project in the landing page directory.

User Story # 14: Landing Page Graphs

As a player, I would like to see a landing page that shows graphs of historical data of cryptocurrencies.

#	Description	Estimated Time	Owner
5	Create tests to check for error handling and displaying a blank graph if errors persist.	30 mins	Ryan

Error handling tests were created and fixes to graph plotting were made to ensure error handling when erroneous data is attempted to be graphed (such as an absence of data, a price of zero, etc.)

User Story # 21 Coin table graphs

As a player, I would like to see a time series graph displaying the historical price of a cryptocurrency on the game page.

#	Description	Estimated Time	Owner
1	Create a backend api for historical data based on a given time span	1 hr	Ryan
2	Create graphs for each coin	1 hr	Ryan
3	Combine historical data with live data to	2 hrs	Ryan

	continuously update the graph for whatever the selected time span is.		
4	Create deeply-rendered black box visual tests	1 hr	Ryan

The game page is fully operational, including displaying coin graphs that update in real time and display historical data based on the user's selection. The efficiency of requesting that coin data from the back end was also increased by creating a new data type for both coin data and pricing data.

User Story # 23: Get Coins by game ID (filtered)

As a player, I would like to modify the time span of crypto data to display (min/hr/day/month/yr), so that I may modify the data view to fit my buying and selling needs.

#	Description	Estimated Time	Owner
1	Frontend: Send time span and above sort by categories	2 hr	Blake

We finally got the frontend connected to the backend well, and used this to inform later connections. The game page displays the proper information when requested.

User Story # 25: Liquefy endpoint

As a player, I would like to be able to liquefy all current assets immediately.

#	Description	Estimated Time	Owner
1	"Liquefy" endpoint, which sells all of a users' coins	3 hrs	Blake
2	Create black box integration tests	1 hr	Blake

We were able to get this story completed since we completed its dependencies. It is operational on the game page and works as expected.

User Story # 26: Navigate to Leaderboard

As a player, I would like to be able to navigate to the leaderboard for the current game.

#	Description	Estimated Time	Owner
1	~~Create "Leaderboard" game page that can be accessed by players~~	2 hrs	Akash

Acceptance Criteria:

- Given any game, when I navigate to that game's leaderboard page, it should call a get highscores api.

This was a story from the previous sprint. The last acceptance criteria, "get high scores api", was implemented in this sprint. No huge blockers here except the research involved in understanding API calls.

User Story # 28: Serve Live Cryptocurrency Prices

As a developer, I would like to serve live cryptocurrency prices (and potentially other cryptocurrency statistics) to the user with websockets.

#	Description	Estimated Time	Owner
1	Debug client socket.io connection problems in production	2 hrs	Tyler

After some significant research and debugging, we finally got websocket messages to appear without delay on the production server. Previously notifications and price changes would take up to 30 seconds to be received on the client because the production client was using the polling implementation (default behavior for socket.io in production mysteriously) instead of the websocket implementation.

User Story # 39: Leaderboard Notifications

As a player, I would like to receive in-app notifications if there are any significant movements in the leaderboard.

#	Description	Estimated Time	Owner
0	Dependency on User Story #62	0 hrs	Akash
1	Write scripts to notify user if there are any large movements on the leaderboard	3 hrs	Akash

This was a story from Sprint 2. From having references and assistance within the project on creating notifications, this story went by rather fine.

User Story # 40: Track Achievements

As a user, I would like to have a trophy system to keep track of my achievements and have some major goals to look forward to in this game

#	Description	Estimated Time	Owner
0	Dependency on User Story # 42: Profile Page	0 hr	Blake
1	Create the achievements tab on the profile page	1 hr	Sam
2	Come up with list of achievements	1 hr	Sam (+ Group)
3	Store list of achievements with achievement IDs in database	3 hr	Sam
4	Create ProfileAchievement table in the database to track what achievements a player has achieved.	3 hr	Sam
5	Write a method on the backend to determine if a player has accomplished any achievements and update their ProfileAchievement table if applicable	3 hr	Sam
6	Create a friendly UI and cool icons that make user's experience navigating to the achievements page enjoyable	2 hr	Sam

This story was fully implemented. It was created with extensibility in mind so more achievements can easily be added and tracked and no additional frontend work will be required to display more achievements.

User Story # 41: Generate Goals

As a user, I would like to be given a set of goals that I can accomplish every week to motivate me to come back to the game from time to time.

#	Description	Estimated Time	Owner
0	Dependency on User Story # 42: Profile Page	0 hr	Blake
1	Create the weekly goals tab in the profile page	1 hr	Sam
2	Come up with weekly goals	1 hr	Sam (+ Group)
3	Store list of weekly goals with goal IDs in database	3 hr	Sam
4	Create ProfileGoal table in the database to track what goals a player has achieved.	3 hr	Sam
5	Write a method on the backend to determine if a player has accomplished any goals and update their ProfileGoal table if applicable	3 hr	Sam
6	Create a friendly UI and cool icons that make user's experience navigating to the goals page enjoyable	2 hr	Sam

This story was also fully completed. This story was relatively easy to implement following the achievements. The main difference being that goals can be swapped out weekly to entice players to return to Fortune week after week.

User Story # 42: Profile Page

As a user, I would like to have a page that displays my profile's details.

#	Description	Estimated Time	Owner
1	A profile button should be added to the menubar dropdown	15 min	Blake
2	Clicking the profile button should navigate to a profile page	30 min	Blake
3	The page should have the username as a title and a	15 min	Blake

	button that takes the player to the play page.		
4	The page should have a tabbing system to account for the things that will be added with a sample tab “options”	2 hr	Blake
5	The page should be manually tested to verify that it is navigated to and has the username as a title.	30 min	Blake

This was completed early enough to fulfill all dependencies by other stories. A profile page is available to each user to display information such as achievements and friends as well as a section for the ability to change their username and password.

User Story # 43: Change Username

As a user, I would like to be able to change my username

#	Description	Estimated Time	Owner
0	Dependency on User Story # 42: Profile Page	0 hr	Blake
1	Create a text box that allows the user to enter a new username	15 min	Blake
2	Create a button that says “Change Username”	5 min	Blake
3	Setup redux action handler	1 hr	Blake
4	Create a change username endpoint that updates the username in the backend	2 hr	Blake
5	Create black box integration tests	1 hr	Blake

This story was accomplished with relative ease and works as expected when a user tries to change their username. When I implemented this, I was able to set a good framework for the profile page.

User Story # 44: Change Password

As a user, I would like to be able to change my password

#	Description	Estimated Time	Owner

0	Dependency on User Story # 42: Profile Page	0 hr	Blake
1	Create a text box that allows the user to enter a new password and a box for old password	15 min	Blake
2	Create a button that says “Change Password”	5 min	Blake
3	Setup redux action handler	1 hr	Blake
4	Create a change password endpoint that updates the hashed password in the backend	2 hr	Blake
5	Create black box integration tests	1.5 hr	Blake

This story was fully completed, operational, and tests were created for it. Along with its implementation, I also went through and made changes to story 43 to make it cleaner.

User Story # 46: Add Friends

As a user, I would like to be able to add friends, so that I can compete with them and compare our progress

#	**Description**	**Estimated Time**	**Owner**
1.	Conceptually create a new table that is used to indicate that two users are friends	30 mins	Ryan
2.	Implement the new table into our ORM	30 mins	Ryan
3.	Create option in menu bar that opens up a modal for sending and accepting friend requests	1hr	Ryan
4.	Create backend API endpoint for sending and accepting a friend request	2 hr	Ryan
5.	Update Redux store to call on send/accept friends endpoint	2 hr	Ryan
6.	Test that the backend response works and returns an error when adding a user that does not exist.	1 hr	Ryan
7.	Test frontend UI displays friends correctly and displays an error when adding a user that does not exist.	1 hr	Ryan

The adding friends feature is fully operational and users can add friends via the friends tab on their profile page and that relationship is created and stored in the database. Tests were also created to ensure error handling was implemented properly.

User Story # 47: Friends List

As a user, I would like to be able to view and manage my friends list

#	Description	Estimated Time	Owner
0	Dependency on User Story # 42: Profile Page	0 hr	
1.	Create backend API endpoint for getting a user's friends from the database	2 hr	Ryan
2.	Add to the Redux store to call the endpoint on the frontend	2 hr	Ryan
3.	Parse data received by the backend and render table with user's friends	2 hr	Ryan
4.	Handle the case when a user does not have any friends associated to their profile	1 hr	Ryan
5.	Manually test that tables paginate, correct data appears in the tables, and the "no friends" message displays.	1 hr	Ryan
6.	Manually test that the backend sends correct data from the database.	1 hr	Ryan

Users can view their friends and pending friend requests under the friends tab in the profile page. They can accept pending friend requests and see them get added to their friends list.

User Story # 49: Current Leader

As a user, I would like to see the current leader of any particular game in a window on the game screen.

#	Description	Estimated Time	Owner

1.	Research and create a “Current Leader” box component to be displayed in the game screen to display the current leader	1.5 hr	Akash
2.	Fetch data on current leader of the game from the leaderboard page, insert information into “Current Leader” box on game screen	2 hr	Akash
3.	Write tests to ensure that current leader of the game displayed on game screen corresponds to the #1 rank of the leaderboard for that game	1.5 hr	Akash

The current leader feature was rather simple to implement with other features in the leaderboard already implemented (data fetching and sorting), and thus did not face many blockers here. The only feature in this story that did not get implemented (due to re-prioritization) was having the current leader displayed on the game screen.

User Story # 51: Leaderboard Page Push Notifications System

As a developer, I would like to lay out the architecture and code for notifications to be processed in the leaderboard.

#	Description	Estimated Time	Owner
1.	Research and create a service worker that manages the notifications in the leaderboard page. Register the service worker	2.5 hrs	Akash
2.	Handle push event, trigger push message to display notification using the service worker	1.5 hrs	Akash
3.	Manual testing to see if notifications appear when prompted by the developer	30 mins	Akash

Similar to Story 39, this story went by fine largely because there was code related to notifications implemented elsewhere in our project that could be referenced. In addition to that, getting key questions answered by certain members of our team on the approach to implementing this story resulted in the work to go by much smoother and end up with an implementation that automatically satisfied the acceptance criteria while maintaining the appearance to match notifications that appear on other parts of the application.

User Story # 52: Popup Notification System

As a user, I would like to have a notification popup bar where users can get notified of any new updates, friend requests, and game invitations

#	Description	Estimated Time	Owner
1	Integrate with websockets on the server to send users the notifications	2 hrs	Tyler
2	Integrate with websockets on the client to receive notifications	1 hr	Tyler
3	Display a UI notification when a notification is received	1 hr	Tyler
4	Create authentication mechanism to allow private per-user messages over websockets	4 hrs	Tyler
5	Create a hook to send a notification when a user receives a friend request	1.5 hr	Tyler
6	Create a hook to send a notification when a user receives an invitation	1.5 hr	Tyler
7	Research sending socket.io messages to select individual users and authentication	2 hrs	Tyler
8	Test notification hooks actually create notifications	1 hr	Tyler

Anywhere on the backend can send a user a notification. If the user is online and the server sends a user a notification, it appears on the user's screen and it is stored in the database for later retrieval. Even if the user is not online the notification will still be stored in the database.

User Story # 53: Price Change Notification

As a user, I would like to receive in-app notifications for large movements in crypto price.

#	Description	Estimated Time	Owner
1	Create endpoint for users to create pricing alerts	2 hrs	Tyler

2	Create page/form on the client to allow users to create price alerts for coins	3 hrs	Tyler
3	Create page on the client to view price alerts that users are subscribed to	3 hr	Tyler
5	Create endpoint to fetch a user's price alert subscriptions	1 hr	Tyler
6	Integrate with real-time data service to send notifications upon price updates	2 hrs	Tyler
7	Error handling on backend	1 hr	Tyler
8	Test error handling on backend	1 hr	Tyler
9	Test fetching a user's price alert subscriptions returns only their own subscriptions	1 hr	Tyler

A user can create price alerts when a coin's price goes above or below a specific price and these notifications are handled as notifications. This story was completed even with all of the moving parts of integrating with the real time data service and real time notifications.

User Story # 54: Notification Widget

As a user, I would like to see a widget that displays notification history throughout the game.

#	**Description**	**Estimated Time**	**Owner**
1	Create page on frontend to display a user's received notifications	3 hrs	Tyler
2	Create endpoint on backend to fetch a user's notifications	1 hr	Tyler
3	Paginate a user's notifications on the client	1 hr	Tyler
4	Paginate a user's notifications on the server	1 hr	Tyler
5	Write tests to ensure users can only see their own notifications	1 hr	Tyler

Users can retrieve notifications from the notifications tab and are paginated if there are more than a couple. Users can also only see their own notifications in both the notification history of received notifications but also upon new notifications incoming in real time.

User Story # 55: Game Chat

As a user, I would like to chat with other players who are in the same game session with me.

#	Description	Estimated Time	Owner
1	Research and implement the group chat component on the frontend.	3 hours	Raziq
2	Implement database tables to store messages.	45 minutes	Raziq
3	Create an API endpoint to update the database table when a user sends a new message.	1.5 hour	Raziq
4	Integrate the send message API to the frontend.	1 hour	Raziq
5	Test the database is updated properly when the endpoint is used.	30 minutes	Raziq
6	Create API endpoints to get chat messages in the database. These endpoints need to be optimized to minimize the size of data transfer.	2 hours	Raziq
7	Test correct result is returned when the endpoint is used.	30 minutes	Raziq
8	Research and implement a websocket to automatically call the get message API from the frontend every time a new message is added into the database.	3 hours	Raziq

The main blocker for this story is implementing the frontend component. I have a relatively hard time finding a package that works with typescript and allows me to implement the backend part of the chat feature myself. Then, the package that I ended up using does not come with good documentation and demo codes. There is also an additional API that I did not take into account before but needs to be implemented, which is getPlayersInAGame(). However, the chat component and APIs were successfully implemented and integrated to the game page. Users can communicate with each other through it.

User Story # 56: Admin Page

As a user, I would like to have an admin page to manage players and enforce game policies

#	Description	Estimated Time	Owner
1	Dependency on User Story # 50: Popup Notification	-	-
2	Set up a new page.	30 mins	Raziq
3	Implement a component to broadcast a message to a specific user or all users. This should utilize the API from User Story # 50.	1 hour	Raziq
4	Create an API endpoint to ban a player account.	1.5 hour	Raziq
5	Test that the API removes a player correctly.	30 minutes	Raziq
6	Implement database tables to store report tickets and admin accounts. Also, add a seeded admin account in the database.	1 hour	Raziq
7	Implement a component to view all reports that were made by players. The list should be paginated and the admin should be able to sort it by the status of the report.	3 hours	Raziq
8	Implement an API endpoint to get all reports from the database.	1.5 hours	Raziq
9	Test that the API returns correct reports.	30 mins	Raziq
10	Implement a modal window to view an individual report, mark it as processed, and send a warning to the reported account or remove the account from the game. The warning feature should utilize the API from User Story # 50.	1.5 hours	Raziq
11	Create an API endpoint to update the status of a report.	1.5 hours	Raziq
12	Test that the API updates the status of a report correctly.	30 mins	Raziq
11	Integrate the created APIs to the frontend	2 hour	Raziq

Because of the blocker from the previous story, I could not work on and finish the frontend part in time, but Blake helped me with that part and the admin page was successfully completed. Also, I need to implement two extra APIs in this story that I did not anticipate at the beginning, which are checkIfAdmin() and getUsers(). In the end, all functionalities were implemented. The admin can notify, warn, and ban users from the admin page as well as process report tickets. Non-admin users are also prevented from accessing the admin page as well as utilizing the mentioned APIs.

User Story # 57: Reporting Players

As a user, I would like to report players who make any form of communication abuse.

#	Description	Estimated Time	Owner
1	Dependency on User Story # 53: Game Chat	-	-
2	Dependency on User Story # 54: Admin Page	-	-
3	Implement a report button next to each received message in the chat component.	15 minutes	Raziq
4	Create an API endpoint to create a new report ticket in the database.	1.5 hour	Raziq
5	Test that the API updates the database correctly	30 minutes	Raziq
6	Integrate the API to the frontend.	1 hour	Raziq

In the chat component, there is a report button next to each message that the user receives. Clicking the button will automatically create a report ticket about the message and send it to the admin for review.

User Story # 58: Interface Correction

As a developer, I would like to make an intuitive, straightforward, and interactive interface.

#	Description	Estimated Time	Owner
1	Do field testing (ask friends and family to use the app, do not give help or ask leading questions), note any confusions about the interface	4 hrs	Blake

2	Correct confusing UI elements.	3 hrs	Blake

Field testing was conducted and tweaks were made to the navigation of the game. Upon completion, the interface looked readable and navigable.

User Story # 59: Navigation Correction

As a developer, I would like the players to be able to navigate through the application and execute trades seamlessly.

#	Description	Estimated Time	Owner
1	Do field testing (ask friends and family to use the app, do not give help or ask leading questions), note any confusions about navigation	2 hrs	Ryan
2	Correct confusing navigation elements	2 hrs	Ryan

Field testing was done and notes on confusion were taken, particularly when first seeing the game's landing page. Notifications were created when the user first registers for an account explaining where various aspects of the game are located and how to get there.

User Story # 62: Responsive Application

As a developer, I would like the application to be responsive to all requests made by the user.

#	Description	Estimated Time	Owner
1.	Stress test the frontend by clicking many buttons, different parts of the UI, etc. in rapid succession.	1 hr	Ryan
2.	Stress test the backend by sending many different requests to the API, such as registering for multiple accounts at a time.	1 hr	Ryan
3.	Implement preventative measures into the API to prevent multiple request from trying to be fulfilled	2 hrs	Ryan

A testing document was created that enumerates various tests for both the frontend and backend to ensure the application continues responding and does not crash upon receiving multiple inputs or requests.

What did not go well?

General:

For this sprint, we aimed for the low side in terms of estimating time for tasks. While this resulted in hours that weren't inflated on our sprint planning document, some stories ended up taking a lot more time than was originally allocated. That meant some developers were not able to get to some of their stories. The abundance of stories remaining from sprint 2 also hindered development time for the "supplemental" features in this sprint, trying to catch up on those features that are essential to gameplay.

User Story Specific:

User Story # 18: Navigating Current Active Games

As a player, I would like to be able to navigate to any of my currently active games.

#	Description	Estimated Time	Owner
7.	Write an API test to ensure that correct results are returned when the user filters games.	30 mins	Raziq
9.	Write a test to ensure that only games on the requested page are returned by the API.	30 mins	Raziq
10.	Write an API test to ensure that correct results are returned when the user sorts games according to date and title.	1 hour	Raziq

These are the remaining tasks for this story from Sprint 2. All of those tests were already implemented. However, I could not make them to work along with other existing tests in time and therefore the changes did not get merged to the development branch.

User Story # 19: Joining Private Games

As a player, I would like to be able to join a private game that has been created.

#	Description	Estimated Time	Owner
6.	Write a test to ensure that the user was added into the game properly.	30 mins	Raziq

Same as #18. This is the remaining task for this story from Sprint 2. The test was already written, but I could not make it to work with other existing tests in time, and therefore it did not get merged to the development branch.

User Story # 38: See Players Usernames and Net Values

As a player, I would like to see players' usernames and current net values on the leaderboards.

#	Description	Estimated Time	Owner
1	~~Create table to display leaderboard page information~~	3 hrs	Akash
2	Create SQL queries to fetch users net worth along with their usernames, sort by decreasing net worth to be placed in hiscores page	7 hrs	Akash
3	~~Display current leader of hiscores~~	30 min	Akash
4	Create tests to ensure correct values for hiscores leaderboard are displayed based on games	2 hrs	Akash
5	~~Ensure smooth UI on leaderboard page~~	1 hr	Akash

This was a story from Sprint 2. Creating the query was rather difficult as it involved understanding all the technologies we are using and how they interact with each other (React, Redux, Typescript, Python, Flask, Docker, etc.). However the challenge did not stop after the query was created. Testing became an issue as further work was needed in Leaderboard.tsx and elsewhere for the query to actually function in displaying appropriate information (usernames and scores) to the leaderboard table. This was a step I didn't anticipate at the beginning and resulted in a huge delay and consequently all the other stories which all depended on this one to

work were put in the backlog. Because of even further delays at the time (code conflict resolutions, docker startup debugging), this story had to be placed on hold while an alternative solution was implemented because of short time. Ultimately, a change in approach was implemented last minute in order to have certain leaderboard functions be demonstrable.

User Story # 48: Invite Friends to Game

As a user, I would like to invite my friends directly by selecting their usernames

#	Description	Estimated Time	Owner
1.	Create a new table for game invites and add to our ORM	30 mins	Ryan
2.	Create API endpoint to create game invite entries in the database after an invitation is sent	1 hr	Ryan
3.	Create a component for the create game page to add and remove friends being invited to the game.	1.5 hr	Ryan
4.	Use Redux store to call on API endpoint to tell backend to add game invite entry to backend	30 mins	Ryan
5.	Implement an API endpoint to get game invites for displaying invitations	1 hr	Ryan
6.	Render pending requests onto the select a game page with the option to accept or reject invites.	1 hr	Ryan
7.	Create backend endpoints for sending, accepting, and rejecting invites	1 hr	Ryan
8.	Create a GameProfile database entry if a user accepts a game, and delete a request if a user rejects a game.	1 hr	Ryan
9.	Manually test that the component on the create a game page works and they select a game cards display properly	1 hr	Ryan

This story was a lot more in depth than what it initially seemed. I thought that with the friends feature implemented it would be trivial to invite players via their friends. This was not the case and as a result this story did not get accomplished.

User Story # 50: Filter Leaderboard

As a user, I would like to filter the leaderboard by friends.

#	Description	Estimated Time	Owner
1.	Dependency on User Story #46: Friend's List	0 hrs	Akash
2.	Create dropdown option on leaderboard page with button "filter by friends", "filter by game", "filter by global"	2 hrs	Akash
3.	For "fitler by friends", fetch the user's friends list data from the backend	1.5 hrs	Akash
4.	Write scripts to filter leaderboard such that only the user_IDs that exist on a particular user's friends list are displayed on that user's leaderboard	1.5 hrs	Akash
5.	For "filter by game" option, the leaderboard should revert back to its original format of users by game	1 hr	Akash
6.	For "filter by global", write scripts to fetch data of all players among all games. Sort and display.	1.5 hrs	Akash
7.	Write black box integration tests to test "filter by friends", "filter by game", and "filter by global"	1 hr	Akash

This story ended up not getting implemented due to huge delays in the other stories and a priority to complete and test the leaderboard stories from the previous sprint. In addition to that, there was a dependency to have the friends feature implemented. There was also a challenge faced in querying the data appropriately, which was given precedence over this story.

User Story # 60: Color Correction

As a developer, I would like to use a pleasing color scheme that will add to the aesthetic of the application. The color palette that I plan to use can be viewed here https://coolors.co/4aa7d6-2a628f-13293d-79b473-db504a.

#	Description	Estimated Time	Owner

1	Edit styling so the above colors are used.	2 hrs	Ryan

This story has a dependency on nearly every other story / feature in the project. These features did not all make it in with enough time to complete color correction. Perhaps it would've been better to assign everyone to adhere to the color styling for their own features rather than one person fixing them all at the end.

How should we improve?

Having sprint 3 conducted entirely remotely was a challenge. As it was the final sprint and we were all accustomed to the work expected of us, some of the pressures of working remotely were relieved. This sprint was a learning experience in terms of communication in remote environments. If we had to do this sprint again, one of the ways we would improve would be increasing the amount of communication. Sometimes, having just a few 30 minute Zoom meeting sessions a week was not enough for us to all remain on the same page in terms of project direction and workload. But, given the circumstances, we did a pretty good job in terms of staying on track for development.

In some of the stories, we had also underestimated the time that is required to finish them. There are also some tasks that we did not expect needed to be done, but are integral to the completion of the story. Completing those additional tasks required us to put in more hours than what we had originally allocated at the start of the sprint. That impeded us from finishing our stories in a timely manner as well as caused us to not be able to finish some of our other planned stories. Planning the tasks for our stories more carefully in the future should prevent similar issues from happening again.