VirtuCards

CS 30700 Sprint 1 Retrospective

TEAM 8

June Seo

Ryan Hawks

Aryan Wadhwani

Kade Boltjes

Shayne Marques

Umang Sharma

What went well during Sprint 1?

We were able to familiarize ourselves with the new technologies we are working with - Unity, Photon and Firebase.
We were able to set up a solid foundation for our host and client applications.
We built in most of the basic functionality of our game, which will allow us to focus more on interface and gameplay in future sprints.
We worked well as a team while fixing bugs, organizing development tasks, and drafting documents for the project.

User Story #1
As a user hosting the game, I would like to be able to create a game lobby on my laptop/desktop as a common external screen, acting as a table.

#	Description	Estimated Time	Owner
1	Have a landing screen with a *Create Game* button, login/registration options, user profile link and a settings button on the host computer.	5 hours	Kade
2	Create a screen that displays the join-code, a setting for the maximum number of players, the selected game, and a cancel button.	4 hours	Aryan
3	List all the players in the room.	3 hours	Shayne
4	Add a start game button when the lobby has enough players in it.	2 hours	Shayne
5	Create unit tests that test the game lobby GUI.	2 hours	Aryan

<u>Completed</u>
When the host user opens the application for the first time, they are able to view a landing screen with buttons for the different sign in options. After they complete the sign in process, they are taken to a waiting room screen that displays the unique join code, a dynamic list of players in the room, the selected game, a cancel button and a start game button.

User Story #2

As a user hosting the game, I would like to create a game lobby easily.

#	Description	Estimated Time	Owner
1	Set up a Photon account and connect it with the game application. Also set up all users to connect to appropriate regions.	3 hours	Umang
2	Generate a room with the host name on Photon.	2 hours	June
3	Create a unique code and add it to Photon as a room.	1 hour	Umang
4	Setting the maximum number of players and the selected game.	1 hour	June
5	Implement unit tests to debug creating Photon rooms with the unique code.	2 hours	Umang

Completed

We were able to set up the Photon server for the game, which allowed us to implement a lot of the multiplayer functionality. We were successfully able to create a room with the required specifications.

User Story #3

As a user hosting the game, I would like to be able to allow my friends to join my game using a generated join code.

#	**Description**	**Estimated Time**	**Owner**
1	Create a method for connecting using a join code.	2 hours	Kade
2	Attempt to connect the client to the game room.	2 hours	Ryan
3	Throw error if joining the room fails.	1 hour	Ryan
4	Return room information if joining is successful.	1 hour	Ryan
5	Create a method to disconnect the client from the room.	2 hours	Aryan
6	Create unit tests for clients joining a game room.	2 hours	Shayne

Completed

We created the methods required to join a game with the unique join code and we also ensured that appropriate error messages were thrown when an invalid code is entered. We were able to connect a client to the game room and handle the scenario when a user decides to leave the game room.

User Story #4

As a user, I would like to join the game host's common screen on my mobile device through the join code the host provides.

#	**Description**	**Estimated Time**	**Owner**
1	Create Landing Screen for Mobile app with *Join Game* text field and button, login/registration options, user profile link and settings button.	5 hours	Aryan
2	Connect *Join Game* button to method. Show error message for incorrect join code.	2 hours	Ryan
3	For correct join code, display a "Waiting…" screen while users wait for the lobby to fill up for mobile clients with a *Leave Game* button and information about game settings.	4 hours	Umang
4	Create unit tests that correct settings are displayed, and buttons are functional.	2 hours	June

<u>Completed</u>

On the client application, a completed landing page is now visible. It contains buttons for the different sign in options and a screen to enter the join code for the game created by the host. The client application also contains a different waiting screen that is displayed until the host starts the game.

User Story #5

As a user, I would like to be able to register for an account for VirtuCards.

#	**Description**	**Estimated Time**	**Owner**
1	Set up the Firebase authentication and realtime database and connect host and mobile apps to it.	3 hours	Aryan
2	Set up a testing Firebase database.	2 hours	Aryan
3	Create a registration method in the host/client apps.	2 hours	Aryan
4	Check if the email is valid.	1 hour	Ryan
5	Check password strength.	1 hour	Ryan
6	Disable the register button until all registration fields are filled accurately.	3 hours	Shayne
7	Check if the account already exists with the same email or username, if so, display an error message.	2 hours	Kade
8	Add account to Firebase.	2 hours	Aryan
9	Implement unit tests to verify that user accounts can be created.	3 hours	Kade

<u>Completed</u>

Those of us that were unfamiliar with Firebase, watched tutorials online to get acquainted with the service and how it works in Unity. Following that process, we were able to work together to build authentication into our app through Firebase. This entailed setting up the database, creating methods within the scripts to sign in and register. We also perform some input validation on the client and host application, which throws appropriate warnings and errors when the credentials are invalid or do not meet the specifications.

User Story #6

As a user, I would like to be able to login into my account for VirtuCards.

#	**Description**	**Estimated Time**	**Owner**
1	Create a login method	2 hours	June
2	Validate entered credentials using Firebase	3 hours	June
3	Display an error message for invalid credentials	1 hour	June
4	Create a confirmation message if the login was successful	1 hour	June
5	Design unit tests to verify that a user can log into an existing account	3 hours	Ryan

Completed

When a user enters a login page, they will be shown an “enter email” and an “enter password” input box. When they enter their credentials, the method is called to check the Firebase for the account. If the account exists, the method returns true, displays a success message, and the user can go to the next screen. However, if the account doesn’t exist, the method returns false, displays an error message, and the user has to put in valid credentials.

User Story #7

As a user, I would like to reset my password, in the event I forget.

#	**Description**	**Estimated Time**	**Owner**
1	Create a *Forgot Password* button	1 hour	Shayne
2	Create an interface to enter the email address linked to the account	1 hour	Shayne
3	Generate a Reset Password Code	1 hour	Shayne
4	Send email to user with Reset Password Code	2 hours	Shayne
5	Method to validate the Reset Password Code in client.	1 hour	Shayne
6	Update password on Firebase	1 hour	Shayne
7	Create unit tests to verify that a password can be reset	3 hours	Kade

<u>Completed</u>

For users that forget their password, we created a separate ‘forgot password’ page that allows users to enter their registered email address and reset their password. Upon entering a valid email address, the user is sent an email with instructions and a password reset link. Once the user has accessed the link and reset their password, their credentials are updated on Firebase.

User Story #8

As a user, I would like to login with my Google account.

#	**Description**	**Estimated Time**	**Owner**
1	Set up Firebase to allow Google authentication	3 hours	Umang
2	Open Google Sign-in window	2 hours	Umang
3	If an account with VirtuCard linked to the Google Account doesn't exist, create a new account	4 hours	Umang
4	If an account with VirtuCard linked to the Google Account exists, display a confirmation message	2 hours	Umang
5	Display error message if sign-in with Google Account fails	1 hour	Umang

Completed

Google Authentication was implemented a lot slower than anticipated due to problems understanding how some of the libraries worked in Unity. After installing the Google Sign-In package from its Github repository, we were able to implement the functionality and hook up its respective script to its UI button. Testing its functionality for the demo took longer than anticipated as the mobile build had to be exported and run on a mobile device each time a change was made. Additionally, due to some issues with a merge request, the functionality had to be re-implemented into the Client again.

User Story #9

As a user, I would like to login with my Facebook account

#	**Description**	**Estimated Time**	**Owner**
1	Set up Firebase to allow Facebook authentication	3 hours	Umang
2	Open Facebook sign-in window	2 hours	Umang
3	If an account with VirtuCard linked to the Facebook Account doesn't exist, create a new account	4 hours	Umang
4	If an account with VirtuCard linked to the Facebook Account exists, display a confirmation message	2 hours	Umang
5	Display error message if sign-in with Facebook Account fails	1 hour	Umang

<u>Completed</u>

The process of setting up Facebook Authentication was the loop of changing some lines of code, building the app, finding what error arose, and repeating. After several trials, we were able to get Facebook Authentication reliably working. Due to security laws set up by Facebook, implementing the Facebook Sign-in feature is limited to using accounts generated by the Facebook Developer Portal, and not any arbitrary account.

User Story #10

As a user, I would like the option to play as a guest, not having to make an account.

#	**Description**	**Estimated Time**	**Owner**
1	Add a "*Play as Guest*" button	1 hour	Aryan
2	Deny access to the profile/statistics page unless signed in.	2 hours	Aryan
3	Generate a temporary random name for the user	1 hour	Aryan
4	Ensure the random name is unique in Photon	1 hour	Aryan
5	Discard name upon exiting the game	1 hour	Aryan
6	Add unit tests that check if a player can play as a guest, receive a random username and ensure name is unique	3 hours	Shayne

Completed

Setting up Firebase Anonymous Authentication was fairly straightforward, given that the methods have already been set up for Login and Registration at that time. This meant that the process of creating an Anonymous account was relatively uneventful. There was however, a need to update the design for the user information to include an identifier for anonymous accounts.

User Story #11

As a user hosting the game, I would like to be able to join the game room through my phone as a game player.

#	**Description**	**Estimated Time**	**Owner**
1	Show option while creating game to allow host account to join game	2 hours	Kade
2	If the host account attempts to join the game from an app, check if the user setting was enabled.	2 hours	Ryan
3	If setting is disabled, show error message while attempting to join	2 hours	June
4	Create unit tests to ensure a person can join as a player from their mobile device if they have already created a lobby from their laptop or desktop computer	3 hours	Aryan

Completed

We were able to set up the capability for the host to join the room. There is a checkbox that either allows or prevents the host from joining. When the host is barred from joining but attempts anyways, a popup displaying an error is shown.

User Story #12

As a user, I would like to view the actions of other players in real-time and without any delay.

#	**Description**	**Estimated Time**	**Owner**
1	Set up Photon to synchronize multiplayer user actions	5 hours	Shayne
2	Account for the delay while displaying information to the other players	4 hours	Shayne
3	Show status of a current, active player based if it's their turn to play	2 hours	June
4	Show status of passive waiting for their turn	1 hour	June
5	Design unit tests that evaluate the propagation delay between user actions and the appropriate response from the server	3 hours	June

Completed

While building a multiplayer game, it is important to ensure that all players are able to receive real-time updates about the current status of the game, their expected actions and the actions of other players. We decided to integrate this into our application in the form of a common card deck on the host application that is visible to all players and status banners that indicate whose turn it is to both the active and passive player.

User Story #13

As a user, I would like to draw cards from the table using my mobile device.

#	**Description**	**Estimated Time**	**Owner**
1	Once the game has started, have a button to allow the player to draw a number of cards from the table	2 hours	Ryan
2	Only enable button this if it is an allowed move and it is their turn.	2 hours	Ryan
3	Add the cards to the player's hand	2 hours	Ryan
4	Remove the cards from the deck	2 hours	Kade
5	Create unit tests that check that a player can draw cards successfully	4 hours	Kade

Completed

When the user is playing the game, there is a button that allows them to draw cards. It only appears if it is their turn. Otherwise, it is replaced by a panel displaying the name of the user whose turn it currently is. When the button is pressed, it removes a card from the undealt deck on the host side and adds it to the player's hand.

User Story #14

As a user, I would like to play the cards I have been dealt with on the table, from my mobile device.

#	**Description**	**Estimated Time**	**Owner**
1	Once the game has started, allow the user to select a card and play it	3 hours	Ryan
2	Update the interface to display the selected card by the user	2 hours	Shayne
3	Verify if the selected card is valid to play according to the game rules	2 hours	Kade
4	Remove card from a player's hand, add it to played cards on the common screen.	3 hours	June
5	Create unit tests that verify that a player can play valid cards during a game	4 hours	Ryan

Completed

When the user selects a card, it automatically communicates with the host to check if the card is valid to play. If the card is valid and played, it is removed from the player's hand and put into the played deck of cards on the host.

User Story #15

As a user, I would like to be able to pass my turn.

#	**Description**	**Estimated Time**	**Owner**
1	Add a button that allows the user to skip their turn	1 hours	Kade
2	Verify if the game rules allow skipping their turn	1 hour	Ryan
3	Pass the turn to the next player, notifying the player	2 hours	Ryan
4	Implement unit tests that check a player can skip their turn	2 hours	Kade

Completed

When it is the current player's turn, they can press the skip button, which allows them to pass the turn onto the next player instead of playing a card. It is only enabled if the game rules allow it, and it updates all the clients to show that the turn has been passed to the next player by displaying a panel saying whose turn it is.

User Story #16

As a user, I would like to be able to send public messages to the game lobby.

#	**Description**	**Estimated Time**	**Owner**
1	Set up a chat operation to send and receive messages using Photon while in the game room.	3 hours	Aryan
2	Create a message button and interface on the user's screen to send messages	2 hours	June
3	Allow message display to be hidden/unhidden with a button	2 hours	June
4	Create unit tests that verify messages can be sent from a player to all other players	4 hours	June

Completed

When the user enters a game with the chat enabled, the players have the chat visible on the host and the client-side. However, the client side has the option to send messages while the host chat is just meant for display. Whenever somebody sends a message, it will display on the client chat and the host chat. If the user decides that the chat is in the way of their gameplay, there is a button for them to press to hide the chat.

User Story #17

As a user, I would like to be able to view public messages on the common screen shared between players.

#	**Description**	**Estimated Time**	**Owner**
1	Create a public chat log that is shown on the common screen	2 hours	Kade
2	Update chat when messages come in.	1 hour	Kade
3	Display sent messages on other user's screens	2 hours	Ryan
4	If chosen by the host, the chat should not be displayed on the shared screen.	2 hours	Ryan
5	Design unit tests to verify that the public messages are displayed	2 hours	Kade

Completed

When a player sends a message from their mobile device, the message will show up on the common screen and the player's screen almost immediately. However, if the host decides the chat is too annoying, they can always disable the chat from the common screen.

User Story #18
As a user hosting the game, I would like to be able to disable the chat option, if it may be detrimental to the game experience

#	**Description**	**Estimated Time**	**Owner**
1	Add an option in the Create Game screen to turn off the chat feature for the entire game	2 hours	Shayne
2	Add a button to *Disable Chat* on the host's device during the Game	1 hour	Kade
3	Enable/Disable Chat as the button is toggled.	2 hours	Aryan
4	Ensure players are not able to access the chat interface once disabled.	3 hours	June
5	Implement unit tests to check that chat is either displayed or hidden depending on the game settings	2 hours	Ryan

<u>Completed</u>
We have done this task successfully because the host has the choice to enable or disable the chat before the game has started. When the chat is disabled, it sends the signal to the client-side and it disables the chat for everybody. The UI for the chat will disappear and everybody will be given a prompt that says "Chat is disabled."

What did not go so well in Sprint 1?

One part that was not ideal from Sprint 1 was that we were not able to finish all the test cases described in various user stories. Although plenty of manual testing was conducted to ensure that our project is operational for various edge cases, we have not been able to complete the process of writing test cases currently.

Another thing we did not do well was spread out our work efficiently. We often assigned many people to a single user story which would often cause delays in work. This was because we often had people waiting on others to implement features they easily could have just done themselves. This was tough to avoid since our team is relatively inexperienced with working on a group project of this scale and planning it that far ahead of time.

On top of that, we also had a hard time figuring out how long features would take to implement on our sprint planning document. Our work was distributed fairly evenly to start out, but there were a lot of things that took way longer to do, and things that took way shorter than expected. As we continue in this course and we get more familiar with Unity and our strengths, we should improve on our time-guessing abilities.

With this, we also had a lot of overlapping user story acceptance criteria. This was tough to see at first but as we continued on with the sprint we realized how similar our acceptance criteria was. This was not a terrible issue to have, but it did make some of our work difficult to measure progress in.

One major issue we ran into was a major malfunction with our version control. Seemingly overnight, a lot of people's work disappeared on the main branch and no one knew what was going on. This caused a significant delay in our work and a lot of stress, especially since this was two days before our sprint review. We eventually resolved the issue by creating a new branch to replace our main branch and then working from there with the work that we had on a recently updated branch.

Furthermore, there was a rush towards the last week to complete the user stories, which although in the end helped us check off all our user stories, we all agree that it was not at all ideal to perform development in such a manner.

Not Completed

User Story #2

5	Implement unit tests to debug creating Photon rooms with the unique code.	2 hours	Umang

Not Completed:
While the unit test for verifying a unique join code exists, there is no unit test to debug creating Photon Rooms. We have decided to make this a manual test as it is difficult to check if the Photon room was created using a unit test and it would be easier to do so with visual inspection.

User Story #3

6	Create unit tests for clients joining a game room.	2 hours	Shayne

Not Completed:
We still need to create that ensures that clients are successfully connected to the Photon room and the game created by the host after entering the unique join code displayed on the host application.

User Story #10

5	Discard name upon exiting the game	1 hour	Aryan
6	Add unit tests that check if a player can play as a guest, receive a random username and ensure name is unique	3 hours	Shayne

Not Completed:
After creating an anonymous user during sign in, the user's credentials are supposed to be destroyed in Firebase after they exit the game. This feature was implemented but needs to be tested further and certain bugs need to be fixed as well, since it currently fails to do so reliably. We also need to add a test that checks the guest/anonymous login feature and ensures that the users can create their own username and name.

User Story #11

4	Create unit tests to ensure a person can join as a player from their mobile device if they have already created a lobby from their laptop or desktop computer	3 hours	Aryan

Not Completed:

We have decided to make this a manual test as well because it is difficult to test the connectivity with a unit test and the mobile client. The manual test can use the player list user interface to verify if users are able to join the game lobby.

User Story #12

5	Design unit tests that evaluate the propagation delay between user actions and the appropriate response from the server	3 hours	June

Not Completed:

These tests need to be completed. It has been decided that it will be a manual test case because designing a unit test for propagation delay would be difficult. The test must verify that there is little delay between user actions and the host's response.

User Story #13

5	Create unit tests that check that a player can draw cards successfully	4 hours	Kade

Not Completed:

The functionality for drawing cards is finished, but the test has still yet to be written. It needs to verify that a player can draw cards successfully when it is their turn.

User Story #14

5	Create unit tests that verify that a player can play valid cards during a game	4 hours	Ryan

Not Completed:

The tests to verify that a player can play valid cards during their turn were not complete. These tests must verify that a player can choose a valid card and play it during their turn.

User Story #15

4	Implement unit tests that check a player can skip their turn	2 hours	Kade

Not Completed:
The tests for skipping player turns were not completed last sprint. The tests must verify that a player can skip their turn only if it is their turn and the game rules allow it.

User Story #16

4	Create unit tests that verify messages can be sent from a player to all other players	4 hours	June

Not Completed:
These unit tests were not completed. The tests will eventually have to verify that messages are sent to all players and can be seen on the common screen.

User Story #17

5	Design unit tests to verify that the public messages are displayed	2 hours	Kade

Not Completed:
We need to design and add unit tests that check if the public chat messages are being successfully displayed on the chat interface and can be viewed by all the players in the game.

How should we improve?

The first two weeks of the sprint went pretty well overall, but near the beginning of the third week, we realized we had given ourselves much more work than we had anticipated. Our evenly spread workload between the weeks turned into a crunch at the end of the sprint. This was mostly due to our misunderstanding of the time estimates in the sprint planning document, which led us to the majority of tasks. This resulted in a reduced quality of work, which led to the introduction of a couple bugs. Now that we have a better idea of how long tasks will take, we will better predict the amount of work for sprint 2. We also aim to handle the bugs we've encountered first, before moving on to sprint 2 tasks.

One huge issue we identified was that we had assigned individuals to tasks rather than vice versa. This led to hiccups in workflow because team members had to wait for others to complete their tasks before they could begin certain tasks themselves. To improve on this, we aim to assign entire user stories or related tasks to specific individuals such that progress can be smoother and made in a more modular fashion. This will prevent unnecessary dependencies and should ensure work will happen more consistently across the entirety of the sprint rather than a focused portion.

As Sprint 1 continued on, we started to get together more frequently for coding meetings, which proved beneficial. We weren't easily distracted and accomplished tasks more efficiently. The communication between how we were planning to implement certain features helped keep everyone up to speed as to what was happening on different parts. Additionally, it allowed other members to provide feedback if they thought they had a more optimal solution. We will definitely plan more meetings for next sprint to help gauge current progress and help members that are struggling with specific tasks. Furthermore, our experience towards the end of the sprint in "Integration Hell" taught us the need to use version control more carefully, choosing the correct files in the commits we make, to branch more often, and test merged branches before making pull requests.