Sprint 1 Retrospective

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

What went well?

General:

After this sprint, we have a good baseline codebase for us to add onto for the next two sprints. We focused on creating a skeleton with minimal functionality and were successful in that. Our front end is ready to have more pages/components added on and our backend, while a little primitive in functionality, can communicate with the database in terms of sending and receiving requested information. This should help remove overhead for structural tasks and focus more on creating the core of the game itself.

Also, everyone was able to learn new technologies and contribute relevant code in a timely fashion. Everyone in the group had to learn at least one new language/ framework/ platform relevant to the project and to do so quickly and have everyone's pieces come together the way they did is something to be proud of.

Another thing that went well this sprint was the team's communication. Being able to effectively communicate issues that arise is important to having a successful team especially when the team only meets a few times a week. Group members were consistently present and

punctual to group meetings, only missing for very outstanding circumstances which, again, was communicated effectively and early.

User Story Specific:

User Story #1

As a developer, I would like to develop the frontend with React and the server in Python with Flask

#	Description	Estimated Time	Owner
1	Set up Flask server and backend boilerplate	4 hrs	Tyler
2	Create create-react-app project	2 hrs	Blake Steel
3	Ensure UI Conformity between different pages of the game	3 hrs	Akash
4	Add Redux and state management skeleton	5 hrs	Tyler
5	Create basic client routing and navigation	5 hrs	Blake Steel
6	Write/set up basic testing framework and smoke tests on the client	5 hrs	Akash
7	Read ORM documentation and translate database tables to Python ORM classes	5 hrs	Tyler

The backend and frontend are both functional. The backend can interact with the database, and the frontend with the backend. The user can navigate around the frontend as well.

User Story #2

As a developer, I would like the application to be able to properly handle any raised errors.

#	Description	Estimated Time	Owner
1	Create JSON error handlers with Flask	3 hrs	Raziq
2	Create custom exceptions to raise like	4 hrs	Akash

	UnauthenticatedException or BadRequestException		
3	Create catch-all error handler for unexpected server errors	1 hr	Raziq

The backend catches exceptions ranging from bad request data to unauthorized accesses to unexpected server exceptions and serves these exceptions as JSON.

User Story #3

As a developer, I would like to store user and cryptocurrency data with a hosted PostgreSQL database.

#	Description	Estimated Time	Owner
1	Register for AWS	1 hr	Ryan
2	Create AWS RDS instance for PostgreSQL	1 hr	Ryan
3	Create migration scripts	4 hrs	Akash
4	Migrate the database	2 hr	Sam
5	Set up database connectivity in Flask	5 hrs	Ryan

We got Fortune up and running on a VPS and it can interact with a hosted PostgreSQL database from the backend.

User Story #5

As a player, I would like to be able to register for a Fortune account.

#	Description	Estimated Time	Owner
1	Create login page	6 hrs	Blake
2	Create registration page	6 hrs	Ryan
3	Create API endpoint for login	6 hrs	Sam

4	Create API endpoint for registration	6 hrs	Sam

A user can visit Fortune and can register for an account as well as log into an account. The backend will verify the user's credentials and/or store their credentials in the database.

User Story #8

As a player, I would like a choice of game title.

#	Description	Estimated Time	Owner
1	Add text box to create game page with label to name private game	2hrs	Blake
2	Write tests	2 hrs	Blake
3	Check for invalid input and alert the user when invalid input is entered	1 hr	Blake

The user can enter text into a text box to name the game. We check that the text must be within a certain length.

User Story #9

As a player, I would like a choice of duration of the game.

#	Description	Estimated Time	Owner
1	Add calendar to create game page to select end time of game	3 hrs	Raziq
2	Display duration of selected end time	1 hrs	Raziq
3	Send data on save to game endpoint to update	2 hrs	Raziq
4	Write tests	5 hrs	Raziq

When creating a game a user can enter when the game should end with a calendar widget; it disallows invalid input (dates before now since the game must end in the future).

User Story #10

As a player, I would like a choice of which cryptocurrencies (BTC, ETH, etc) are to be traded during the game.

#	Description	Estimated Time	Owner
	Create dropdown list of coins on create game page to select from	2 hrs	Raziq
	Send data on save to game endpoint to update	2 hrs	Raziq
	Write tests to verify the dropdown shows each option and can select the option the user wants to choose	5 hrs	Raziq

A user is able to select from a list of coins when creating a game.

User Story #12

As a developer, I would like to deploy Fortune through Amazon Web Services in order to integrate our frontend and backend easily.

#	Description	Estimated Time	Owner
1	Deploy EC2 instance for production purposes	3 hrs	Ryan
2	Install and configure web and application hosting software such as nginx on production EC2 instance	3 hrs	Ryan
3	Use AWS Directory Service to join EC2 Instance and RDS database through the same domain.	2 hrs	Ryan
4	Point github.io domain of Github repository to EC2 instance for ease of connection	2 hrs	Ryan

We have a production server up and running on an EC2 instance along with an accompanying RDS Postgres database. This has also been accomplished through Amazon Virtual Private Cloud, to allow for security as well as a common domain. We were unable to point the EC2 server to a github.io domain, but instead have it pointing to an alternative URL, fortune-game.duckdns.org.

What did not go well?

General:

The division of tasks created a few issues throughout the sprint.

- Tasks were not divided up appropriately, with some people's tasks heavily dependent on other tasks given to another person creating blockages while some people were waiting on others to finish. One story often had multiple developers with different tasks in it and if one developer wasn't finished with their task the user story didn't see much progress.
- We did not divide tasks up by experience or knowledge, leaving some people bored or confused on what to do. This was mostly due to the circumstances of how quickly the deadline for the sprint 1 planning document came right after the design document was due and we had to finish the planning document remotely. Some members weren't able to contribute to the selection of stories and tasks that they were knowledgeable in or wanted to do and instead were assigned various tasks and stories originally thought up by others.
- Likewise, we weren't able to meet every single acceptance criteria for many of the user stories which ended up impacting our sprint review grade.

User Story Specific:

User Story #4

As a developer, I would like to integrate historical cryptocurrency prices into the platform.

#	Description	Estimated Time	Owner
1	Obtain historical data files	1 hr	Tyler
2	Parse historical data files	1 hr	Tyler

3	Write script that inserts data into database	4 hrs	Tyler

We are still waiting to hear back from Nomics (our cryptocurrency API provider) for a price quote, so we were unable to complete this story. Once we have access to the API we should be able to complete this.

User Story #6

As a developer, I would like to only allow authenticated users to access some routes in Fortune.

#	Description	Estimated Time	Owner
1	Create Python authentication decorator	4 hrs	Tyler
2	Persist user's token in localStorage on browser	2 hrs	Sam
3	Write tests to ensure the authentication decorator only allows valid users with valid tokens	4 hrs	Tyler

We were able to create the authentication decorators, but due to time constraints we were not able to implement them into the actual functionality of the game. The tokens were not persisting in local storage on the browser for unknown reasons and needs further debugging.

User Story #7

As a player, I would like to create a private group game through the play page.

#	Description	Estimated Time	Owner
1	Create "Create New Game" page	9 hrs	Blake
2	Create API endpoint for creating a new group game	4 hrs	Sam
3	Create API endpoint for updating a game	4 hrs	Sam
4	Write tests	5 hrs	Sam

Users can create a game through the UI that connects with the backend, however there was minimal bounds checking on the input and we did not get to updating a game.

User Story #11

As a player, I would like a choice of initial cash amount each player has.

#	Description	Estimated Time	Owner
1	Create text box on create game page with label to specify initial starting cash each player starts with	2 hrs	Akash
2	Write tests to verify the text box is visible and can update when the user types text into it	2 hrs	Akash
3	Check for invalid input and alert the user when invalid input is entered	1 hr	Akash

When the user creates a game they can specify the starting cash, however we did not check for invalid input when the user tries to create a game. This led to behavior that contradicted what we wrote in our acceptance criteria plus does not make sense for a game.

How should we improve?

For this next sprint, we're planning on focusing on dividing user stories and tasks evenly and fairly among the group so that there are minimal blockages between developers with related tasks and so that everyone gets tasks and stories that they have experience with and/or are interested in completing. This should go relatively smoothly this time around as we are planning to meet up to create the sprint 2 planning document rather than trying to complete it remotely. We can freely discuss what we each plan on accomplishing during the sprint and have everyone's opinion heard and added to the sprint planning document. In terms of preventing blockages between developers, it would be beneficial to make sure user stories are given to a sole developer or two developers working closely together with similar plans and goals for the week.

We were also not aware of how important it would be to follow the sprint planning document in terms of how we were going to be graded by the end of the sprint. For this next sprint we need to make sure we are following our acceptance criteria closely and modify our planning document if we run into any roadblocks before our sprint review. An easy way to prevent having to make many modifications during the sprint is to take the time in the beginning of the sprint to 1: write appropriate and realistic acceptance criteria for each story, and 2: consider the acceptance criteria before proceeding with implementation of the given task or user story.