CRYPTOLEAGUE

Team 8: Sprint 1 Retrospective

Varun Gupta, Ritwik Bhardwaj, Utkarsh Jain, Nisarg Kolhe

February 25, 2018

WHAT WENT WELL?

General

- The work was very well organized
    - We split up the backend and the frontend into different folders, with the frontend running on Angular and the backend on Node.js. All of us split the work based on different features of the app. So, only one person was working on a particular feature on either the frontend or the backend which made it really easy to maintain the project. Also, the uniform consistency in naming our components and API routes added to uniform organization of the project.
- Very robust implementation of components

    - Our app is very robust as of now, all thanks to the unit tests which we have for every front-end component. We are also manually testing our backend API routes by making mock requests through Postman.

- Good communication between teammates

    - We have a really good communication with each other and all of us think that it is the main reason behind this successful first sprint. All of us worked on separate branches for our features and reviewed each other's code before merging it to master.

Tasks

The tasks are listed in the following format:

User Stories

- Tasks
    - Comments

1. As a developer, I would like to make the UI seamless and easy to use.

    - Implement global responsive styling
        - Global responsive styling was implemented with UIKit on Angular using SASS.

    - Implement Dashboard UI
        - The dashboard page was replicated accurately from the mockups we had with UIKit on Angular using SASS and was also made responsive for all devices/screens. Since this was the first view implemented, it ended up being a good base for future UIs.

    - Implement Collapsing Sidebar

- Took a few iterations using UIKit's overlay styles but we ended up implementing it with custom styles making it quite robust and responsive.

- Test UI on various screens
  - Testing was done with Chrome Developer Tools for various screens including PCs, laptops, phones and tablets.

2. As a user, I would like to be able to register for a CryptoLeague account with social platforms like Facebook or Google.

   - Set up back-end server
     - Code scaffolding and routing setup was done elegantly with Express generator.

   - Set up front-end client
     - Code scaffolding and routing setup was done elegantly with Angular CLI.

   - Set up MongoDB database
     - Local MongoDB database was setup and shared starting with a Users collection.

   - Implement the login/signup UI
     - The login page was replicated accurately from the mockups we had with UIKit on Angular using SASS and was also made responsive for all devices / screens.

   - Implement the login server API using passport.js
     - Passport.js was setup successfully with social accounts (Facebook and Google) with clear API endpoints providing JWT authentication.

   - Implement JWT authentication on the back-end
     - JWT authentication was done using jsonwebtoken NPM package and a custom passport.js recipe for JWT authentication.

- Implement JWT authentication on the back-end
  - The JWT token was decoded with angular2-jwt and the acquired user object and the token was saved into localStorage.
- Implement landing route for user to setup their profile
  - A separate route was setup for user to update their username on first login.
- Manually test the API route
  - The testing for the API was done using Postman with various inputs and all the testcases passed on the frontend's testing framework.

3. As a user, I would like to be able to manage my CryptoLeague account.
   - Implement the settings UI and make it responsive for all different screens/devices
     - The settings UI was implemented successfully with the UIKit on Angular using SASS and was also made responsive for all devices/screens.
   - Implement the server API to update the user data
     - Routes for updating the user object were implemented successfully in the backend and the information was also updated successfully in the mongodb database.
   - Manually test the API route
     - The testing for the API was done using Postman with various inputs and all the testcases passed on the frontend's testing framework.
4. As a user, I would like to be able to watch real-time market prices of cryptocurrencies
   - Implement the market UI and make it responsive
     - The market UI was implemented successfully with the UIKit on Angular using SASS and was also made responsive for all devices/screens.

- Display the data
  - The user can successfully view the coin data for the top 100 coins in the market in form of a table on the frontend.
- Implement server API to retrieve the data from the coinmarketcap
  - All of the coin data was successfully implemented using coinmarketcap. Chasing-coin API was also used to get more information about coin which will help the user's evaluate the coins more accurately.
  - The coin data was cached in the server to serve them in the backend API for the frontend.
- Manually test the API route
  - The testing for the API was done using Postman with various inputs and all the testcases passed on the frontend's testing framework.

5. As a user, I would like to be able to see current cryptocurrency related news.
   - Get data from newapi.org API
     - Newsapi.org integration was successfully implemented to get latest news regarding cryptocurrencies and cached in the server to serve them in the backend API for the frontend.
   - Implement the server API for getting the news
     - Successfully implemented a route for the frontend to access the JSON object made to get the news and its display picture, link and a short description for the frontend to display the picture.
   - Implement displaying news received from the server API
     - The dashboard page has a card which successfully displays top 15 news received from the backend API and refreshes it every 15 mins.
   - Manually test the API route

- The testing for the API was done using Postman with various inputs and all the testcases passed on the frontend's testing framework.

6. As a user, I would like to be able to see current leaderboard of top performing players.

    - Implement the UI of global leaderboard
        - The global leaderboard page was replicated accurately from the mockups we had with UIKit on Angular using SASS and was also made responsive for all devices/screens.

    - Implement the server API to get all users' data
        - Successfully implemented a route for the frontend to get all the user's public data in ascending order of their ranks according to their tokens.

    - Display the data of the users in the leaderboard UI
        - The leaderboard page successfully displays all the top users received from the backend API with proper pagination implemented.

    - Manually test the API route
        - The testing for the API was done using Postman with various inputs and all the testcases passed on the frontend's testing framework.

WHAT DID NOT GO WELL?

General

- Underestimating hours for some of the user stories
    - Unfamiliarity with the frameworks resulted in underestimating the amount of time it took us to implement some tasks. For example, it took us a lot of time figuring out how to setup passport.js social login with JWT authentication. Therefore, we had to work longer hours than what we expected and thus we were not able to work as smoothly as we wanted to.
- Lack of documentation for the backend routes
    - We did not have any documentation for the routes and API's which we created and used. Thus, this resulted in a lot of miscommunication with the different team members working on the frontend and backend. Lack of documentation also made it hard for us to look back at the routes and determine the functionality of it and made testing a lot harder when something went wrong.
- Did not equally distribute tasks for all the weeks
    - We had a lot of tasks piled up in the last week of the sprint as we did some light user stories in the first couple weeks of the sprint and also due to underestimating the amount of hours required for each story. Thus, we had to put in a lot of extra effort in the final week of the sprint. Also, since the functionality of updating the user was implemented in the backend later in the sprint, there was significant time difference in just designing the UI and making it work with the backend.

NOTE: We did not have any unsuccessful user stories.

HOW SHOULD WE IMPROVE?

General

- Try to equally split tasks between the three weeks in the sprint
    - Like mentioned above, due to the unfamiliarity with the frameworks, we were not successfully able to assign each task an accurate due date which resulted in us having to do a lot of tasks due in the final week of the sprint. We are going to implement the backend functionalities first, and then move on to the frontend, so that the backend API routes are not a bottleneck for the frontend tasks. This way we will be able to work more efficiently, faster, and effectively. Therefore, we are going to try finish up the more important backend tasks in the first week so that we don't have to wait to implement the frontend for those features until the last moment.
- More accurately estimate hours required for each task
    - We definitely underestimated the time required for some of our tasks. We thought that we would find enough documentation for some of the APIs we called, and would be able to implement them quickly. This was seen in examples as mentioned above about the passport.js and the newsapi.org problem. Now that we have more experience with the frameworks, everyone is familiar with the codebase, and how long it took to implement different kind of tasks, we would be accurately able to estimate hours for the future tasks by using our past statistics as a reference and also by being able to properly communicate with our project coordinator.
- Have some documentation for the routes and API calls
    - Due to the lack of time, we didn't spend time on making any documentation for the APIs and the routes we had implemented. In the future, we will be better at time management and will make sure that we have adequate documentation

for the APIs and routes we make so we do not have any problems with recalling the functionality of each API and route and are able to test the app in a more efficient and effective way.