Testing

Instructions:

- Frontend
  - Npm test -- run all the tests that are built in frontend
- Backend
  - Npm run test::integration -- run integration test
  - Npm run test::unit -- run all Unit tests with coverage displayed
  - npm run test::singleUnit -- run the unit test of your choice
  - Note: npm run test::unit will run all the unit test files. To run specific file with file name of "fileName.spec.ts", simply change the testMatch option in /backend/jest.singleUnit.json to "**/?(fileName.)+(spec).ts?(x)".
- Locations
  - Frontend tests are spreaded inside the frontend components folders, for example, for the landing page component, the test is /frontend/src/pages/Landing/Landing.test.tsx.
  - Backend tests can be all found inside in /backend/test

Continuous Integration (Automated building and testing):

We have set up a github action to run the integration tests whenever new changes are about to be pushed or merged to the master branch. The script can be accessible in /frontend/.github/workflows/build.yml. What the script does is to try to run the backend integration test, run all the backend unit tests, check if the backend can be built properly, run frontend integration test, and check if the frontend can be built properly. It helped us to rule out a lot of small mistakes that are made by the team and prevent a lot of unnecessary confusions.

FrontEnd:

Page:

- Landing:
  - Check if the Landing page contain section “home” and “search”
  - Check if the Landing page has title “Game Performance Tracker”
  - Check if the Landing page has text “Track Performance”, “Analyze Matches”, “Leaderboarding”, “Evaluate Champions”.
- Leaderboard:
  - Check if the Landing page has text “Top”.
  - Check if the UserNotFound page contain all the components:

    - including “leaderboard heading”, “leaderboard content”, “leaderboard selectors”.
- Match History:
- Overview:
    - Check if the UserNotFound page contain all the components:
    - including “overview”, “user-profile”, “circle-charts”, “user-champs”, “match-chart”.
- Search:
    - Check if the Search page contain all the components:
      including “search-bar”.
    - Check if the Landing page has title “Game Performance Tracker”
- User Not Found:
    - Check if the UserNotFound page contain all the components:
    - including “topSection”, “SummonerNotFound”, “ExtraLink”

BackEnd:

Controllers:

- summonerController:
    - Check if the controller is properly managing queries gathered from request and handle the data fetching from database and Riot API properly, and return the proper user and leaderboard data. And make sure the status codes are correct for different requests.
- matchController:
    - Check if the controller is properly managing queries gathered from request and handle the data fetching from database and Riot API properly, and return the proper match history and chart data. And make sure the status codes are correct for different requests.

Services:

- matchService:
    - Check if each function in the service is functional or not by setting tests with different inputs, then check against the expected output to make sure no errors are happening. Created mock match data input for some of the tests so that it can simulate the scenario in which they are used.
- summonerService:
    - Check if each function in the service is functional or not by setting simple tests to see if there is any obvious error, also set some mock data to test if the functions are behaving correctly as expected.

IntegrationTest:

Make sure the server is fully functional and can react to the sequence of requests that are made by the actions of users. It covers most of the basic behaviors that are anticipated to happen during normal activities.