LineUp Test Manual

Testing plan :

The test plan is to perform unit tests on the backend and write test cases for testing the front end. We would manually perform the UI test cases on the application. We would use hoppscotch.io to test the requests to the APIs.

Testing strategy :

Yitao, Tong, and Jane would test the APIs using Hoppscotch.io. We would write the cases in Hoppscotch and export them to hoppscotch-collection.json. This way another person could import that json file into https://Hoppscotch.io, run each request manually and see the response status. Tong and Jane would test the frontend by using the application. The test objectives were to test the essential features the Teacher and Student users would need to use and check that they work as expected. We wanted to ensure that the application can properly handle the cases of invalid input and actions with pop up error messages. The manual tests for the backend would ensure that each part of the API works as expected so we would be able to reduce the rigor of the manual frontend tests.

Testing scenario :

Unit test:

- Auth: The user registration, logging in/out request can be correctly operated, and the invalid input should be well handled
    - Action
        - userLogin()
            - Correctly process the valid username/password
            - Correctly handle invalid username/password input
            - Correctly redirect to dashboard if successfully logged in
        - userJoin()
            - Correctly post the username/password input for registration
            - Correctly handle the user registration failure
        - userSignOut()

                - Correctly commit the signing out info
                - Correctly return to the homepage after signing out
            - reset()
                - Correctly reset (remove the existing) login info
                - Triggered when the login is expired
    - Getter
        - isAuthenticated()
            - Correctly return the authentication status if both of the username password are not null
        - getUsername()
            - Correctly return the username
        - getPassword()
            - Correctly return the user password
        - getDisplayName()
            - Correctly return the display name if not null
        - getStatus()
            - Correctly return the user status if not null
    - Mutations
        - setUsername()
            - Correctly set the username to the given state
        - setPassword()
            - Correctly set the password to the given state
- Event: The event represents the course item or something equivalent, which contains multiple queues
    - Action
        - createEvent()
            - Correctly generate a new event by the given event name and queue
        - updateEvent()
            - Correctly set the eventId, name and token to the given ones
        - joinEvent()
            - Correctly return the return value to the user who joined in the event
        - userEvent()
            - Correctly fetch the events some user getting involved in
    - Getter
        - getEvents()
            - Correctly return all the events of the current state
    - Mutations
        - updateEventId()
            - Correctly assign the given eventId to the given state
        - updateEvent()

                - Correctly assign the given event to the event given state
            - updateUserEvent()
                - Correctly assign the given payload to the events of the given state
- Local: Handle the website status on user's terminal device
    - Mutations
        - updateLoading()
            - Correctly update the given state to the given status
        - updateError()
            - Correctly assign the given error status to the given state
- Record: Represents the minimal element in the queue
    - Action
        - createRecord()
            - Correctly create a new record to the given queue
        - getQueues()
            - Correctly return the current queue with mathed users
        - getQueueUser()
            - Correctly return the user by the given id
        - leaveLine()
            - Correctly find the user by its id stored in the map
            - Correctly set the status of the given user to end
    - Getter
        - getReloadRecord()
            - Correctly return the reloadRecord of the given state
    - Mutations
        - reloadRecord()
            - Correctly flip the status of the reloadRecord of the given state
- User: Represents the user in the queue. One user can get registered in multiple events.
    - Action
        - getUser()
            - Correctly update the user info before fetching
            - Correctly return the data of the current local user
        - updateUser()
            - Correctly update the displayName and status of the current user
    - Getter
        - reloadRecord()
            - Correctly update the user info before fetching
            - Correctly return the data of the current local user
        - getStatus()

                - Correctly return the current user's status if exists
                - If status has not be initialized, return an empty string
            - getName()
                - Correctly return the username if exists
                - If username has not be initialized, return the display name
    - Mutations
        - updateUser()
            - Correctly assign given payload to the user in the current state
- Mixins: Coordinate the form submission
    - disableSubmit()
        - Correctly check if the form submission is disable
        - Return true if the form is not valid or has been submit
    - isFormDirty()
        - Correctly check if the form is dirty
        - Return False only when every key in the veeFields are not dirty
    - isFormValid()
        - Correctly check if the form is valid
        - Return true only when there is no error and the form is not dirty
    - validateState()
        - Correctly check if the given state is valid when exists
        - When the corresponding value exists and is valid, check if there is any error. Return true only when there is no error.
    - invalidResponse()
        - Correctly check if the most recent response is invalid
- Router
    - Router() construtor
    - beforeEach()
        - Correctly check authentication before move from one router to another

Manual test:

- UI Testing
    - Login/ Registration Page
        - Signing in with an existing username and correct password
            - Correctly allow the user to proceed to the dashboard.
        - Creating an account with an unused username
            - Correctly allow the user to create the account. Create the user in the User table of the database.
        - Creating an account with a used username

            - Correctly disallow the user to create the account and tell the user that the username is in use. Do not add the user to the User table.
        - Signing in with an existing username and incorrect password
            - Correctly disallow the user to proceed to dashboard
    - DashBoard
        - Create an Event with a valid name and no queues
            - Correctly create the event in the Event table and create the queue “General”.
        - Create an Event with a valid name and queues
            - Correctly create the event in the Event table and create the queues requested.
        - Join an event with a valid Event code and the user is currently not part of the event.
            - Correctly let the user join the event. Add the user to the Event User table.
        - Join an event with a valid Event code and the user is currently in the Event
            - Correctly let the user know they are already in the event and don’t let them join the event again.
        - Join an event with a invalid Event code
            - Correctly let the user know that they cannot join the event since it doesn’t exist.
    - Record
        - Adding a new request to a queue after joining an Event
            - Correctly add their request to the queue user table that was selected and add their message if they provided one.
        - Open “Show all requests” panel
            - Correctly show the current request the user has and the previous ones they’ve requested.
            - The user has put in a request and is currently waiting
                - Correctly show the progress bar is blue and the status is WAITING
            - The user has put in a request and is currently getting helped
                - Correctly show the progress bar is green and status is IN_PROGRESS
            - The user has put in a request and is done getting helped and leaves the queue
                - Correctly show the status is LEFT and the past record updates with the request.
            - The user has put in a request and is done getting helped and is removed from the queue

                - Correctly show the status is FINISHED and the past record updates with the request.
    - Manage
        - Event option
            - Event info should list the current queues and assistants per queue.
                - Correctly show the events that the user is an admin of and the assistants the event has
            - Queue option
                - Correctly show the queues that the user is an admin of and the current requests per queue.
    - Navigate
        - Display name option
            - Display name correctly changes for the user and updates accordingly in the event queue and past requests.