STAKEHOLDER REVIEW #2

Lake Mountain Leprechauns

TESTING DOCUMENTS

General Strategy

Our general strategy will be based on initial manual tests followed by encoding these tests into automated cases. During the initial testing, the tester will document every interaction taken with the system with the intention of checking for functionality or lack thereof. These documented interactions will include any unexpected anomalies. The important note to make in initial testing is that we do not intend to have the tester follow only our documented test cases and any additional features to test that the tester comes up with will be documented as well to be encoded into an automated test case. By automating each manual interaction with the system, we will eventually construct a testing suite that will catch any regression bugs that arise out of future changes and in the event that the project is inherited by another developer unfamiliar with any potholes, they will have a comprehensive collection of aspects to be tested in the system with minimal time invested on their part.

Quality Goals

Our major expectation of our product is robustness. The web application, server, nor the mobile application should crash or exit unexpectedly outside of hardware complications that would be infeasible to account for. The second priority for quality we hold is to ensure that the expected input provides the expected output with no unexpected side effects. We should confirm that our product behaves as detailed in our elaboration.

Resources

We expect the extent of our resources to amount to two people at most as well as a server to host a the web application, a mobile device to run the mobile front end, and a means to connect the two on the same network (such as a cable between the two and port forwarding the server to the mobile device). We are using the Rails built in testing framework for the web application and JUnit to test the mobile front end.

Schedule

We will also require the time necessary to test the test cases and construct the automated tests that will follow. The testing will take place in first week of the transition phase (11/23 to 11/30) in order to leave time for any modifications to be applied as a result of failed tests. We will then spend our last three days of the transition (12/7 to 12/9) testing to ensure that the final release is stable and up to our standards.

Test Cases

TC1: Create a user account.

- Prerequisites
    - The user must be on the homepage not logged in.
    - A user with credentials name “test1”, password “password”, and email “test@fake.com” must exist in the database and no other user with the name “test2” or the email “test@real.com” exists.
- Process
    - Click the “Sign up” button and ensure the user is directed to the signup page.
    - Attempt to sign up thrice, once with the name empty, another with the email empty, and once with the password and password confirmation empty.
        - Each attempt should reject citing the appropriate reason.
    - Attempt to sign up with the identical credentials to the “test1” user, clicking submit after entering the information.
        - The program will reject the entry, citing a user with that name exists.
    - Attempt to sign up with the same credentials, but with the name as “test2”.
        - The program will reject, citing a duplicate email address.
    - Attempt again, with “test2” as the name, “real.com” as the email.
        - Rejection, citing that the email is not valid.
    - Attempt with “test2”, “test@real.com”, “pass” as password and “pa$$” as the confirmation.
        - Reject citing the the passwords do not match.
    - Attempt again with “test2”, “test@real.com”, “pass” as password and password confirmation.
        - The program will accept and redirect the user to the home page.
        - The user with the entered credentials exists in the database.

TC2: Log in with an existing user.

- Prerequisites
    - The user must be on the homepage not logged in.
    - A user with credentials name “test1”, password “password”, and email “test@fake.com” must exist in the database and no other user with the name “test2” exists.
- Process
    - Click the “Log In” button and ensure the user is directed to the login page.
    - Try to log in with credentials “test2” and “password”.
        - The program will reject citing an incorrect combination.
    - Try with “test1” and “pa$$word”.
        - Reject for the same reason.
    - Try again with “test1” and “password”
        - The program will accept and redirect the user to their home page.

TC3: Create a convention.

- Prerequisites
    - The user is signed into an account and at the home page.
    - There exists a convention called “Test Convention1” in the database.
- Process
    - Click the “Create Convention” button and ensure the user is directed to the create convention page.
    - Try to create a convention multiple times, once for each field available leaving it blank.
        - The program should reject for any empty field.
    - Try to create a convention with the name “Test Convention1” and valid input for the rest of the fields.
        - Reject citing that a convention with that name exists.
    - Try to create a convention with a different name and the start date chronologically after the end date.
        - Reject citing the start date follows the end date.
    - Create a convention with valid credentials
        - The program accepts it and redirects the user to the convention’s home page.

TC4: Edit convention.

- Prerequisites
    - User is logged in at their convention’s homepage (implying that the convention exists).
    - A convention named “Test Convention1” exists (the one being edited) and another named “Test Convention3” exists.
- Process
    - Click the “Edit Convention” button and ensure the user is directed to the edit convention page.
    - Try to submit a number of times equal to the number of fields, each time leaving a different field blank.
        - Rejection due to the appropriate field being blank.
    - Try to change the name to “Test Convention3”.
        - Rejection citing that a convention with that name exists.
    - Try to change the start date to be after the end date.
        - Reject based on start being later than end.
    - Try to change the end date to be before the start date.
        - Reject based on end being earlier than start.
    - Change name to “Test Convention2”
        - Accept, redirect user to the convention details page with the appropriate changes in place.

TC5: Add events.

- Prerequisites
    - User is on the view events page and is logged in.
    - There exists an event named “Event1”.
- Process
    - Click the “Add Event” button and ensure the user is directed to the add event page.
    - Try and submit once for each field being blank. Reject on empty field.
    - Try to submit with name “Event1”. Reject on a duplicate event name.
    - Try to submit with name “Event2” and valid fields for the rest.
        - Accept and redirect to the view events page, ensuring the event has been added.

TC6: Edit events.

- Prerequisites
  - User is on the view events page and is logged in.
  - There exists an event named “Event1” (the one being edited) and another named “Event3”.
- Process
  - Click the “Edit Event” button for “Event1” and ensure the user is directed to the edit event page.
  - Try and submit once for each field being blank. Reject on empty field.
  - Try to submit with name “Event3”. Reject on a duplicate event name.
  - Try to submit with name “Event2” and valid fields for the rest.
    - Accept and redirect to the view events page, ensuring the event has been modified appropriately.

TC7: Remove events.

- Prerequisites
  - User is on the view events page and is logged in.
  - There exists an event named “Event1”.
- Process
  - User clicks the “Delete Event” button. Ensure page refreshes and the event is properly removed.

TC8: Add and remove hosts and rooms.

- Prerequisites
  - User is on the view convention details page and is logged in.
- Process
  - Enter “Host1” into the “Add Host” form. Ensure it is added correctly.
  - Try to enter “Host1” again. There should be no change as it is a duplicate host.
  - Click the remove button and ensure the host is removed.
  - Do the same steps for the rooms.

TC9: Add documents.

- Prerequisites
  - User is logged in and on the convention home page.
  - User has access to two test images.
- Process

    - Click the “View Documents” button and ensure the user is directed to the view documents page.
    - Click “Upload” and the program will reject because no file is chose.
    - Click “Choose File” and navigate to the file location. Click submit.
        - The page should view a file with the name “<no name>”. Click the link and ensure the file is the intended one.
    - Select another image with “Choose File”. Enter “Image” into the display name field.
        - The page should add “Image” to the document list. Ensure it is the intended image.
        -
        -

TC10: Remove documents.

- Prerequisites
    - User is logged in and on the view documents page.
    - User has uploaded an image.
- Process
    - Click the “Delete” button next to the intended document
        - Ensure the document is removed from the list, database, and the server.

TC11: Scheduler creates a schedule with no conflicts.

- Prerequisites
    - User is logged in and at the convention home page.
    - User has a convention with five events “Event1” to “Event5” with no overlap and one room.
- Process
    - Select the “Schedule” page. Ensure the user is directed to the correct page.
    - Click the “Generate Schedule” button. Ensure the page shows a schedule of all five events in their preferred time.

TC12: Scheduler handles situations with inevitable conflicts.

- Prerequisites
    - User is logged in and at the schedule page.
    - User has a convention with five events “Event1” to “Event5” with 2, 3, and 4 overlapping each other (but not completely intersecting) and two room.
- Process

    - Click the “Generate Schedule” button. The page should display the following:
        - A schedule that omits “Event2” and places 3 and 4 in different rooms.
        - A schedule that omits “Event3” and places 2 and 4 in different rooms.
        - A schedule that omits “Event4” and places 2 and 3 in different rooms.
        - A notification that a perfect schedule cannot be generated and that the user must manually resolve the conflicts.

TC13: Mobile application can search conventions.

- Prerequisites
    - Mobile application can connect to the server.
    - User is on the mobile application’s search page.
    - The server contains conventions called “Test A”, “Test B”, “Test AB”, and “Test O”
- Process
    - Enter “A” in the search bar. Ensure the application only displays stubs for “Test A” and “Test AB”.
    - Enter “AB” in the search bar. Ensure the application only displays a stub for “Test AB”.
    - Enter “C” in the search bar. Ensure the application states that no conventions match the search results.

TC14: Mobile application can download convention details.

- Prerequisites
    - Mobile application can connect to the server.
    - User is looking at non-empty search results with “Test A” among them.
- Process
    - User selects “Test A” from the convention. Ensure the application displays the details and a “Download” button.
    - Select the “Download” button. The application will now view the convention’s main page.

TC15: Mobile application can view the assets of a downloaded convention.

- Prerequisites
    - User is viewing the convention’s main page.
    - The user has downloaded a convention with one asset (room, host, event, document) each named what the asset is (Like “Room” for a room).

- Process
  - User selects “Events”. Ensure there is a single event called “Event”.
  - Go back and select “Rooms”. Ensure the only room is a room called “Room”.
  - Go back and select “Hosts”. Ensure the only host is a host called “Host”.
  - Go back and select “Documents”. Ensure there is a single document called “Document”.

TC16: Mobile application can add events to a personal schedule.

- Prerequisites
  - User has a convention downloaded with two events called “Event1” and “Event2” occurring at the same time.
  - User is viewing the schedule.
- Process
  - Click on “Event1”. Ensure it correctly shows the event’s information.
  - Click the “Add to personal schedule” button. The button should grey out.
  - Navigate to the “Personal Schedule” list. Ensure that “Event1” is on the list
  - Navigate back to the event list and perform the same operation for “Event2”.
    - Ensure that on the “Personal Schedule” list that both events are highlighted red to indicate their conflict.

TC17: Mobile application can remove events in a personal schedule.

- Prerequisites
  - User has added two events to his personal schedule, “Event1” and “Event2”
- Process
  - Select the “Remove from personal schedule” button next to “Event1”. Ensure that “Event2” is still on the schedule and “Event1” is removed (but still in the event list).

CODE REVIEW

For our code review, we reviewed the Sturmanators. Their project is a Google Chrome extension that keeps track of how much browser time users are spending on productive vs. unproductive websites. Their design and UI are simple and effective. They are using Javascript for the project. Javascript 5, a non-object-oriented language, is required for the extension, but they code in Javascript 6 with its object-oriented capabilities and use an automatic converter to translate it to Javascript 5. In the code for the UI, various elements were grouped as classes, such as the domain name, template, and browser history. The code we were shown appeared to be well organized and commented. Unfortunately, two of their group members were missing, so we weren't able to see all of their code or hear about every aspect of their project. It would have been good for them to plan for the members' absences so as to give a more effective presentation.

We were reviewed by Team Backlog. Most of their comments pertained to the lack of connectivity between the different elements we presented, but we hadn't planned to have these elements connected for the Stakeholder 1 Review. They said we should make sure our naming conventions were consistent between the different parts of the project, which we made sure to do over the next week when we were connecting the pieces. They also pointed out that one of our team members didn't have any comments on his code. We amended this in time for the Stakeholder 2 Review. One helpful suggestion they gave was to include a date widget in the Event time selection part of our web app. We implemented this, and it worked well. They also suggested using the Github issue tracking system and a program called Lint that checks for style consistency.

CONTRIBUTION SUMMARY

Michael Mortimer developed the web interface. Rachel King developed the mobile interface. Kyle Samson provided database support to link the web components to the mobile components, and wrote the testing documents. Maggie Borkowski added features to the scheduler and wrote the code review, contribution summary, and status report.

STATUS REPORT

Since the Stakeholder 1 Review, we have successfully integrated the different components of our project. The scheduler is accessible through the web interface, and can run with the events that have been added to a particular convention on the web app. The mobile app can pull information from the database that the web app uses, connecting the front-end and back-end. On both the front-end and back-end, users can search for and add conventions. The back-end UI has been refactored to allow for ease of use, including the addition of saving multiple versions of schedules, and user account creation. We have ensured that all of our code is thoroughly commented, and we've written testing documents and a summary of our code review.

The front-end still has a few bugs to work out. We've implemented most of the promised functionality, but we still need to allow the user to view their personal schedule, and make sure the front-end can consistently pull objects from the database. We'll be fixing this over the next week. Between now and the end of Transition, we will run all of our planned tests, and add a few UI elements, such as event tags and sorting. Our schedule says that we will also add notifications to the front-end, but we have decided to cut this functionality from our plan. There are a few reasons for this. Firstly, we are slightly behind on our deliverables and want to give ourselves plenty of time to catch up. Secondly, we were told not to implement any more major changes after the end of Construction, and we consider this to be too large of a change to include. And lastly, the notifications functionality was originally marked as a “would” goal, meaning we were fine with not getting to it if it proved to be too difficult. Overall, despite some delays we are in fairly good shape, and we anticipate few problems with polishing off our UI, debugging our database retrieval, and testing the overall project in time for our final presentation.