SW Engineering CSC648/848 Spring 2019

Milestone 3

SimpleWrestling

Team 6
Huy Nguyen: Back End Lead
Michael Swanson: Front End Lead
Alexander Nunez: Back End Developer
Heewon Han: Front End Developer
Tuan Le: Back End Developer

May 12, 2019

1) Product Summary

SimpleWrestling

Committed functions:

- Login / sign up
- Coaches can manage a roster for their team, add, remove, modify wrestlers.
- Coaches can join tournaments posted by the admin.
- Admins can add, remove, modify tournaments.
- Admins can audit requests to join tournaments.

About SimpleWrestling

The product itself is something unique because it is in a market that has not been changed in over 10 years. The current websites used in the Wrestling Industry is old and needs a new product.

The current product has not had an update in over five years. This initial prototype is meant to show the abilities of creating and managing rosters, the final goal is to manage brackets and officiate tournaments as well. We are prepared to make a difference in this community because they deserve better.

1) Usability Test Plan
Search for a Tournament

Test Objective

The search bar is a input field that queries our database for any tournaments, and displays the results in a neatly order card format. Our goal is to make tournaments visible and advertiseable, and to reduce the effort for our users, who are coaches, to find tournaments that are relevant to them.

The search feature is a critical and very public facing component in our application, it is of the highest priority that we ensure that our user's are capable of utilizing our search bar and also enjoying their experience with our search bar.

Test Description

In order to perform a usability test we are going to have a new user navigate to our search bar from the root of the application, search for a specified tournament, and click the join button on the tournament page.

Using the latest version of *Chrome, Safari, or FireFox* the user will start at the root route of our application is the home page, at **/home**. From here the user will navigate the Tournaments button the navbar, to get to **/tournaments**, on the tournaments page the user will be presented with a search bar, and a default selection of tournaments, to narrow the search results the user will enter their search query into the search bar.

After submitting, by pressing the *enter* key, or by pressing the search button the UI, a collection of tournament results will appear underneath the search bar in card format.

The search is to be successful if the searched for query is on the first page of the results after entering a query.

The user will also be given a questionnaire after the test to determine their overall sentient with, navigating the website, the intuitiveness of the search bar, and the clarity of the search results.

Questionnaire

Action:

*The user navigated from the **/home** page to the **/tournaments** page.*

Question:

- Navigation between pages in the application was easy and/ or intuitive.
  (Circle One)
  Strongly Agree, Agree, Neutral, Disagree, Strongly Disagree
  *Comments:*_____

Action:

The user entered the search query into the search bar and submitted the search to display a series of tournament results.

Question:

- The buttons on the search bar were large enough to be visible and clickable and were pleasing to the eye.
  (Circle One)
  Strongly Agree, Agree, Neutral, Disagree, Strongly Disagree
  *Comments:*_____

Action:

The user selected a tournament from the search results, navigated to that page, and clicked the join tournament button.

Question:

- Finding your result in the collection of search results and the join button on the subsequent page was clear.
  (Circle One)
  Strongly Agree, Agree, Neutral, Disagree, Strongly Disagree
  *Comments:*_____

Metrics

Use Case	% Completed	Errors	Comments
Navigating to the tournaments page.	100%		Navbar is generally easy to use.
Searching for a tournament.	90%	User could not find the search result due to exact queries.	There are no additional filters. Search works by name only, not location.
Joining a tournament.	90%		The card format is visually pleasing and easy to understand.

3) QA test plan

QA Objective

Ensure key features are functional and meeting or exceeding requirements. White box testing is used on the requirements to ensure that promised features are actually baked into the application and working at a base level.

QA Test

A coach user shall be able to modify, add, and delete to their roster from their home page dashboard.

Test Plan and Overview

The user will attempt to perform create, modify, and delete operations on their roster. The user will begin as an unauthenticated user to additionally test authenticated and authorized routes within the application.

The user will be using the most recent development build of Simple Wrestling, using the latest build of Chrome (74.0.3729.131), on Ubuntu (10.04.2 bionic).

<u>Test cases:</u>

- *Coaches* reroute to their home/dashboard after login
- *Coaches* have a roster
- *Coaches* can add wrestlers to their roster
- *Coaches* can remove wrestlers from their roster
- *Coaches* can modify wrestlers on their roster
- *Coaches* can save or discard any changes made to their roster
- *Changes* saved to the coach's roster is persistent (made to the database)

Number	Description	Input	Output	Result
1	Login and redirect to Coach dashboard.	Username: CoachTest@SimpleWrestling.email Password: 123123123abc!	User is redirected to Coach's dashboard landing page.	PASS
2	Coach user is able to add a wrestler to their roster.	*+ Wrestler* -> Name: TestWrestler Age: 20 Weight: 150 Class: 150	*roster* = {...roster, {"name": "TestWrestler", "age": 20, "weight": 150, "class": 150 }} TestWrestler is visible on UI	PASS
3	Coach is able to modify a wrestler on their roster.	*TestWrestler* -> Name: TestWrestlerTest	*roster* = {...roster, {"name": "TestWrestlerTest", "age": 20, "weight": 150, "class": 150 }} TestWrestlerTest is visible on UI	PASS
4	Coach is able to delete a wrestler on their roster.	*TestWrestler* -> *Remove*	*roster* = {...roster, {"name": "TestWrestlerTest",	FAIL *Delete Wrestler button is missing from*

			“age”: 20, “weight”: 150, “class”: 150 }} TestWresterTest is visible on UI	*roster*
5	Coach is able to save their roster.	*+ Wrestler* ->* Name: TestWrestler Age: 20 Weight: 150 Class: 150 ->* *Save Roster* ->* *(refresh page)*	*roster* = {} TestWrestler is no longer visible on UI	FAIL *Save Roster query on backend is not linked to frontend api*

QA Test Results Summary

The test coach user was able to complete the first 3 use cases of the QA test, there are two features, remove wrestler, and save roster, that are missing, the end to end connections for *delete wrestler* and *save roster* have not been fully implemented and connected.

According to the implementation road map, they are scheduled to be completed in time for the product demonstration. This QA test will need to be repeated once further progress on coach dashboard has been made.

4) Code Review:

a) **Code structure**:

☑ Tab size: 2 spaces
☑ Newlines: All files have a newline at the end
☑ Encoding: All text files must be encoded with UTF-8
☑ Keywords: simple definition name for all functions

b) Indenting and alignment:

☑ General style: Braces are placed on the same line as the start of the function, conditional, loop, etc. The else/elseif is placed on the same line as the previous closing brace.

Example code:

```
@Component({
 selector: 'app-advanced-search',
 templateUrl: './advanced-search.component.html',
 styleUrls: ['./advanced-search.component.css']
})
```

```
 openDialog(message: string, subscribe: boolean = false) {
   const dialog = this.dialog.open(RegisterDialog, {
     width: '250px',
     data: {
       message: message
     }
   });
   if (subscribe) {
     dialog.afterClosed().subscribe(result => {
       this.router.navigate(['/properties']);
     });
   }
 }
}
```

- ☑ Names of functions
  - Clear and easy to understand: AdvancedSearchComponent, openDialog, onSearchClick
  - Should change or add more comment on these functions: ngOnInit, ngOnDestroy
- ☑ Braceless control structures:
  - All blocks have good statement

Example code:

```
onSearchClick() {
   if (!this.listingSearch.city.length) {
     this.openDialog('Please enter some text for the city
field');
   }
   else {
     this.isLoaded = false;
     this.searchService.getSearchListings(this.listingSearch)
       .subscribe(listings => {
           this.isLoaded = true;
           this.searchService.saveSearchListings(listings);
           this.router.navigate(['/properties']);
         },
         err => {
           this.isLoaded = true;
           this.openDialog('Unable to retrieve any listing based
on your search. Please try again');
         });
   }
 }
```

☑ JS, CSS, and media files

- File names match module name:

```
import {Component, OnDestroy, OnInit} from '@angular/core';
import {Router} from "@angular/router";
import {ListingSearch, SearchListingsService} from
"../core/services/search.listings.service";
import {Listing} from "../core/services/listings.service";
import {MatDialog} from "@angular/material";
import {RegisterDialog} from "../register/register.dialog";
import {FormControl} from "@angular/forms";
```

☑ Return values (in particular error returns) are not ignored.

☑ Constants and literals are not hard coded.

☑ All variables used have obvious or descriptive names, and correct scope.
☑ Local functions and non-automatic variables are declared static.
☑ Code is logically correct (Code performs intended functions, operates correctly)
☑ Errors are detected and handled, and processing continued
☑ Error handling conventions are followed (standard use of error handling task, etc.)
☑ The structure is clean and indentations correct.

c) Feedback:

☐ The module must have a module header block containing:

- Name file:
- Description:
- Copyright

☐ Every function must have a comment header block containing

-

5) Self-check on best practices for security

Sensitive information:

- Passwords, middleware to hash passwords before storing into database
- Caching authentication tokens for currently logged in session

Authorizations:

- Coaches authorized to create and modify their roster
- Admins authorized to create and modify all tournaments

Input Validation Layers

Front end layer validation on login signup used for user experience and to lessen the number of bad requests to the backend.

- Required fields
- Email formatting
- Max characters

Back end layer of validation to ensure database type checks are proper and to prevent SQL injection attacks.

- Required fields
- Email formatting
- Max characters
- Sanitise escape characters

6) Self-Check: Adherence to original Non-Functional specs

Security:

1. Login shall be mandatory for coach and admin - **DONE**
2. Username shall be the user's registered email - **DONE**
3. Password shall be encrypted before saving it in the database - **DONE**
4. User's session shall be ended after 30 minutes of inactivity - **ON TRACK**
5. User's session timeout limit shall be set up by the admin - **ON TRACK**
6. User's session shall only be ended by code design - **ON TRACK**
7. Uploaded content from any part shall be audited by the admin - **DONE**
8. This site shall not accept third-party cookies - **ON TRACK**

Audit:

1. New *join tournament* requests shall be audited by the admin - **ON TRACK**
2. New *join tournament* requests shall be approved by the admin - **ON TRACK**
3. The site Tournament Administrator shall be the only person authorized to configure the host interfaces - **DONE**
4. Coaches shall not be allowed to modify any web configuration files - **DONE**
5. Coaches shall not be allowed to login into the admin page - **DONE**
6. Unregistered users shall not be able to login into other coaches or admin pages - **DONE**

Performance:

1. The site loading time shall be less than 1 second for all the screens - **DONE**
2. “Search” shall be executed in a background thread for improving performance - **ON TRACK**
3. Query shall be executed in a background thread for improving performance - **ON TRACK**

Capacity:

1. The total data storage allowed by the web site shall not exceed 80 % of the server capacity for this site - **DONE**
2. The web site shall be prepared to support scalability for adding future new features - **DONE**

3. The web site shall be capable to handle at least 50 users - **DONE**

Reliability:

1. Downtime for maintenance shall be less than 1 hour per month - **DONE**
2. Downtime for maintenance shall not affect the main functionality of the site - **DONE**
3. In all cases, downtime for maintenance shall be informed to the users either by email - **ON TRACK**
4. In all cases, downtime for maintenance shall be informed to the users by publishing an announcement in a visible place of the main page.

Recovery:

1. In a total failure case, the whole site should be put down to revision - **DONE**
2. If broken, the mean time to recovery shall not exceed one day - **DONE**

Data Integrity:

1. Database tables shall be backed up every week
2. Tournament Administrator shall be able to execute a recovery when needed
3. Images size shall be limited up to 1 megabyte
4. Images shall be uploaded in the correct format (jpg or jpeg)
5. Video content shall be uploaded only by Tournament Admins in a compressed format such as mpg4

Compatibility:

1. The site shall be compatible with the last version of Safari browser - **DONE**
2. The site shall be compatible with the last version of Firefox browser - **DONE**
3. The site shall be compatible with the last version of Chrome browser - **DONE**
4. The site shall be compatible with at least an old version of all the browsers listed above - **DONE**
5. Third party applications shall not be able to modify any content that may affect the site compatibility - **DONE**
6. The site shall be ready to support with any or minimal changes any other compatibility that may be added in future versions - **DONE**
7. The site should be compatible to escalate to new databases - **DONE**

Conformance with Coding Standards:

1. Architecture and design standards shall meet all the requirements listed under the High-Level Architecture section of this document - **DONE**

2. Only working code that meets all the code standards shall be submitted to the project repository - **DONE**

3. Any working code shall be tested and debugged before being considered to be working code - **DONE**

4. Any internal errors or exceptions returned by the code shall be stored in a log - **DONE**

5. Any error that may affect the functionality of the site shall be reported to the user - **DONE**

6. Any error shall be handled in a way that does not affect the functionality of the site - **DONE**

7. The whole production cycle of this site shall be finished 2 weeks before the delivery date - **ON TRACK**

8. This site shall not be launched without all the priority one featured finished and tested - **ON TRACK**

9. This site shall be tested and debugged as a whole 2 weeks before the delivery due date - **ON TRACK**

Look and Feel Standards:

1. The application and its layouts shall look professional - **DONE**

2. The site shall be enough simple to handle by all the parties involved - **DONE**

3. Elements on the screen shall have the correct density to meet the compatibility standard of the browsers - **DONE**

4. Elements on the screen shall have the correct size to meet the compatibility standard with all mobile devices - **DONE**

5. Elements on the screen shall have rich and beautiful colors for user delight - **DONE**

6. The site shall be able to work correctly without mouse interaction - **DONE**

7. The site shall be able to work correctively without keyboard interaction - **DONE**

8. Elements in screen shall be resized automatically without user interaction when being loaded in all the different platforms supported by the site - **DONE**

Internationalization / Localization Requirements:

1. The default language of this site shall be English - **DONE**

2. The site shall support scalability to add more supported languages in future versions - **ON TRACK**

3. The site shall support geolocation to locate tournament addresses in a map. - **ON TRACK**