Fitness Achievement For Kids

Test Plan

November 13, 2021

Team Achieve:

Alan Marin

Ivan Fang (Team Leader)

John Bower

Hoyeon Moon

Table of Contents

1 Introduction

The Test Plan is designed to prescribe the scope, approach, resources, and schedule of all testing activities.The plan will identify: items to be tested, features to be tested, types of testing, personnel testing responsibilities, resources, and risks.

2 Test Methodology

2.1 Testing Scope:

In-Scope

- *Unit Testing*
- *Integration Testing*
- *End-To-End Testing*

Out-of-Scope

- *Performance Testing*
- *Penetration Testing*
- *Regression Testing*
- *UI Testing*
- *Database Testing*
- *Web Security Testing*

2.2 In Scope

Unit Testing

The smallest testable parts of the application will be tested to pass before further testing is to be performed. This includes all parts of a class such as methods and constructors. To pass unit testing, each part of our code such as methods, calls, and variables must result in expected results. If the result of the unit test returns successful, for example in a method that deals with input validation, then the unit test is considered passed. Unit testing will be done during each sprint cycle after completing a piece of code that is testable, such as method or class. The constraints for each piece of code undergoing unit testing are that it must take no longer than 10 seconds to complete. Ideally a unit test should be completed in a very quick time frame such as 1 or 2 seconds, but due to variability in how complex our code may be, we may have to optimize our code to fall under the constrained time.

Integration Testing

Integration testing is going to be used to determine if the components of our web application work together as expected. Integration testing will not require the whole system to be tested but rather a bundle of features/components will be tested to make sure there are no compatibility issues.

End-to-End Testing
End to End testing will be done to test the entire application as a whole, to make sure all components function and features work together as intended. We will measure the success of an end-to-end test by analyzing the behavior of the application to match the expected behavior. The flow between components within the web app must be seamless and must provide the expected behavior.

Test Data
Our testing process will make use of mock data that will allow us to simulate data entries and user actions that may occur in our application. All of the testing will be recorded to help us in troubleshooting the issues in our code.

2.3 Out of Scope

Performance Testing
Performance Testing is a type of software testing that ensures our application performs properly under the expected workload. We are going to perform this to check the network, load balancer, database and web server for our load level we are planning to support.

Penetration Testing
This test will be performed to test our security. This includes gaining/maintaining access. We are going to perform this once we are done making our application and then perform at least once a year to ensure security.

Regression Testing
We are going to perform this testing when there are modified parts of code. We can save time by automating the process of this testing by using a tool usch as WATIR.

UI Testing

This testing will be done to check if all UI works correctly. This can be done either manual or automation. GUI will be checked by the person who is in charge of designing.

Database Testing
This is a type of software testing to check schema, table triggers etc. We are going to create queries to check its job for specific functionalities.

Web Security Testing
This testing will be done to find potential vulnerabilities in our web application and to detect possible security risks. Penetration testing is included in web security testing. We will check network security, software security, client-side application security and server-side application security.

3 Test Deliverables

3.1 Summary
All testing is not complete until an accompanying filled out test report is filed under that module, and turned into the team's test reports directory. Reports are to be filled out for both manual and automated bench testing. Reports of failed tests are useful, but repeated failures with same or very similar report data do not require multiple reports. In these cases, simply note the number of times the test fails, or the frequency of failure in the test notes block.

3.2 Test Report Template

Test Number and Name	X.X.X Test Case Name
Test Operation	Manual Bench Test/ Automated Bench Test
Test Performed By	Team Member Name
Test Data	List test data used.
Test Result	Pass/Fail
Notes	Include test code used, screenshots or video of major steps in the test such as input and end state, Pass/ Fail flags from IDE, runtime result if needed, ect.

4 Test Dates

2/18/2022	5.1.1, 5.1.2, 5.2.1
2/18/2022	5.1.3, 5.1.4, 5.1.5
3/4/2022	5.1.6, 5.3.1, 5.3.2
3/11/2022	5.3.3, 5.3.4, 5.3.5
3/25/2022	5.1.7. 5.1.8, 5.5.1
4/1/2022	5.4.1, 5.5.2
4/8/2022	5.4.2, 5.5.3
4/15/2022	5.6.1, 5.4.3
4/22/2022	5.6.2
4/29/2022	5.6.3

5 Test Case Modules

5.1 Equipment Rentals

5.1.1 View Equipment Rentals
Test Operation: Manual
Test Summary: Equipment inventory stored in our database is viewable to users through the equipment rental feature of the web application.
Dependencies: Database is operational and connected to our backend code.
Preconditions: Sample data has been inputted in the equipment tables consisting of three rows of different equipment, with all columns containing information (not null).
Test Steps: 1. Load equipment rental page. 2. Select location of equipment corresponding to sample data.

Test Data: Different valid sample data stored in each row except for location which will be the same for all.
Post-Condition: Three rows of sample data are displayed for the location.
Failure: View does not include equipment saved in the database that matches the equipment location being viewed. View is not consistent when page is reloaded, ie, data is being lost. Relevant equipment is not observable to the user in under six seconds, ie table view/ or web page does not load with data in less than six seconds.

5.1.2 Update Inventory
Test Operation: Manual
Test Summary: User interface is functioning to allow equipment data to be added to the system.
Dependencies: BRD Use case 1 in function. Database is functional. User login is functional.
Preconditions: User is logged into the system as a coach or admin user.
Test Steps: 1. View equipment inventories. 2. Select add equipment to a location. 3. Enter valid input for new equipment to be added to the database.
Test Data: Any valid input for sports equipment.
Post-Condition: System now stores the new equipment in the database. This equipment can be viewed on the equipment inventory page.
Failure: Inventory table is not updated with new information in less than six seconds.

5.1.3 Add and View Equipment Rentals
Test Operation: Automated
Test Summary: Equipment inventory stored in our database is viewable to users through the equipment rental feature of the web application.
Dependencies: Database is operational and connected to our backend code.
Preconditions: Test function is written that randomly generates 80 rows of valid input, (except for location which will be the same for all), for equipment rental equipment to be stored by the system.
Test Steps: 1. Run automated-test to generate rows of equipment data. 2. Load equipment inventory page and view 20 rows of info per page. Expand the view to 80 rows.

Test Data: Random valid input generated for all columns except for location which will be hard coded the same for all equipment data rows.
Post-Condition: Inventory of equipment at location now holds the 80 rows of random data valid data added to the database and this can be viewed on the equipment inventory page.
Failure: Inventory table is updated with new information in under ten seconds. Page does not display twenty rows of data, or eighty rows if expanded.

5.1.4 Check Out Available Equipment
Test Operation: Manual
Test Summary: Equipment that is available in the inventory at a location can be selected for check out by a user of the system.
Dependencies: Working database connection. Working email API for the system.
Preconditions: User account is in the system, as well as a coach account controlling a location that has at least one equipment item available for rental.
Test Steps: 1. Select equipment to check out. 2. Select valid pick up date for equipment and select to confirm.
Test Data: Working user email address associated with the user account.
Post-Condition: User is returned to the inventory page and can view the equipment as unavailable. User and coach controlling location receive email confirming equipment rental.
Failure: Equipment inventory does not update with equipment unavailable in under ten seconds. Equipment rental information is not emailed to the user and coach in control of location in less than ten minutes.

5.1.5 Add to Waitlist
Test Operation: Manual
Test Summary: Users can select to be added to the waitlist and receive the notification email.
Dependencies: Working database connection, and email API's in backend code.
Preconditions: User is logged into the system. At least one sports equipment is unavailable at a location.
Test Steps: 1. User selects to add to the waitlist for equipment that is unavailable. 2. User confirms to add to waitlist.
Test Data: Working user email address for waitlist notifications.

Post-Condition: User is returned to view inventory page, and an email has been sent to the user notify them of their position on the waitlist for that equipment. The waitlist for the equipment is generated if need be, and the user is added to the end of the waitlist for that equipment.
Failure: Update to equipment waitlist is not made in under ten seconds. Waitlist feature does not enforce a waitlist maximum of ten different users at one time.

5.1.6 Waitlisted Equipment Now Available
Test Operation: Manual
Test Summary: Users are notified when the turn comes up on the waitlist to check out equipment.
Dependencies: Working database, email API.
Preconditions: User is the first inline on a waitlist for sports equipment.
Test Steps: 1. Return equipment that the user is on the waitlist for.
Test Data: Sports equipment that is on waitlist to be returned.
Post-Condition: User receives an email notifying them of the availability of the sports equipment they were on a waitlist for. Users can now check out the equipment.
Failure: Update to equipment waitlist is not made in under ten seconds. Email is not sent to user that equipment rental is ready for their pickup. Email is not sent to coach of who has been informed that their rental is ready for pickup.

5.1.7 Admin Changes Waitlist
Test Operation: Manual
Test Summary: Admin or coach users alter the waitlist as they need to.
Dependencies: Working waitlist, email system.
Precondition: User is logged into the system as either an admin or coach user. A waitlist exists on the system for sports equipment item/s. Waitlist contains at least three different users.
Test Steps: 1. View a waitlist for equipment in the system. 2. Move a user to the front of the waitlist. 3. Remove a user from the waitlist.
Test Data: Sports equipment item, all columns in one row complete with valid data. Three different user accounts or more, each on the waitlist. One admin or coach user account.

Post-Condition: The user selected is now moved to the front of the line for the waitlisted equipment. Removed user is no longer on the waitlist and receives a notification of removal from the waitlist.
Failure: Removed user remains on the waitlist. User selected does move to the front of the line. Update to waitlist does not occur in less than ten seconds in the system.

5.1.8 Return Equipment
Test Operation: Manual
Test Summary: Coach user updates the system to show that equipment has been returned and is now available for checkout.
Dependencies: System's database is functioning, inventory page and user accounts.
Precondition: Equipment rental occurred in the system and has been recorded.
Test Steps: 1. Coach or admin user views the inventory page and observes the equipments row showing it check out. 2. Coach/admin user selects return equipment option. 3. User inputs valid data for return: condition, quantity. 4. User confirms return.
Test Data: Sports equipment condition changed, quantity the same.
Post-Condition: Equipment is now shown in the database as available, with updated condition and quantity if applicable.
Failure: Update to equipment inventory is not made within ten seconds. Invalid data, such as inventory less than one is accepted by the system.

5.2 Geolocation

5.2.1 GPS Locate
Test Operation: Manual
Test Summary: A user can use GPS location tracking to find a nearest donation or equipment rental location.
Dependencies: Google Maps API, Google Maps UI
Precondition: User is viewing the locate equipment page.
Test Steps: 1. User clicks on the GPS locate icon. 2. Website asks location permission 3. User accepts the location request.
Test Data: User Location

Post-Condition: Maps UI now shows users location along with all nearby locations highlighted.
Failure: Request for user location is denied and is not able to calculate GPS location of user. It will prompt the user for a zip code instead.

5.3 Volunteer Calendars

5.3.1 View Calendar
Test Operation: Manual
Test Summary: Check if user can view calendar that is set up to 365 days from current day
Dependencies: User log-in, Google Calendar API
Precondition: User needs to be logged in.
Test Steps: 1. Go to the user profile page. 2. Select the volunteer calendar.
Test Data:
Post-Condition: User can view calendar
Failure: Calendar is not shown / User can view calendar without logging in / Calendar is shown that is not starting from the current date.

5.3.2 Volunteer Select their Sports-Area
Test Operation: Manual
Test Summary: Users can select the sports-area they want to volunteer from the preset of the list. Users can select multiple areas.
Dependencies: Database for volunteering, Calendar API
Precondition: User is viewing the calendar page.
Test Steps: 1. Select one sports-areas. 2.Select multiple sports-area. 3. Don't select.
Test Data: set of volunteering datas with selected sports area
Post-Condition: Return dataset for selected sports-area. If the user didn't select any sports-area, all dataset is returned. Calendar should be updated with selected sports-areas.
Failure: Calendar is not updated / Return incorrect dataset / Return nothing

5.3.3 Volunteer Select Location and Range

Test Operation: Manual
Test Summary: Users can select radius from their location how far they willing to commute for volunteering. Once user select range, user select the place they want to volunteer.
Dependencies: Database for volunteering, Google geolocation API, Calendar API
Precondition: User is viewing the calendar page.
Test Steps: 1. Select range: one mile to fifty miles from the user's current location(five miles increment). 2. Select location
Test Data: Selected range, selected location
Post-Condition: Return dataset within the selected range and location. Calendar is updated.
Failure: GPS doesn't work -> User is prompted to enter a zip code instead. / Return incorrect datas / Calendar is not updated

5.3.4 Select Available Volunteering Time
Test Operation: Manual
Test Summary: Users are able to click the specific time they want to volunteer on the calendar page.
Dependencies: Database for volunteering, Calendar API, Time zone API, Email API
Precondition: User is viewing the calendar page.
Test Steps: 1. Select date 2. Select start time and end time
Test Data: selected date, selected time
Post-Condition: Return dataset with selected date and time. Calendar should be updated. New window is showing up to the user to confirm if the selected sports-area, location, date and time is correct. Email is sent to the user.
Failure: Allowing users to select the same date and time for both start and end of the volunteering. / Return incorrect datas / Calendar is not updated /Email is not sent in 5 minutes

5.3.5 Email for Volunteer Approval
Test Operation: Manual
Test Summary: On the coaches' calendar page, coach can see the volunteer request that is made up to 16 weeks. Request is made once user send email back, and coach can look at it then decide if coach will confirm or not. Once coach

confirms the request, volunteers can see their scheduled appointment on their calendar page.
Dependencies: Coach logged in, Email API, Calendar API
Precondition: Coach is viewing the calendar page, confirmation email is sent from the user.
Test Steps: 1. Add request to database 2. Delete request from the dataset if coach doesn't confirm. 3. Change request status to confirmed if coach approve that request
Test Data: volunteer requests
Post-Condition: calendars for both coach and volunteer should be updated based on the status of the requests.
Failure: Calendar is not updated / request status doesn't change / database is not making any changes

5.4 Community Forum

5.4.1 Browse
Test Operation: Manual
Test Summary: Users can browse selected community forum. Default will be the latest posts.
Dependencies: Database
Precondition: User is logged in, navigated to community forum page
Test Steps: 1. Select forum 3. Select nothing
Test Data: dataset of posts with selected forum
Post-Condition: Return correct posts based on selected forum. Return all posts in time order if there is no selected forum. Page should be updated with the selected forum in time order.
Failure: Return incorrect posts / forum page is not updated / posts are not ordered by time.

5.4.2 Post
Test Operation: Manual
Test Summary: Users create posts that are added to the community forum.
Dependencies: Community forum page, user accounts.
Precondition: User is logged into the system and views a community forum page.

Test Steps: 1. Select post. 2. Enter in data and confirm post.
Test Data: Any short sentence.
Post-Condition: Post is added to the forum page and is observed by the user and all other users view that page.
Failure: Post is not visible after posting within six seconds. Post is not visible to other users. Post does not persist in memory. Post contains inaccurate data.

5.4.3 Post Reply
Test Operation: Manual
Test Summary: A user posts a reply to another post.
Dependencies: Community forum page, user accounts.
Precondition: User is logged into the system and view a community forum page.
Test Steps: 1. Select reply. 2. Enter in data for the reply. 3. Select post reply.
Test Data: Any short sentence.
Post-Condition: Reply is added below original post.
Failure: Reply is not visible after posting within six seconds. Relpy is not visible to other users. Post does not persist in memory. Reply contains inaccurate data.

5.4.4 Update Post
Test Operation: Manual
Test Summary: User edits one of their existing posts.base. User accounts.
Precondition: User is logged into the system and is viewing one of their own posts.
Dependencies: Community forum page. Data**Test Steps:** 1. Select edit post. 2. Enter in or delete some text. 3. Confirm changes.
Test Data: Delete one word from the message and replace it with another.
Post-Condition: User views their post with edits made. Post says update at time stamp.
Failure: Data in the post remains unchanged. Changes to the post cannot be seen by another user. Post is not marked with an edit made at date time stamp.

5.5 Fitness Calculator
5.5.1 Select Calculator

Test Operation: Manual
Test Summary: Presents fitness calculator options to the users. The calculators return results for individuals according to the activity completed. The fitness calculators are for basketball, softball/tee-ball, soccer, volleyball, running, dance and yoga. Users enter information about duration of activity, level of intensity, distance run, height, weight and age when applicable.
Dependencies: Fitness Calculator Page UI, User Logged In
Precondition: User is logged into the system under their account and has navigated to the fitness calculator page.
Test Steps: 1. User views calculator page with calculator options. 2. User selects one of the calculators 3. User is brought to the specific calculator page 4. User inputs data into required fields.
Test Data: User fitness calculator measurements
Post-Condition: Results of the calculator are displayed to the user on the page. User is able to input new data or exit the calculator. Fitness Calculator data is stored on the user's profile page.
Failure: If a user is not logged into their account, they will be redirected to the login page. If user inputs invalid input, such as numbers outside of range. An Error message appears stating the acceptable valid input range.

5.5.2 Delete Calculator Records
Test Operation: Manual
Test Summary: Users can select one or more records to be deleted. After selecting record/s to be deleted, a prompt requires the user to confirm deleting records or cancel. After deletion the records are no longer able to be viewed by the user and will no longer be saved in the system.
Dependencies: User Account Created, Fitness Page
Precondition: User is logged into the system under their account and views their calculator results on their profile page.
Test Steps: 1. User selects check boxes next to records they wish to delete. 2. User clicks the delete button at the top of their records list.
Test Data: Fitness Calculator Measurements,
Post-Condition: User views their profile page and the records have been deleted are no longer seen. System deletes records from the database.

Failure: If the user is not able to delete their records from the database, we will display an error message to the user.

5.6 Sports and Fitness Education Teaching Plans

5.6.1 Education Page

Test Operation: Manual
Test Summary: Education page allows users to navigate and select education plans.
Dependencies: User accounts, account privileges enforcement
Precondition: User is logged in. A premade sample education plan.
Test Steps: 1. Navigate to the education plans page. 2. View available education plan. 3. Select to view a sample education page.
Test Data: Education page with sample data. Can be short simple sample titles and minimal text entries, just enough to check the data is correct.
Post-Condition: User can view an education plan.
Failure: Page does not load in less than ten seconds.

5.6.2 Publish Education Plans

Test Operation: Manual
Test Summary: Coach level users can create education plans.
Dependencies: User accounts, account privileges enforcement
Precondition: User is logged in as a coach. A premade sample education plan.
Test Steps: 1. Navigate to the education plans page. 2. View available education plan. 3. Select to publish new educational plan. 4. Entry in data. 5. Confirm to publish educational plan.
Test Data: Minimal valid data for each input field, one word titles, and three word text bodies.
Post-Condition: User views their published education plan on the educational plan page.
Failure: Page does not load in less than ten seconds. Invalid input accepted.

5.6.3 Enroll in Sports/Fitness Lessons

Test Operation: Manual

Test Summary: Users can enroll in an educational plan and be added to the that lesson plans roster of enrolled users.
Dependencies: User accounts, email.
Precondition: User is logged into the system and with a working email. Coach with working email has published a lesson plan.
Test Steps: 1. Navigate to the education plans page. 2. Select an education plan. 3. View lesson plan summary. 3. Select confirm enrollment.
Test Data: Enrolling user account and coach user account. Educational plan with sample data.
Post-Condition: Roster of enrolled students is update on the lesson plan. User account page list the enrolled in lesson plan. Email confirmation sent to user and coach about enrollment.
Failure: Emails not sent. Enrollment occurs with selecting to confirm enrollment. Enrollment occurs for wrong lesson plan. Lesson plan not recorded on user account. Lesson plan roster does not record user account as enrolled.

6 References

Test methodology: https://www.geeksforgeeks.org/types-software-testing/?ref=lbp