Title: Software Design Document

Date: 04/01/22

Olympic Developers

Peter Galvan, Camden Hortline,

Jason Gaglione, and Collin Rampata

Sponsor: Sean H. Anthony

Mentor: Melissa D. Rose

Overview: In this deliverable, the team clearly lays out the software testing plan for the HYPO2 management application. This will give information on Unit, Integration and Usability testing the team will be deploying soon.

1. Introduction

HYPO2 has been extremely successful. The current business model still faces a few challenges relating to client communication and scheduling. HYPO2 staff communicate with their customers through individual text messages and emails which often requires constant, redundant work. For example, when a HYPO2 client wants to change a scheduled activity, they first ask HYPO2 if the change is possible, leading HYPO2 to cross reference data from all of the camps, facilities, and services. If the change is approved, a master PDF must be manually updated for the client and communicated with all HYPO2 staff through email. If the change is denied, there is a negotiation of other options and the process begins again. This could lead to using valuable camp time to plan rather than to train. The variability of training camps alongside the lack of a standard for automated communication has required HYPO2 employees to dedicate time to company operations that could instead be devoted to camp performance.

This application is to improve on the existing workflow by using a software system that allows employees to effectively communicate with clients, create training schedules, and receive notifications of changes in scheduling or client requests. The solution is a web application that offers a client portal allowing coaches to access and change their schedules with ease. This works alongside an admin portal that allows HYPO2 staff to access and update training camp information such as activities, costs, and team schedules. Any updates made within these portals will be communicated automatically to respective recipients through digitized templates already used by HYPO2. The Olympic Developer’s solution creates an opportunity for HYPO2’s employees to better provide real time schedules and up to date information to prospective clients regarding their offered services.

Software testing is used to make sure the product works properly and accounts for all user edge cases for error. This is needed because it allows the team to test whether the application can break via user interaction. This is a necessary step to take before deploying the product and handing it off to the client because it allows for a user-friendly experience and minimal bug interference. Without software testing, bugs are more prone to seep into the application. In order to have a successful testing plan, the team has broken up the testing into three categories. Those categories include: unit testing, integration testing, and usability testing. Unit testing is where the team will be testing individual functions to ensure that the functions return proper values within the application. This allows these functions to be displayed or used correctly throughout the application. Integration testing will be enlisted where the team takes all of the modules and classifies them within their respective areas. Usability testing for this application will be broken up into three other parts due to this application's complexity in having three different types of users. These users consist of: admin, client, and staff. All of these users will be given select tasks that are in relation to their specific user interaction and a rating system of how easy the task was to complete. The team will go through each one of the categories of different software testing areas in a way that outlines their particular interaction and usability within this project.

2. Unit Testing

Unit testing is a type of software testing where code is sectioned into individual units to be tested and validated. The purpose of unit testing is to ensure that individual components of an application work properly and return the expected results. Testing the individual components of a large application helps break down the location of bugs and see the output that each component returns. Testing an application as a whole may take much more time to locate issues in code and present less clarity in incorrect outputs.

2.1 Unit Testing for Post Methods

The web application will communicate with a database where all information inputted into the website is stored. In order to do so, the server side of the application uses post methods to send data to the MySQL database containing all information. Data sent from the web application should be expected to send the correct information and properly store and update in the database to match the data type of the field that is being populated (e.g., int, boolean, char).

2.2 Unit Testing for Get Methods

The web application will need to pull and display information from the database containing the camp information. In order to do this, the use of get methods allows the web application to retrieve information that has been stored in the database. To test that correct data is being retrieved, camps created and stored in the database will be used to act as tests for the get methods where the information will be pulled and displayed. The results from the web application should match the fields in the database.

2.3 Unit Testing for Delete methods

The deletion method in the application will allow for unneeded or unwanted data to be removed whether there is an error in the database or a camp that is no longer needed for documentation purposes. Clients will be able to delete athletes from their rosters but will need to make requests to delete calendar events, confirmation must be made by an authorized user type being staff or admin. The staff will only be able to delete calendar events or confirm calendar deletions from clients. Admin will be able to make any sort of deletions and confirm deletions made by other users. The admin may delete athletes from a roster, calendar events, and camps. When a deletion is made, the data will no longer exist and should not be present in the database.

2.4 Unit Testing for Intake form

The intake form is one of the most important tasks when starting a camp at HYPO2 and will be used to send information to the database. The user will be shown a form to fill out with required and optional fields. If a required field is not completed, then the user will be given a message to complete the field and will not be able to submit the form. There will also be fields when checked yes, which will reveal a text box that may ask for the number of an item needed or extra comments. For text boxes that ask for a number, the user should only be able to enter a number. Once the user fills all required fields, the user may submit the form and will be presented a camp summary with all of the information that was filled out.

2.5 Unit Testing for Add Event Form

Clients will be able to make requests to add an event to their camp on any day and time of day. The Admin will be able to confirm any add event request and the event will be sent to the database and posted to the calendar that is viewable by the client who made the request, the admin, and any staff that is involved in that event. Admin will also be able to add any event to any camp, where the same procedure will follow as who can view the event in their calendar. Any add event request that is rejected should not show in any calendar and should not be stored in the database.

3. Integration Testing

3.1 Integration Testing for Frontend

Our application consists of three different portals based on account classification from our sign-in page. After a specific user signs in to the application, they are displayed the corresponding interface. For example, when an admin class user signs in they see and are only able to interact with the admin interface. Since each classification has a different set of permissions within their interface it is imperative that there is no ability to view a different interface.

For example, if a client class user signs in and then changes the directory to an admin interface page, the application should handle that accordingly. The interfaces fetch and display data from our database, In order to maintain a fully functional application, the application needs to handle errors in fetching and connecting with ease. Every case should have a corresponding descriptive message that is portrayed to the user to ease any confusion about the application's error.

In order to properly test the integration of the front-end the team will deploy the Nock react library to mock HTTP requests to our application. In order to properly test each page the

team will test moving to every other page in the entire application's repository to test if each page handles classification properly.

Since the team deployed the Axios library for sending information from the frontend to the backend. The team needs to test if the values being sent are staying consistent in not only value but type given these values will be stored in the database. In order to test edge cases, physical tests of all Axios statements in the front-end will be manually tested and logged into the console with their value and type. As well as these values will be tested again within usability testing to test real world applications of the functionality. Axios is also used to get information back from the backend, to test appropriately, test cases will be manually stored in the database, and expected outcomes will be created in a separate document and cross-referenced with the values displayed to assure the proper outcome.

3.2 Integration Testing for Backend

Our Backend connects to our MySQL database and the team needs to make sure that the API between our backend and database technologies is sending and receiving information properly. The information stored in the database is the lifeblood of the application as all operations are based off of it. In order to make sure the application works as intended the team needs to verify that queries to the database are returning information that is correct.

Data that is being stored in the database will be manually created for edge cases, At test creation the desired database state will also be created, Upon running the test, the desired outcome will be referenced against the actual outcome to verify correctness. Data that is being returned from the database will be created during the front-end testing alongside expected return values and types for each field in the query. The queried data will be logged and checked against the expected return to check the validity of the functionality.

3.3 Integration Testing Diagram

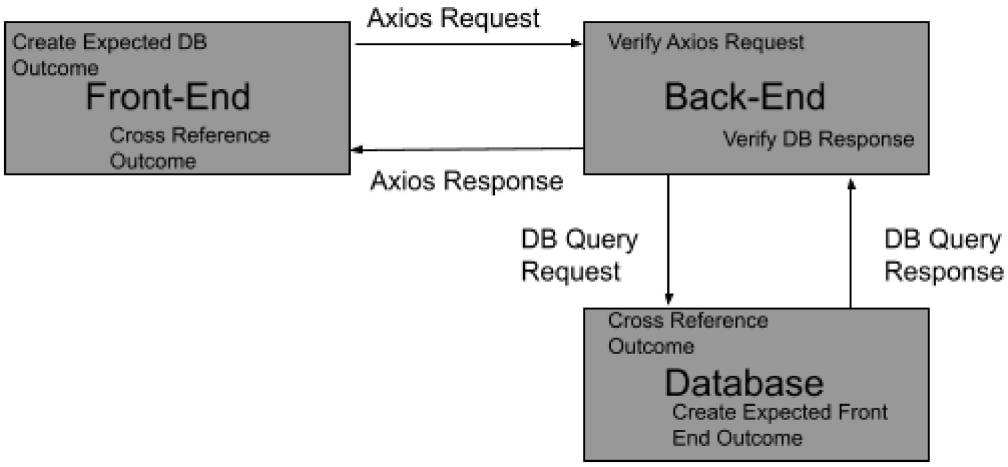

Figure 1.0 Integration Testing Diagram

Looking at the diagram, the tests are being evaluated at both ends of the application. Expected outcomes are created at either end and are then being used to cross-reference with the actual outcome. In the case of unexpected outcomes, the backend verifies values and types from both the Axios Requests and DB responses to catch any mistake in the middle of the process. This allows us as developers to easily pinpoint what side and operation the error resides in.

4. Usability Testing

Usability Testing is a software refinement process that centers around how a piece of software interacts with the end user. This testing method is utilized to evaluate the user friendliness of each software interface and assess the accessibility of each software functionality. By conducting usability testing it allows for the overall user experience of the software to be improved. A typical usability test begins with the developer devising a set of tasks for the end user to perform. Each task is meant to test a specific software feature and after each one is completed, the user will be asked to rate their experience. Feedback can be given in a variety of ways including a five-point Likert Scale, questionnaire, or survey. This information is collected and used to ameliorate the features that the end user found especially difficult to use. Part of this testing process will also be used to refine each user interface. Discussions about the system's overall user experience will take place in an effort to improve the software's usability.

First, the team needed to select candidates for the usability testing. Candidates would be carefully considered in order to ensure a robust testing regime. Since HYPO2's web management portal has three different types of users, three separate focus groups were devised. The 'Admin User' focus group only needed to consist of a single individual - our client, Sean Anthony. Since he will be the web application's main user, the team wanted to tailor the admin user experience to

his liking. This gives the developers valuable insight into how exactly each software functionality should be designed. For ‘Staff User’ candidates, our client will be sharing the web application with HYPO2 staff members. Using actual employees as members of this focus group will yield valuable feedback since these people have first-hand experience working at a training facility. Furthermore, these people know precisely what features they need in the web application to carry out their job effectively. For ‘Client User’ candidates, the team exercised more freedom in who was selected. This is because HYPO2 clients are comprised of professional and casual athletes alike. Thus, the focus group will be made up mostly of pseudorandom individuals. The reasoning behind a pseudorandom selection is to ensure that users in this focus group meet the following criteria: candidates must have prior experience as an athlete. Using this approach would yield candidates who are more likely to be familiar with the terminology present in the software. This will allow the team to gather several unique viewpoints on the user experience of HYPO2’s web management portal.

For HYPO2’s Web Management Portal, the team will be conducting Usability Tests using a combination of the techniques outlined above. Each user will be asked to complete a set of tasks that pertain to the type of account they are using (Client, Admin, Staff). For example, an Admin user may be asked to build a schedule for a training camp. Or, a Client user may be asked to request an event deletion. These tasks will be worded in a way that is purposely vague to simulate the users doing these tasks on their own. At the end of each task, users will be asked to read three statements. They will then rate how much they agree or disagree with each statement using a five-point Likert scale. The development team will then examine the outcomes from each focus group and decide how to refine the user experience moving forward. Included below is the testing regime the development team will adhere to when conducting usability testing.

4.1 Usability Testing for Admin User

- **List of Potential Admin User Tasks:**
    1. Sign in to the web management portal
    2. Add a new user account
    3. Delete an existing user account
    4. Confirm a new training camp request
    5. Decline a new training camp request
    6. Build a schedule for a training camp
    7. Delete an event in the schedule
    8. Confirm a request for event deletion
    9. Confirm a request for event addition
    10. View the training camp roster
    11. Update the price of Ferritin Iron Binding Capacity to $2000
    12. Change the price of Ferritin Iron Binding Capacity back to $38
    13. Archive a training camp
    14. Delete a training camp

- **After each task, the Admin User will be asked to fill out the following survey:**

Figure 2.0

1. I thought this [software feature] was easy to use.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1	2	3	4	5

2. I think that I would like to use this [software feature] frequently.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1	2	3	4	5

3. I imagine that most people would learn to use this [software feature] very quickly.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1	2	3	4	5

4.2 Usability Testing for Client User

- **List of Potential Client User Tasks:**
    1. Sign in to the web management portal
    2. Create a training camp
    3. View billing information
    4. Add five people to the training camp roster
    5. Delete three people from the roster
    6. Request an event deletion from a confirmed camp
    7. Request an event addition in a confirmed camp
    8. Sign Out
- **After each task, the Client User will be asked to fill out the following survey:**

*** See Figure 2.0 ***

4.3 Usability Testing for Staff User

- **List of Potential Staff User Tasks:**
    1. Sign in to the web management portal
    2. View active training camps
- **After each task, the Staff User will be asked to fill out the following survey:**

*** See Figure 2.0 ***

5. Conclusion

Flagstaff has established global recognition as one of the epicenters of Olympic training. The main goal for the Olympic Developers is to streamline HYPO2 operations while maintaining the ease of use for HYPO2 clients. Previously, HYPO2 has had difficulty navigating the proper use of digital scheduling and communication. This often led to redundant operations. However, the new model hopes to eliminate the need for constant emailing to find proper times to set up blocks within user schedules. The team will be testing this outcome over the remainder of the capstone with the three testing methods above unit testing, integration testing, and usability testing. Unit and integration testing will be used to make sure the application works with its intended purpose and that no functions are giving incorrect input as well as no parts of the applications are having trouble connecting to other modules in the system. The team hopes to start incorporating these two testing methods as quickly as possible in the upcoming weeks. Usability testing for the Olympic Developers is a big focal point expressed by our client. If users feel that the application is too hard to use, too confusing, or takes more time than the alternative solution in place then this application will need to be rethought with these concerns in mind. Usability testing is already in the works based on the above plan. The Olympic Developers feel optimistic about the future and the current state of the project to start finding bugs and getting user feedback to turn this fully working application into a market-ready product for HYPO2.