Title: Software Design Document

Date: 02/08/22

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 design of the HYPO2 management application. This will give information on the implementation, architecture, module, interface descriptions and a progress plan for the design of the software.

1. Introduction

Overview

The Olympics have symbolized the pinnacle of professional athleticism for over a century. The games were designed to bring together athletes from across the world as a demonstration of national solidarity. In order to bring pride to their prospective countries, these athletes train as much as they can within the 4 year span between which the summer games are held. The goal of HYPO2 is to offer a fair, competitive edge to athletes who train during off seasons. Athletes attend HYPO2 high-altitude training camps to increase endurance and improve overall health. The purpose is to train and support these athletes so that they can excel in world-class competition.

Problem

While 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.

Solution Vision

The goal of this project is to improve on the existing workflow by creating 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 proposed solution includes 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.

2. Implementation Overview

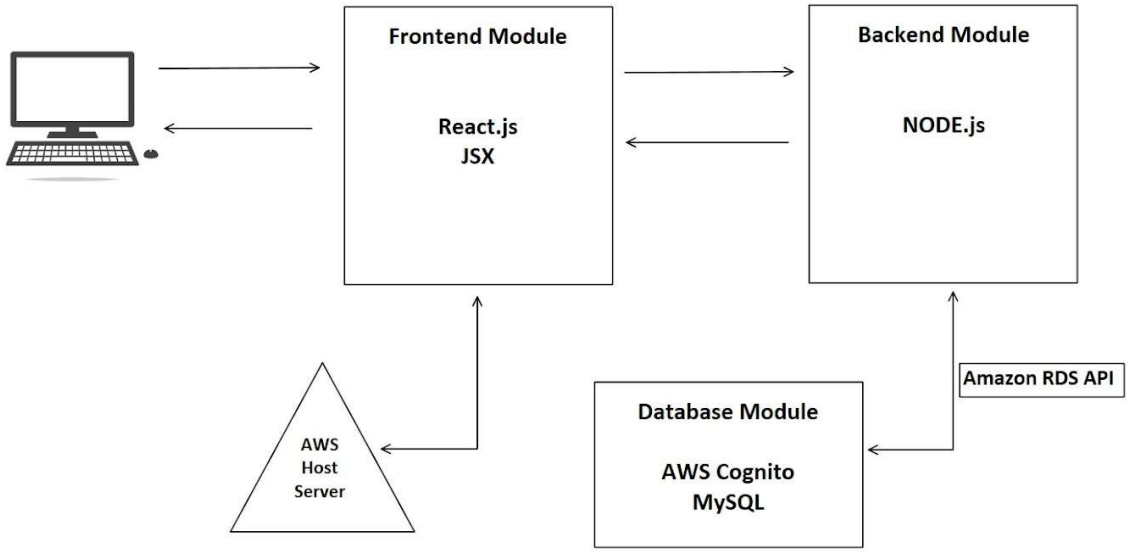
Team Olympic Developers will work with different technologies to create a web application that will centralize all of the needed information and services to streamline training camp management. The web application will be hosted on Amazon Web Services (AWS) where the hosting service experiences minimal downtime to ensure that HYPO2's clients have access to their camp information at all times. AWS also provides other services that will be integrated into the technology design like Simple Notification Service (SNS) that will send reminders through text message to clients.

A MySQL database has been created to store relevant client data, which will allow levels of data access for coaches, staff, and admin, which is necessary to protect sensitive data from users who should not have access to comply with Health Insurance Portability and Accountability Act (HIPAA). The database will be stored in AWS which handles security and ensures the best chance of protecting information.

The web application will use React, a front-end framework, to develop a fact and responsive user interface. React will provide a simple and modular user interface that will allow users to access information that are within their permissions. An application programming interface (API) will be used to communicate information from the database to the web application.

3. Architectural Overview

Figure 1: HYPO2 Management Portal Software Architecture



The database module is responsible for storing all of the relevant information related to HYPO2's web portal. This module is represented by two servers - a MySQL relational database and an AWS Cognito server. The Cognito server will be responsible for storing user account data in a secure user pool. This information will be referenced later in the MySQL database when data like billing information or a camp schedule needs to be linked to a specific user's account. Data can be communicated between the two servers using AWS Amplify which is a frontend developer toolkit. This service provides an application programming interface or API that can be used to link services like Cognito with existing software architecture. The relational database will be used for storing information that is inputted by the user whenever a training camp is created. It is composed of A "GeneralIntake" Table which will hold data like names, phone numbers, and emails. Another Table, "CoreCampNeeds", will hold information like hotel accommodations, and required facilities. Users can request physical therapy and massages during camp creation and this data will be stored in the "AdditionalServices" Table. Finally, prices associated with HYPO2 services will be stored in the "Billing" Table. This table only contains pricing information and is used strictly for creating a bill of services. This information can then be communicated to the softwares backend module through a series of function calls using Amazon's Relational Database Services (RDS) API.

The backend module is responsible for creating the key functionalities of HYPO2's web application. This includes features like viewing camp information, creating a training camp, and uploading health documents. This module will be built using Node.JS which is an open-source, cross-platform, backend JavaScript runtime environment. The Javascript programming language will be used to program each functionality. The backend module will use information from both the frontend and the database modules to carry out its functionalities. For example, whenever a camp is created it will store the values inputted by the user on the frontend in the proper table in the database module. It will also access the database module to assemble scheduling information and pricing information. Node.js is heavily integrated with the applications front-end technology making data translation between the two systems simple.

The frontend module is the part of the application that users will interact with directly. Upon visiting HYPO2's Web portal, all users will be prompted to sign in through a common login interface. The user will then be redirected to one of three dashboards depending on their accounts classification (Admin, Staff, or Client). An Admin user will be displayed the "Admin Dashboard" interface which will contain services scheduled by all users on the platform. Staff users will be displayed the "Staff Dashboard" interface which will show appointments that clients have scheduled with them. Finally, Client users will be shown the "Client Dashboard" interface which will allow them to create new training camps and view active camps. These UI components will be built with React.Js which is an open-source front-end JavaScript library. The JavaScript programming language will also be used in tandem with this technology for interface creation. The software frontend module will communicate directly with the backend of HYPO2's web portal using Express.Js which is a backend web application framework that works well with high speed networks. The website itself will be hosted on an AWS EC2 Host server which can be accessed by SSH. The EC2 server will also provide a secure testing environment for the web application and allow it to run at all hours of the day. The website will be maintained locally and new versions will be uploaded to this server when each release is ready. Combining

the elements mentioned in the frontend, backend, and database modules will yield a minimum viable product for HYPO2.

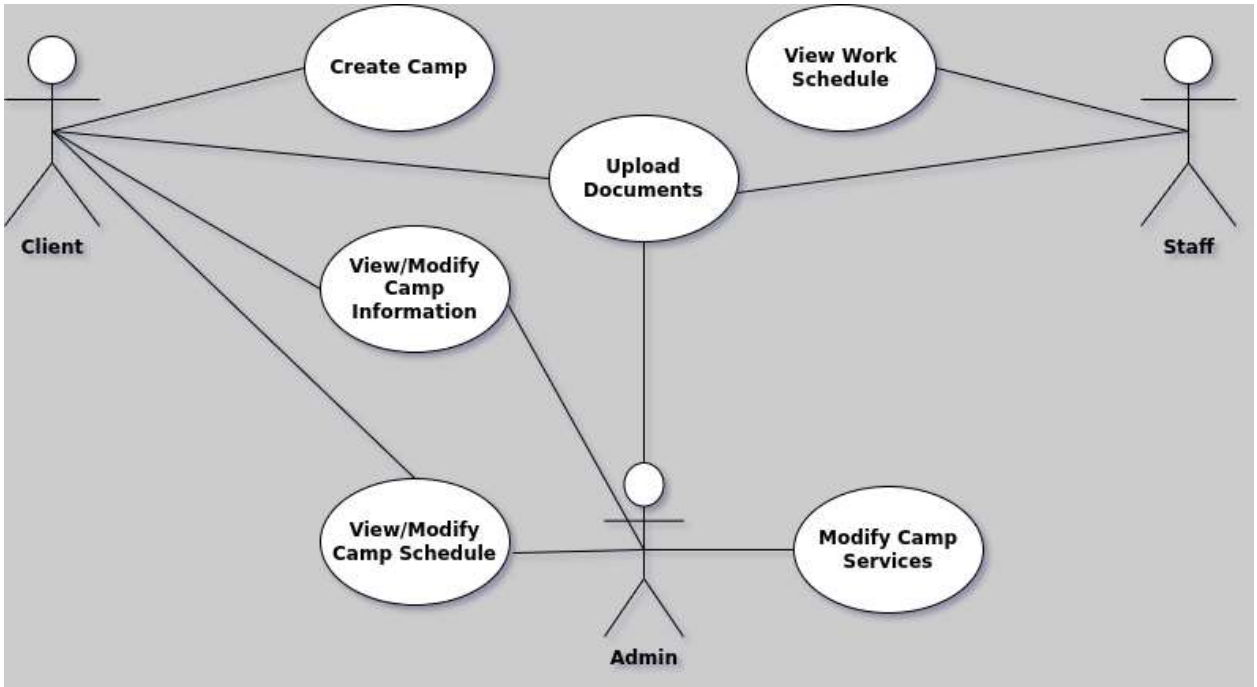
4. Module and Interface Descriptions

The software architectural overview encompasses three modules for our web application: the frontend, backend, and database modules. A detailed description highlighting the required technologies for each of these modules is provided in the following subsections.

4.1 Frontend & Backend Modules

The team will be using React for the front end to create a user friendly interface for the web portal. User information will be stored into a MySQL database. NodeJS will be implemented to retrieve the users' information from the database to display to the user on the frontend. Below, the use cases for the application will be broken down into three types of users: Client, Admin, and Staff. These use cases will be reviewed to assess what is needed to transform them into working modules. The main task of the application is taking information the user provides to make a camp and then displaying this information in the frontend. Due to this, all of the below modules will use frontend and backend technologies as discussed above.

Figure 2: Management Portal Use-Case Diagram



4.1.1 Create Camp

Creating a camp will be accessible to clients where they will request the needed facilities and services. HYPO2 will provide the client with an intake form that will need to be filled out

through checkboxes and textboxes. Once the intake form is filled out, the form will be submitted and the client will be provided with a summary of the information filled out and the estimated billing summary.

Frontend Module: With the “Create Camp” feature, clients will be able to schedule a training camp for a duration of time upon approval of admin. The client will create a schedule that displays fields of needed information with checkboxes where a textbox may become visible if there is further specification needed. After all fields are filled, the information will be sent to the admin for approval through the webportal where the admin will be able to view, modify, and approve the camp.

Backend Module: On the backend, the information inputted by the client will be held in variables using NodeJS where they will be communicated with the intake table within the MySQL database. The information will be put into columns that distinguish the type of service or facility through boolean values that will show the column being true or false. The table will also store a unique camp id that will be used to identify each camp. For each camp created, NodeJS will pull prices from the billing table in the database to create an estimated price for the whole camp.

4.1.2 Upload Documents

Uploading documents will be accessible to clients, staff, and admin. Users will be able to upload documents to the HYPO2 web portal that may include medical documents, forms needing to be signed, schedules in pdf format, and billing forms. Those forms will also be viewable to ensure that all information is correct or in need of any changes.

Frontend Module: The Upload Document features will have three separate access points: the Client, Admin, and Staff depending on the user type. For the interface, the layout will look similar as all user types will be able to upload and access certain documents so React components will be used to display similarities so the code isn’t repeated between all users. The Admin will have access to all documents and ability to upload any document in the system while staff will only be able to access and upload documents authorized by the admin for staff access. The client interface will be shown required actions to upload needed documents by HYPO2 where they can also view any uploaded documents within that camp.

Backend Module: The documents need to be put into a proper place within the database. This will be done by communicating with the database using NodeJS and storing these documents into the blob type in the MySQL database. These documents will be able to be accessed by the appropriate user when the information is needed with NodeJS retrieving the documents from the documents table.

4.1.3 View/Modify Camp Information

Admin and client users will be able to view and modify camp information. These actions may include adding team members to the roster and viewing billing information. The roster

provides HYPO2 with information on who is attending the camp. Viewing the billing information will allow the user to know the price of what the camp will cost at its current state. The billing information will update dynamically with modifications to the camp.

Frontend Module: Camp information will be displayed in separate tabs under the current camp selected. These tabs will include: billing, roster and documents. These tabs will give both users access to viewing and manipulating the camp information that is allowed by their user permission level. For example, in the billing tab, client users can only view the price, but admin users will be able to modify the price if there are any discrepancies.

Backend Module: NodeJS will be getting access to many tables in the MySQL database. These tables will depend on the tab the user is in. For the billing tab, NodeJS will be getting the information from the billing table and giving it to React to display to the user. For the roster, this will be found in a future table that will be added to the database schema in the future. This will be called a roster. The roster table will be attached to the user camp information. For any modifications to the roster or billing tab, NodeJS will send back the proper attributes to adding or removing the information stated by the user in the database.

4.1.4 View/Modify Camp Schedule

Admin and client users will be able to view and modify camp schedules. These can include: canceling upcoming services, requesting new services or confirming new services. Client users and admin users will both be able to cancel upcoming services. However, requesting new services will be restricted to the client and confirming a service is restricted to the admin user. This means clients will request services and once the admin has made sure that the service is available, they will confirm the service so that the client knows that the service can take place.

Frontend Module: The camp schedule will be displayed in a user friendly interface using a react package called React Big Calendar. This is a calendar application that will give complete control of how to display the schedule blocks from the database and including how to add services to the calendar.

Backend Module: NodeJS will be getting access to the schedule table and providing the correct information needed to display on the user’s calendar. For example, Team Canada will only see Team Canada’s services. However, for the admin users they will have a master calendar which is all the team's services going on within the organization. NodeJS will pull down all the scheduling blocks and give them to React to display accordingly. For any modifications, the database, NodeJS, will update the proper fields in the schedule table.

4.1.5 Modify Camp Services

Modifying camp services will only be accessible by the admin users. This will include: changing prices that services have and adding new types of services. For example, if the prices of a service that HYPO2 offers, such as massage, needs to be changed from seventy dollars to hundred dollars based on the new salary for employees–this would be done there.

Frontend Module: React will display all the services that are offered. The corresponding prices attached to the service there will be an empty field that is used to change the price of the service when the changes are submitted. There will also be a plus icon at the bottom of all services where the admin can add a service with the given price of the new service.

Backend Module: NodeJS will be getting access to the billing table and providing the service name and the price that is needed to get that service. When the admin user wants to update the service price or add a new service, NodeJS will take the information provided by the users and put it into its corresponding fields. This updates the old price or creates a new service if the service does not exist.

4.1.6 View Work Schedule

Viewing the work schedule will only be accessible by staff users. Staff will be able to view their scheduled hours throughout the week created by the admin and view tasks needing attention in the HYPO2 portal. Viewing the work schedule through the HYPO2 portal will allow for ease of use and eliminate the use of external services like google calendars.

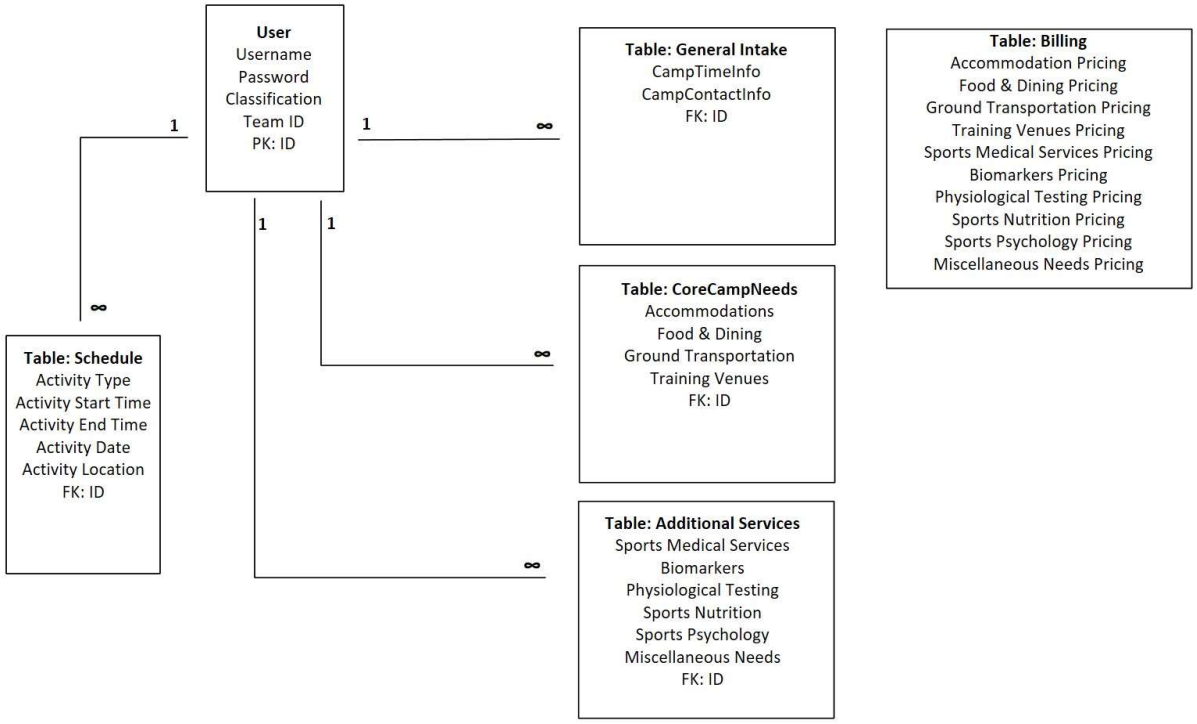
Frontend Module: The purpose is to allow staff users to view their work schedule. For the interface, it will be pulling down blocked out time for services for all camps in session and displaying the appointments for each employee services' specific schedule. So, only massage therapists will see massage therapist appointments and so on. So all job specific employees will have a much similar interface besides the appointments displayed on their schedule. Because of this, React components display the similarity so we don't have to repeat the code between all different job codes.

Backend Module: Only scheduled appointments for HYPO2 staff will be pulled down and displayed. There is currently no plan for anything such as canceling the appointments or manually adding appointments from the staff's point of view. Because of this, there will be no modification of the database from the staff viewing their work schedule.

4.2 Database Module

The Database Module encapsulates the database technologies that will be utilized by HYPO2's web application. It consists of two separate databases - an AWS Cognito server and MySQL relational database. User account credentials will be stored in a secure user pool hosted by AWS Cognito. All information relating to training camps and HYPO2 services will be stored in the MySQL relational database. This design feature ensures that personal user information can be stored securely while information stored in the relational database can be easily accessed. Below is the schema that the database module of HYPO2's web application will follow:

Figure 3: Database Overview



4.2.1 User Table

The User Table will contain information pertaining to user accounts. This "user pool" is serviced by AWS Cognito and provides secure storage for sensitive user information. This database is separate from our relational database, however information can still be referenced from Cognito in MySQL using AWS Amplify. Most notably, it contains the primary Camp_ID key (PK) which is connected to every other table through foreign Camp_ID keys (FK). These keys are used to link camp information between tables. For every one user, there will be several instances of their data in the underlying MySQL tables. This one to many relationship is the foundation of HYPO2's database's, so it is imperative that the following data is included in the User Table:

Username: Every user will have a username associated with their account. A user can be defined by a single individual or an entire team. Usernames will be used to tie relevant camp information like scheduled events to their account. Usernames will be generated by HYPO2 Admin and are data type String.

Password: Every account must be secured by a password. This will protect sensitive information stored with the web application. Passwords are generated by HYPO2 Admin and are data type String.

Classification: Every account will have a specific classification associated with it. This is used to display one of three interfaces as well as account information. Accounts with the

Client classification will be shown the “Client Dashboard” after logging into the web portal. Users with the Admin classification will be shown the “Admin Dashboard”. Accounts with the Staff classification will be shown the “Staff Dashboard”. This information is inputted by HYPO2 Admin and is data type String.

Team_ID: Accounts associated with a specific organization will be given a Team_ID. This signifies that the particular user is more than one person and it allows HYPO2 admin to reference information about a team. Generated by HYPO2 Admin and is data type String.

Camp_ID: Every account will have a set of Camp_ID’s associated with it. Camp_ID’s will be referenced to tie specific information to any given camp. Logically, it is a nametag for a camp. Client accounts will have a set of Camp_ID’s from events that they have scheduled. Staff accounts will have Camp_ID’s from events that they are working. The Admin account will contain the set of all Camp_ID’s. This information will be generated internally by the applications logic to ensure that every Camp_ID is unique.

4.2.2 Schedule Table

The Schedule Table will contain information pertaining to a scheduled HYPO2 camp or activity. This data will be stored in our MYSQL relational database in its own table. Every new instance of a camp or activity will have accompanying information tied to it in the Schedule Table which will ensure that HYPO2’s web application displays the correct scheduling data to its users. The following information will be represented by Column headers in the database:

Activity Type: Every scheduled activity at HYPO2 will have an Activity Type associated with it. This could be “Camp”, “Massage”, “Swim”, etc. This information will be displayed in the “Client Dashboard Interface” along with other relevant formation whenever an activity is scheduled. Activity type is generated internally and is of data type String.

Activity Start Time: Every scheduled activity will have an accompanying start time. This information will be stored in “military time” of form “0000” and be converted to conventional time when displayed to the user. Activity start time is generated internally by the software’s logic and is of data type int.

Activity End Time: Every scheduled activity will have an accompanying end time. This information will be stored in “military time” of form “0000” and be converted to conventional time when displayed to the user. Activity start time is generated internally by the software’s logic and is of data type int.

Activity Date: Every scheduled activity will have an accompanying start date. This information will be stored in form MM/DD/YYYY and will be displayed to the user along with other relevant information. This data is generated internally and is of data type long.

Activity Location: Every scheduled activity will have a specific location associated with it. This is because other facilities may be visited over the progression of a team camp. This will be reflected in our database as a “String” and is generated internally by the software.

Camp_ID: Each activity that is scheduled will have a specific Camp_ID associated with it. This will be used internally so the software can link the above categories to a specific event. Each row in the Schedule Table can be thought of as a single event. The last column will contain the Camp_ID so that the entire row of values can be referenced and displayed to the user in an efficient manner. This is generated internally whenever a user schedules a new activity.

4.2.3 General Intake Table

The General Intake Table will be connected to the user table by a common ID column that will represent the specific camp the information is pertained to. The information stored within the General Intake Table correlates to the logistic and contact information of a camp. In order to have valuable data the following fields will populate the table:

Camp_ID: The camp_ID is the primary key and is being used to create a one to many relationship with the User Table. The campID will be of type int and each camp will be assigned a unique ID at camp creation.

Team_Name: The Team name field will be used to store the specific team name that will be used for output in the camp summary. This field is of type text.

Camp_Start_Date: The Camp_Start_Date field will be used to store a camp's starting date. The field is of type date in the form MM/DD/YYYY.

Camp_End_Date: The Camp_End_Date field will be used to store a camp's conclusion date. The field is of type date in the form MM/DD/YYYY.

Num_Personnel: The Num_Personnel field will be used to store the total number of athletes, coaches and private staff a team plans on bringing to a camp. The field is of type int.

Country Text: The Country field will be used to store the team's origin country. The field is of type text.

Contact_Name: The Contact_Name field will be used to store the name of the lead representative that will be HYPO2's main contact during camp setup. The field of type text.

Contact_Email: The Contact_Email field will store the email address of the lead contact. The field will be of type text.

Contact_Phone Text: The Contact_Phone field will store the phone number of the lead contact. The field will be of type text.

OnSite_Name: The OnSite_Name field will store the name of the person who will be the main contact point during the duration of the camp when the team is present at HYPO2's facilities. The field is of type text.

OnSite_Phone: The OnSite_Phone field will store the phone number of the on site main contact. The field is of type text.

OnSite_Email: The OnSite_Email field will store the email address of the onsite main contact. The field is of type text.

4.2.4 CoreCampNeeds Table

The Core Camp Needs Table will be related to the User table as One User can have many different rows within the Core Camp Needs Table. The Core Camp Needs will store boolean values of services a team signs up for in the intake form. The boolean values will then be used to determine which services will be added to the billing estimate. In order to properly keep track of all information the following fields will be used:

Camp_ID: The camp_ID is the primary key and is being used to create a one to many relationship with the User Table. The campID will be of type int and each camp will be assigned a unique ID at camp creation.

Hotel_Accom: The Hotel_Accom field will store a boolean value that represents if Hotel Accommodations will be necessary for the teams training camp.

Condo_Accom: The Condo_Accom field will store a int value determining how many condo units will be necessary for the teams training camp. A value of 0 means no Condos needed

Univ_Cafeteria: The Univ_Cafeteria field will store a boolean value that represents if access to the university cafeteria will be required for the training camp.

Catering: The Catering field will store a boolean value determined by whether or not a team will need catering services during the duration of a camp.

Charter_transport: The Charter_transport field will store a boolean value that represents if charter transport is necessary for a training camp.

Indiv_shuttle: The indic_shuttle transport field will store boolean value based on the input from the intake form that determines if shuttle services will be necessary.

Rental: The rental field will store a boolean value based on input from camp intake that represents if rental cars are needed.

Permit: The Permit field will store a boolean value based on input from camp intake that represents if a NAU parking permit is required.

Pool50M: The Pool50M field will store an integer value from the intake form that represents how many 50 meter swimming lanes are required for a camp.

Track400M: The Track400M field will store a boolean value from the intake form that represents if a 400 meter track is required.

Track300M: The Track300M field will store a boolean value from the intake form that represents if a 300 meter track is required.

Gym bool: The Gym field will store a boolean value from the intake form that represents if gym access is needed or not,

OutdoorFieldGrass: The OutdoorFieldGrass field will store a boolean value from the intake that represents if a grass field is required or not.

OutdoorFieldTurf: The OutdoorFieldTurf field will store a boolean value from the intake that represents if an outdoor turf field is required or not.

IndoorFieldTurf: The IndoorFieldTurf field will store a boolean value from the intake that represents if an indoor turf field is required or not.

CourtSpace: The CourtSpace field will store a boolean value from the intake that represents if an indoor court will be necessary or not.

CourtUsage: The Court Usage field will store an integer value of how many occasions an indoor court will be used

AntiGravTread: The AntiGravTread field will store a boolean value from the intake form that represents if the Anti-Gravity Treadmill will be needed.

4.2.5 Additional Services Table

The Additional Services Table will be related to the User table in a manner where any one user can have many rows within the Addition Services table. This table will hold information pertaining to any service listed within the Additional Services section in the intake form. This information will be used to populate a camp summary for approval and be used in our back end logic to create a billing estimate. The information will be stored in the following fields:

Camp_ID: The camp_ID is the primary key and is being used to create a one to many relationship with the User Table. The campID will be of type int and each camp will be assigned a unique ID at camp creation.

Massage_Therapy: The Massage_Therapy field will store a boolean value from the intake form representing if Massage Therapy will be used in the training camp.

Physio_Therapy: The Physio_Therapy field will store a boolean value from the intake form representing if Physio Therapy will be used in the training camp.

Strength_Cond: The Strength_Cond field will store a boolean value from the intake form representing if Strength and Conditioning will be used in the training camp.

Ortho_care: The Ortho_care field will store a boolean value from the intake that represents if Orthopedic care services are required in the camp.

Prim_Med_Care: The Prim_Med_Care field will store a boolean value that represents if Primary Medical Care services should be expected.

Hemo_Test: The Hemo_Test field will hold a boolean value determining if Hemoglobin tests will be performed during a camp.

Comp_Blood_Prof: The Comp_Blood_Prof field will hold a boolean value determining if complete blood profiles will be performed during the camp.

Meta_Panel: The Meta_Panel field will store a boolean value determining if a Comprehensive Metabolic Panel will be performed during the camp.

TIBC bool: The TIBC(Total Iron Binding Capacity) field will hold a boolean value determining if a TIBC test will be performed during the camp.

Creatine_Kinase: The Creatine_Kinase field will hold a boolean value determining if a Creatine Kinase test will be performed during the camp.

Other: The other field will hold text of any tests not listed in the intake that will then be output in the camp summary for camp approval. The cost of these additional tests will done manually as they are subject to change at any moment.

VO2_Lactate: The VO2_Lactact field will store a boolean value determining if VO2 Lactate tests will be held during the camp.

VO2_Thresh: The VO2_Thresh field will store a boolean value determining if VO2 Threshold tests will be held during the camp.

Lactate_Thresh: The Lactate_Thresh field will store a boolean value determining if Lactate Threshold tests will be held during the camp.

Supp_O2: The Supp_O2 field will store a boolean value from the intake form determining if supplemental O2 will be used for training.

Int_train_diet_analysis: The Int_train_diet_analysis field will store a boolean value based on input from the intake form to determine if its necessary or not.

Nutrition_Group_Pres_WS: The Nutrition_Group_Pres_WS field will hold a boolean value determining if a Nutrition Group Presentation Workshop is necessary for the camp.

Indiv_Pysch_consult: The Indiv_Pysch_Consult field will hold a boolean value determining if Individual Psych consultations are necessary in the camp.

Psych_Group_Pres_WS: The Pysch_Group_Pres_WS field will hold a boolean value determining if a Psychological Group Presentation Workshop is necessary for the camp

Meeting_Space: The Meeting_Space field will hold a boolean value determining if meeting space is required in the camp

Equip_Stored: The Equip_Stored field will hold a boolean value that will correlate to if a client needs to store equipment in the duration of the camp.

Day_Trip_Excur: The Day_Trip_Excur field will hold a boolean value determining if Day Trip Excursions will occur during a camp.

Team_Build_Exer: The Team_Build_Excer field will hold a boolean value determining if team building exercises will occur during the camp.

otherInfo: The otherInfo field will store text about any other special services not listed in the intake form that a team would like for their training camp. The cost of said services will be inputted manually.

4.2.6 Billing Table

The Billing Table will store the pricing information of HYPO2 services. Since these values may be subject to change they cannot be hardcoded into the software and instead must be represented in a single table in the database. There is no relationship between this table and any of the other tables since this information is independent of the information created by the user. Thus, we will use this table simply for displaying prices and referencing values for the total cost of a HYPO2 activity or camp. The values that will may used in the calculation of the total cost include:

Accommodation Pricing: Expected rate at one of the sponsored hotels. Holds a floating point value representing this cost in USD.

Food & Dining Pricing: Expected cost of food and dining at predetermined locations. Holds a floating point value that reflects this cost in USD.

Ground Transportation Pricing: Expected rate of a Charter bus to transport clients. Holds a floating point value that reflects this cost in USD.

Training Venue Pricing: Expected cost of any external training facility will need to be used. Holds a floating point value that reflects this cost in USD.

Sports Medical Service Pricing: Expected cost of Chiropractic, Physiotherapy, and Sport Massages. Holds a floating point value that reflects this cost in USD.

BioMarkers Pricing: Expected cost of blood testing. Holds a floating point value that reflects this cost in USD.

Physiological Testing Pricing: Expected rate of physiological testing. Holds a floating point value that reflects this cost in USD.

Sports Nutrition Pricing: Expected rate for using nutritionist services. Holds a floating point value that reflects this cost in USD.

Sports Psychology Pricing: Expected cost for any of the mental performance services offered by HYPO2. Holds a floating point value that reflects this cost in USD.

Miscellaneous Needs Pricing: Any special accommodations that need to be made will be included in this category. Holds a floating point value that reflects this cost in USD.

5. Implementation Plan

Figure 4: Olympic Developer Gantt Chart

The Gantt chart above displays our team's estimated timeline for the project. This is based on the requirements and major milestones that our team needs to accomplish. There is a visual key to give information about the task such as: if it is completed, what users the task deals with or if the task is about documentation of the project. Also, in the visual key, it will give information on the week the team is currently working on. The week the team is currently working on is Week 6 (02/14/22). The Gantt chart is designed to have an alpha with all the functions of the application functional by Week 10 (03/14/22). After Week 10 (03/14/22), the team will be focusing on the visual aesthetic and documentation of the website. This will turn the already functional project into a user friendly application. Documentation will be provided to further maintain the development of the product. Currently, the team is focused on three major tasks for the next two weeks: letting the admin create camps for the client after they have requested a camp and allowing the client to request and cancel services once the camp is created. The team is currently on track to get the alpha done at the Week 10 (03/14/22) mark. The Olympic Developers have currently finished 5 tasks: setting up the database schema, Client User Signup & Login & personal accounts, Client User requesting camps, Admin User login and Staff User Login. All of these tasks have been completed on time. The Staff User login did not need to be completed until the end of Week 9 (03/14/22). However, the team has been working in an agile work environment and adapted when creating the Admin and Client User login. The team noticed that creating the Staff login at this time would be more effective.

6. Conclusion

Flagstaff has established global recognition as one of the epicenters for 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 proper use of digital scheduling and communication, which 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. This will be done by using the architectural details explained in this article. MySQL, AWS, NodeJS and React are the technologies that the team will be using for the product. This application will not only benefit HYPO2, but also other companies in the area that rely on tourism such as car rental companies, hotels, restaurants and much more. It is expected that the team's application will give a cost efficient solution to HYPO2 to smooth out the onboarding and the camp for the attending visitors while making profitable sense for HYPO2. As explained within the "Implementation Plan" section, the team has finished 5 tasks. The team has also consistently been meeting target goals and staying on track to have the alpha completed and presentable by Week 10 (03/14/22). The Olympic Developers feel optimistic about the future and current state of the project.