COS 397: Computer Science Capstone I

System Design Document

UMaine Athletic Department Inventory Management System

Version 0.2

Collin Rodrigue, Brennan Poitras, Graham Bridges, Gabe Poulin, Sean Radel

15 November 2023

Table of Contents

1 Introduction

The purpose of the product is to fulfill the customer needs of an inventory management solution. Our system will replace the customer's previous solutions for managing their equipment. The customer previously used Front Rush and currently uses a combination of Excel spreadsheets and word of mouth to track inventory. Our product will allow the customer to organize their inventory by associating equipment with teams and players. The system will be designed with simplicity in mind so that they do not need experienced developers to maintain their product following the delivery date. Equipment, player accounts, and teams will be able to be made on demand to allow the system to scale to the customer's needs. Our customer is the University of Maine Athletic Department, and specifically Jude Killy, Nick Fox, and Kevin Ritz.

1.1 Purpose of This Document

The purpose of this document is to abstractly illustrate our software architecture, further specify our classes, files, and requirements. We aim to document how our product will work together as subsystems of one large system. This document describes the datatypes we will use in our database and how the data objects are connected to each other. This document helps to show the team what software solutions we need to implement to fulfill our requirements.

1.2 References

1. IMSG. “System Requirements Specification” November 1 2023, https://docs.google.com/document/d/1LnOj2DEyu8DPbKXBTDBm2y6UbePr_AXC/edit
2. Google. “Firebase Authentication | Firebase.” Firebase, 2019, firebase.google.com/docs/auth.
3. Vercel. “Next.js by Vercel - the React Framework.” Nextjs.org, nextjs.org/.
4. https://react.dev/
5. “How to Communicate Architecture – Technical Architecture Modeling at SAP (Part 1) | SAP Blogs.” Blogs.sap.com, blogs.sap.com/2008/01/09/how-to-communicate-architecture-technical-architecture-modeling-at-sap-part-1/.
6. “Working with Related Tables.” Help.claris.com, help.claris.com/en/pro-help/content/related-tables-files.html. Accessed 15 Nov. 2023.
7. W3Schools. “SQL Data Types for MySQL, SQL Server, and MS Access.” W3schools.com, www.w3schools.com/sql/sql_datatypes.asp.

2 System Architecture

The system architecture section describes the logical structure of our software product. In the architectural design, we illustrate how the subsystems will abstractly work together and how the user interfaces with our general system. This section will guide the team in choosing solutions for our subsystems. The decomposition description further breaks down the system into functions and objects and finally illustrates our design pattern.

2.1 Architectural Design

The architectural design abstractly illustrates how the subsystems interact with each other.

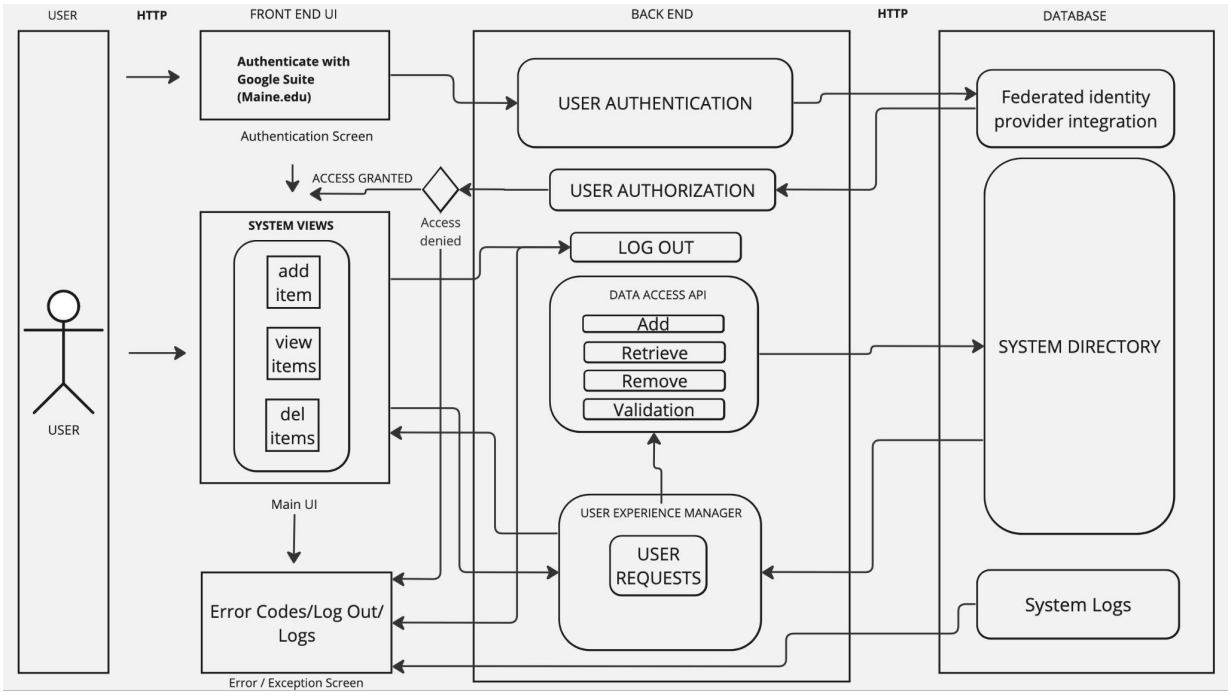

Figure 1. Architectural Design Diagram

2.1.1 Architectural Design Summary

Our system is broken down into three core components: the front end, the back end, and the database. The front end is the portion of the system that the user will see and interact with through their browser. There are three general web pages that the user will be able to see: the login page, the main user interface, and the errors and exceptions page. The user will be immediately prompted to log in on arrival to the website. The system will leverage federated identity provider integration to allow the user to utilize their University of Maine login credentials to access the system like they would with any other University of Maine service. After the authentication and authorization process, the user attains access to the rest of the user interface system. From the main UI, the user will be able to perform actions such as creating, viewing, and removing data from the database. The final interface is shown when the user logs out or reaches an error. The back-end subsystem will process user requests on the web page and handle API calls to the database. The database will be a cloud-hosted solution and will securely store all of our system data as well as manage our user authentication. Alongside storing inventory and profile data, the database will also store system logs.

The team will utilize Firebase and other Google Cloud Platform technologies (GCP) as our database and authentication management solution. Firebase is a scalable cloud-based solution owned by Google and used by leading tech companies. The team plans on using Next.JS and React to develop our back end and front end sections of our system.

The team has many options for our front/back-end solutions, but popular implementations are Next.JS / React

2.2 Decomposition Description

Our object oriented system is broken down into the following classes as shown in the diagram below.

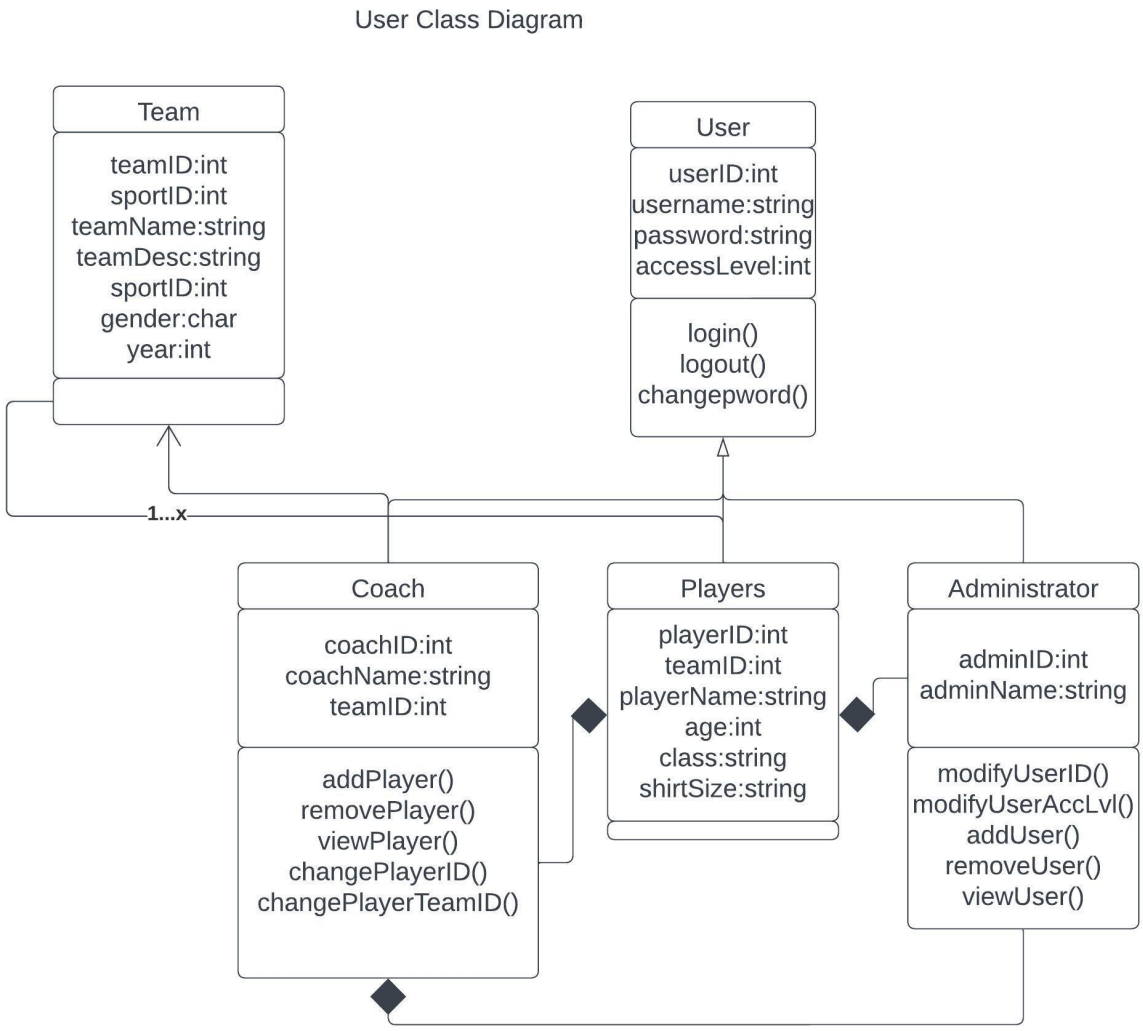

Figure 2. User Class Diagram

This class diagram shows the connection between the user and all of the subclasses that inherit the user. User being the most basic level of access, allowing for login, logout and changing of password, which will just redirect the user to a University of Maine hosted website for password resetting. The next level would be the Players, which can view their own information, as well as change some of their own attributes. Players will be able to view their coach, but not modify anything beyond their own information as allowed. Coaches inherit all of the operations and attributes of the Players, but have access to modify the players more in depth. Administrators are the next and final level of user access, which have complete control over all user accounts, and can modify anything within the scope of the system. Password resets will be hosted by the University of Maine. The Team class will be separate, but both players and coaches will be able to view the team, with coaches being able to modify.

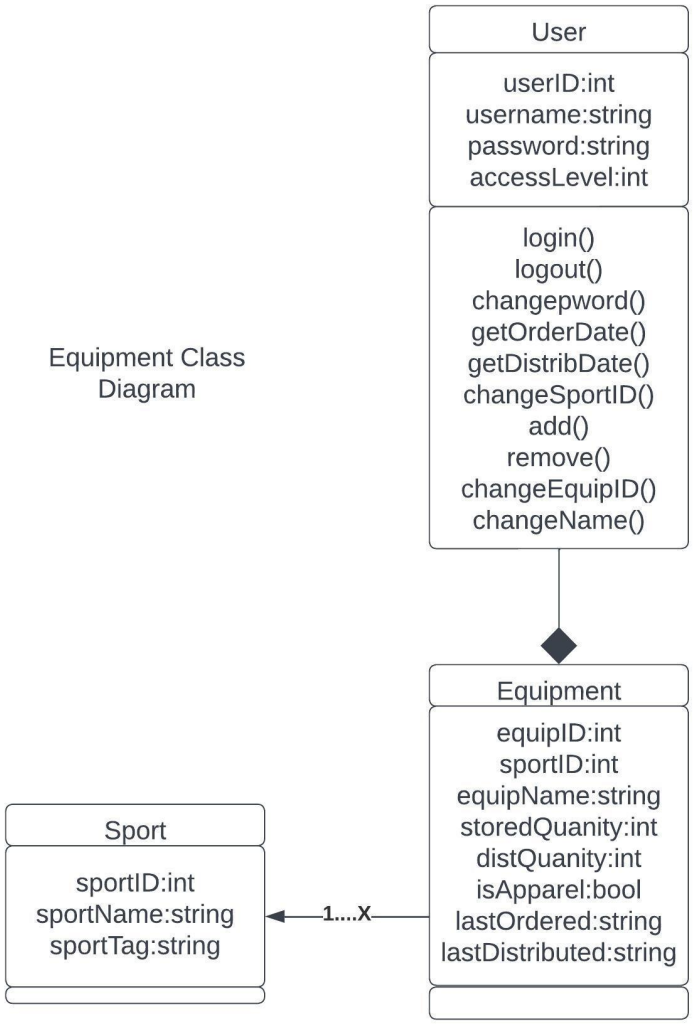

Figure 3. Equipment Class Diagram

This diagram is a little more simple, with all equipment that is listed in the databases having its own modifiers and equipment ID. The connection between the two is that there will be MANY different equipment assignments to each sport. The user is the one who owns the equipment, and is the one who will access and modify the equipment.

2.2.1 Design Pattern

We will be applying the MVC (Model View Controller) design pattern for our product. (diagram pending) We believe this has the most accurate representation of our product in the final stages. A user will interact with the controller through the UI to add, modify or remove users or items. The controller will then communicate with the model, or our database, to request the data that the user is looking for. The model, database, will then send, or deny access to the data requested. The controller will modify the data and prepare it to send to the view

process, which will be what is shown to the user. The user will then see what they have done through the UI again.

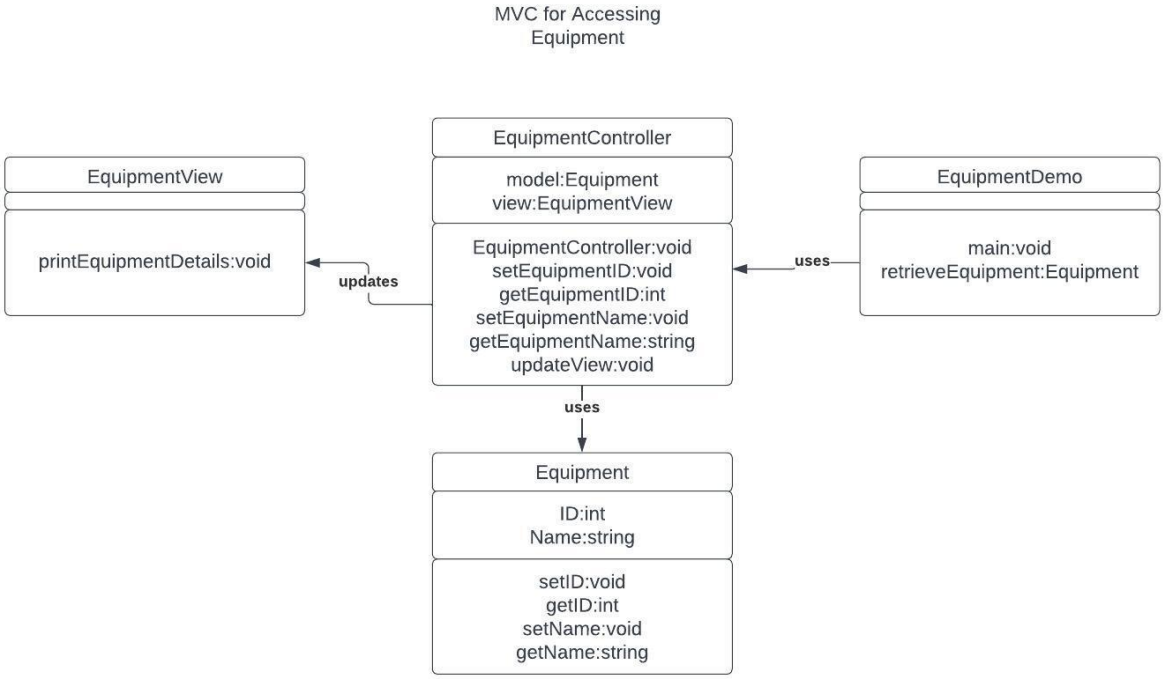

Figure 4. MVC Model for Equipment

3 Persistent Data Design

The database used by the system will be a relational database using MySQL. We will be using a record-based approach that will allow us to establish relationships between the objects in our system. This will promote strong data integrity and avoid data redundancy. Following the MVC design pattern, the database serves as the Model, effectively mapping out the data flow of our application.

There is no file description section in this document because the system will be relying only on the database to keep track of the inventory. This is because the system will need a database to keep track of complex relationships between our objects in the database and the need for querying and indexing of the inventory

3.1 Database Descriptions

The database achieves this through the use of primary and foreign keys. It will allow users to establish one-to-one, one-to-many, or many-to-many relationships between Players, Teams, Sports, and Equipment. The nature of the database will allow us to perform complicated queries and joins that are necessary for tracking and filtering inventory. The database will

contain indexes on fields that are a primary or foreign key, allowing faster data retrieval for commonly used queries. The database schema is shown below using an entity-relationship diagram.

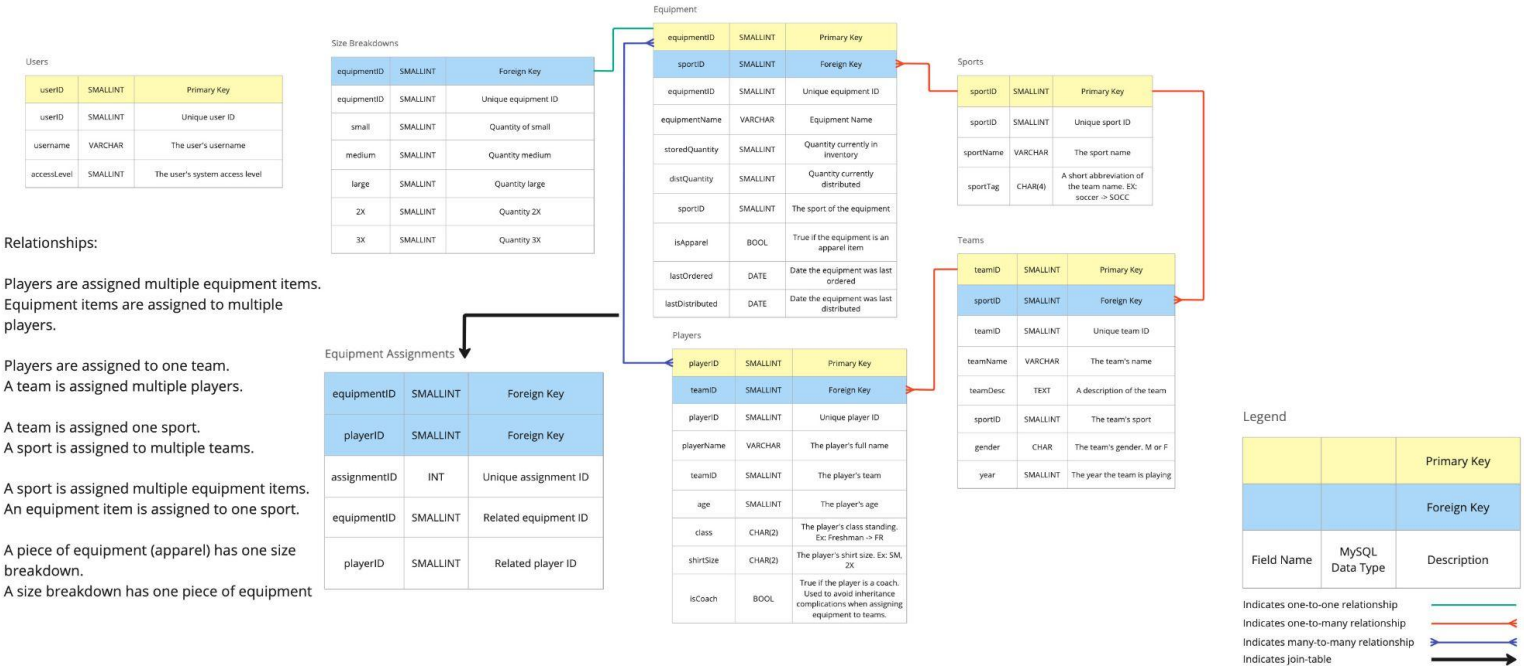

Figure 5. Database Schema

4 Requirements Matrix

The requirements matrix is used to create a framework for what functions and methods that will be used to satisfy the functional requirements for our system. This document demonstrates the relationships between the artifacts and objects and provides an overview of the systems architecture. The requirements matrix not only brings clarity to the development process but also facilitates the actions that will sustain our application. By presenting a structured view of the relationships between the functions, methods, and requirements, this document will act as a pivotal tool between the developers and clients.

4.1 Requirements Matrix Diagrams

This table gives a framework for what functions and methods will be used for our system.

Use Case	Functional Requirement	Source	Satisfied by System Component(s)

Register	FR - 001 : Register	User	UserService.registerUser()
			checkIfExists();
			handleRegisterErrors(errorType);

Log In	FR - 002 : Log In	User	displayLoginForm();
			submitLoginCredentials(username, password);
			validateCredentials(username, password)
			createSession(user);
			handleLoginErrors(errorType);

Log Out	FR - 003 : Log Out	User	logoutUser();
			terminateSession();
			redirectToLogInPage();
			displayLogOutMessage();

Create Team	FR - 004 : Create a team	User/System	createNewTeam(teamID, teamName, teamDesc, sportID, gender, year.);
			
			checkIfTeamExists(teamID);
			If True
			displayTeamExistsMessage(errorType);
			If False
			validateTeamParameters(teamDetails);
			createTeamRecord(teamDetails);
			displayConfirmationMessage();

Create Equipment Item	FR - 005 : Create an equipment item	User/System	createEquipmentItem(equipmentID, equipmentName, storedQuantity, distQuantity, isApparel, lastOrdered, lastDistributed);
			
			checkIfItemExists(equipmentDetails);
			If True
			informItemExists(errorType);
			preventItemCreation();
			If False
			validateEquipmentParameters(equipmentDetails):
			displayConfirmationMessage();

Create New Player	FR - 006 : Create a new player	User/System	selectTeam(teamID);
			inputPlayerDetails(playerID, playerName, teamID, age, class, shirtSize, isCoach):
			checkForDuplicatePlayers(name);
			If True
			numberDuplicatePlayers(name, playerNumber);
			confirmPlayerCreation();
			If False
			confirmPlayerCreation();
			createPlayerRecord(playerDetails);

Assign an Equipment Item	FR- 007 : Assign an Equipment Item to Team	User/System	chooseTeam(teamID);
			enterDesiredAmount(amount);
			
			selectEquipmentItem(equipmentID);
			assignEquipmentToTeam(teamID, equipmentID, desiredAmount);
			
			confirmAssignment();
			checkEquipmentAvailability(equipmentID, desiredAmount);
			displayAvailabilityMessage(availableAmount);

Assign an Equipment Item	FR - 008 : Assign an Equipment Item to Player	User/System	chooseTeam(teamID);
			choosePlayer(name);
			enterDesiredQuantity(quantity);
			
			selectEquipmentItem(equipmentID);
			assignEquipmentToPlayer(name, teamID, equipmentID, desiredQuantity);
			
			confirmAssignment();
			checkEquipmentAvailability(teamID, equipmentID, desiredQuantity);
			displayAvailability(availableQuantity);

Update the details of a previously created equipment item	FR - 009 : Update Equipment Item	User/System	selectEquipmentItem(equipmentID);
			updateEquipmentItem(equipmentID, updatedDetails);
			editItemFields(updatedDetails);
			confirmUpdate();
			
			checkForDuplicateName(updatedDetails, equipmentID);
			updateDatabaseRecord(equipmentID, updatedDetails);

Update details of sports team	FR - 010 : Update Sports Team	User/System	selectTeam(teamID);
			updateTeamDetails(teamID, updatedDetails);
			editTeamFields(updatedDetails);
			confirmUpdate();
			
			checkForDuplicateParameters(updatedDetails, teamID);
			updateDatabaseRecord(teamID, updatedDetails);

Update details of player	FR - 011 : Update Player Info	User/System	selectTeam(teamID);
			selectPlayer(playerID);
			
			updatePlayerInfo(playerID, updatedDetails);
			editPlayerDetails(updatedDetails);
			confirmUpdate();
			updateDatabaseRecord(playerID, updatedDetails);

Remove a Team	FR - 012 : Remove Team	User/System	promptForPassword();
			confirmRemoval();
			removeTeam(teamID);
			updateDatabaseRecords(teamID);

Removing an equipment item from the entire inventory and the teams and players that were assigned the equipment.	FR - 013 : Remove Equipment Item from Inventory	User/System	selectEquipmentItem(equipmentID);
			confirmRemoval();
			removeEquipmentItemFromInventory(equipmentIID);
			unassignEquipmentFromTeams(equipmentID);
			unassignEquipmentFromPlayers(equipmentID);
			displayConfirmationMessage();
			updateDatabaseRecords(equipmentID);

Removing an equipment item from a team to which it was assigned	FR - 014 : Remove equipment item from team	User/System	selectTeam(teamID);
			selectEquipmentItem(equipmentID);
			confirmRemoval();
			removeEquipmentFromTeam(teamID, equipmentID);
			displayConfirmationMessage();
			updateDatabaseRecords(teamID, equipmentID);

Removing an equipment item from a player to which it was assigned	FR - 015 : Remove Equipment Item from Player	User/System	selectTeam(teamID);
			selectPlayer(playerID);
			selectEquipmentItem(equipmentID);
			confirmRemoval();
			removeEquipmentFromPlayer(playerID, equipmentID);
			displayConfirmationMessage();
			updateDatabaseRecords(playerID, equipmentID);

Removing a player from the team they were assigned	FR - 016 : Remove a player from team	User/System	selectTeam(teamID);
			selectPlayer(playerID);
			confirmRemoval();
			removePlayerFromTeam(playerID, teamID);
			displayConfirmationMessage();
			updateDatabaseRecords(playerID, teamID);

The user can view all equipment item details.	FR - 017 : Viewing Equipment Item	User/System	selectEquipmentItem(equipmentID);
			getEquipmentItemDetails(equipmentID);
			displayEquipmentItemDetails(equipmentID);

View players listed under a teams roster	FR - 018 : Viewing a teams roster	User/System	selectTeam(teamID);
			getTeamRoster(teamID);
			displayTeamRoster(rosterData);

Viewing a teams assigned equipment	FR - 019 : Viewing a teams assigned equipment	User/System	selectTeam(teamID);
			getAssignedEquipment(teamID);
			displayAssignedEquipment(equipmentData);

View the details associated with a player	FR - 020 : Viewing a player	User/System	selectTeam(teamID);
			selectPlayer(playerID);
			getPlayerDetails(playerID);
			displayPlayerDetails(playerData);

The user can view all the items in the inventory room	FR - 021 : Viewing general inventory	User/System	displayGeneralInventory();
			retrieveAllEquipmentItems();
			filterInventoryByCategory(category);
			
			Category can either be TeamID or PlayerID

The user can filter the general inventory by name, team equipment, and type	FR - 022 : Filter general inventory	User/System	displayEquipmentInventory();
			applyFilters(filters);
			retrieveFilteredEquipment(filters);
			updateInventoryDisplay(filteredInventory);
			
			Filter can be by name, team equipment, or type

The user can view all previous inventory transactions	FR - 023 : Viewing inventory history	User/System	displayInventoryHistory(); retrieveInventoryTransactions(): filterByActionType(actionType): Action types include create, assign, update, delete

The user can filter the inventory history to search for a specific transaction	FR - 024 : Filtering inventory history	User/System	displayInventoryHistory();
			applyFilters(filters);
			retrieveFilteredHistory(filters);
			updateHistoryDisplay(filteredHistory);
			
			Filters include specific items, teams, players, date range

Figure 6. Requirements Matrix Diagram

Appendix A – Agreement Between Customer and Contractor

Agreement Between Customer and Contractor

---

1. Parties: This agreement made on "11/15/2023" is by and between

Client: University of Maine Athletic Department

AND

Contractor: Inventory Management Software Group

2. Term: The terms of this agreement shall commence on November 5th, 2023, and conclude on May, 2024.

3. Services: The Contractor agrees to provide the following services for the betterment of the Customer: The Contractor will create an inventory management system that allows the Customer to visualize and manage their inventory through an online webpage. Further details are presented in the System Design Document.

4. Expenses: There are no initial expenses. When expenses are incurred, the Contractor will communicate and get approval from the Customer to use funds allotted to them if available.

5. Agreement:
By signing this document, all parties agree to the requirements presented in this document. All parties also agree that the deadlines presented are tentative, and are subject to change as the program is developed.

Customer Signature:

	Date:	Printed:
X: *Kevin Ritz*	11/15/2023	Kevin Ritz
Contractor Signature:	Date:	Printed:
X: *Collin Rodrigue*	11/15/2023	Collin Rodrigue
X: *Gabriel A. Poulin*	11/15/2023	Gabriel A. Poulin
X: *Brennan Poitras*	11/15/2023	Brennan Poitras
X: *Sean Radel*	11/15/2023	Sean Radel
X:*Graham Bridges*	11/15/2023	Graham Bridges

Appendix B – Team Review Sign-off

Team Agreement Sign Off

Team IMSG has thoroughly reviewed the System Design Document for the Athletic Inventory System and has agreed that the following information is accurate and achievable. Collectively we have no major contentions in the information stated in the document. By signing this agreement, one acknowledges all the terms and conditions outlined in the document and understands the importance of effective team collaboration, communication, and shared accountability when achieving the goals of the project. By signing below, we pledge our dedication to the success of the team and the project we plan to undertake. We agree to work collaboratively, and support each other to uphold the guidelines and expectations set forth in the agreement.

Signature:	Date:	Printed:
X: *Collin Rodrigue*	11/15/2023	Collin Rodrigue
X: *Gabriel A. Poulin*	11/15/2023	Gabriel A. Poulin
X: *Brennan Poitras*	11/15/2023	Brennan Poitras
X: *Sean Radel*	11/15/2023	Sean Radel
X:*Graham Bridges*	11/15/2023	Graham Bridges

Appendix C – Document Contributions

Name	Date	Contribution	Version
Sean Radel	11/2/23	Formatting, introduction	0.1
Sean Radel	11/3/23	Architectural Design Section	0.1
Brennan Poitras	11/6/23	Database Description/Schema	0.1
Collin Rodrigue	11/6/23	Requirements Matrix	0.1
Gabriel Poulin	11/6/23	UML Diagrams and Design Pattern	0.1
Sean Radel	11/14/23	Fixed Architectural Design Section and formatting	0.2
Gabriel Poulin	11/14/23	Appendix A and Appendix B	0.2
Brennan Poitras	11/14/23	Additions to the DB schema	0.2