COS 397: Computer Science Capstone I

System Design Document

UMaine Athletic Department Inventory Management System

Version 0.1

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

2 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.

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. Athletic Department managed Google Drive containing all inventory tracking spreadsheets:https://drive.google.com/drive/folders/1dclEKCke2CdXU3GNA5ee5VpROfOWX25w
2. IMSG System Requirements Specification: https://docs.google.com/document/d/1LnOj2DEyu8DPbKXBTDBm2y6UbePr_AXC/edit
3. https://firebase.google.com/docs/auth
4. https://nextjs.org/
5. https://react.dev/
6. https://blogs.sap.com/2008/01/09/how-to-communicate-architecture-technical-architecture-modeling-at-sap-part-1/
7. Information on relational databases. https://help.claris.com/en/pro-help/content/related-tables-files.html
8. MySQL Data Types: https://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

https://miro.com/app/board/uXjVNSkWp2A=/?share_link_id=443092433719

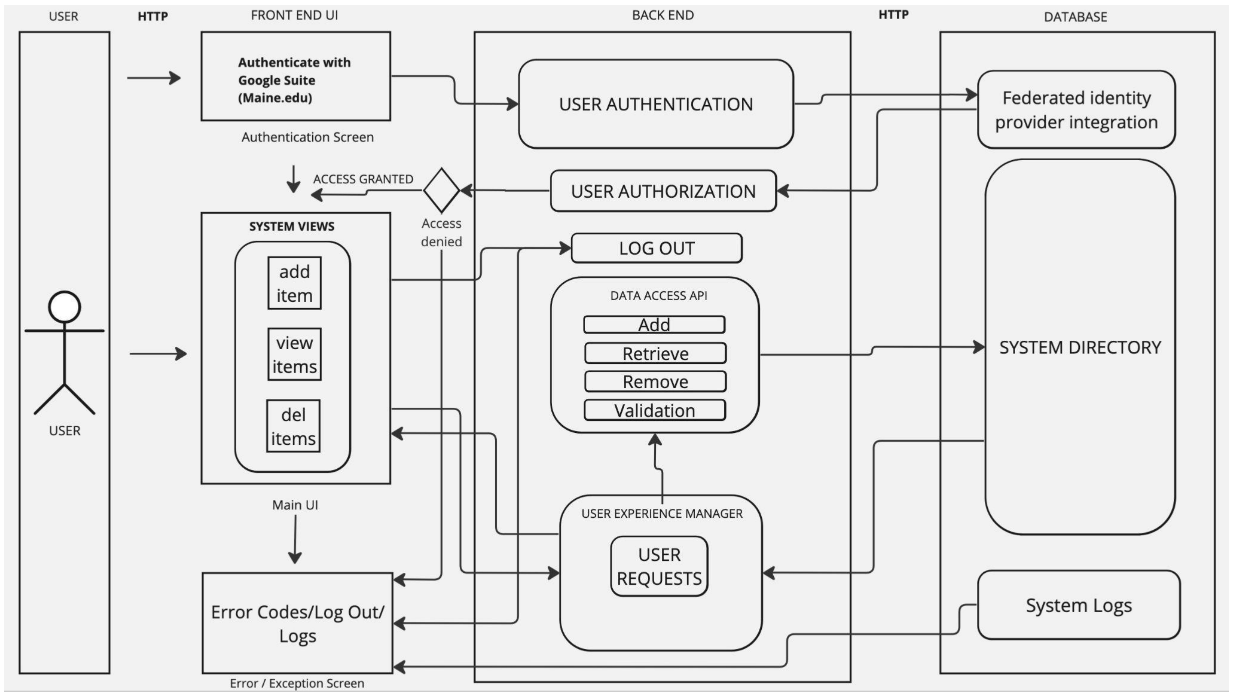


Figure 1. Architectural Design Diagram

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

Illustrate and describe the decomposition into components of the system that you presented in Section 2.1 (*e.g.*, functions, objects, scripts, files). This is the view of the system as you know it at this point in time. Provide diagrams as follows:

Object-oriented – If your system is object-oriented, use class diagrams to illustrate the implementation design of your system. Include any attributes and methods (public, private, and protected, constructors, accessors, and mutators) that are known as of this time. Use the suggested UML class diagrams reference at the end of this document where appropriate.

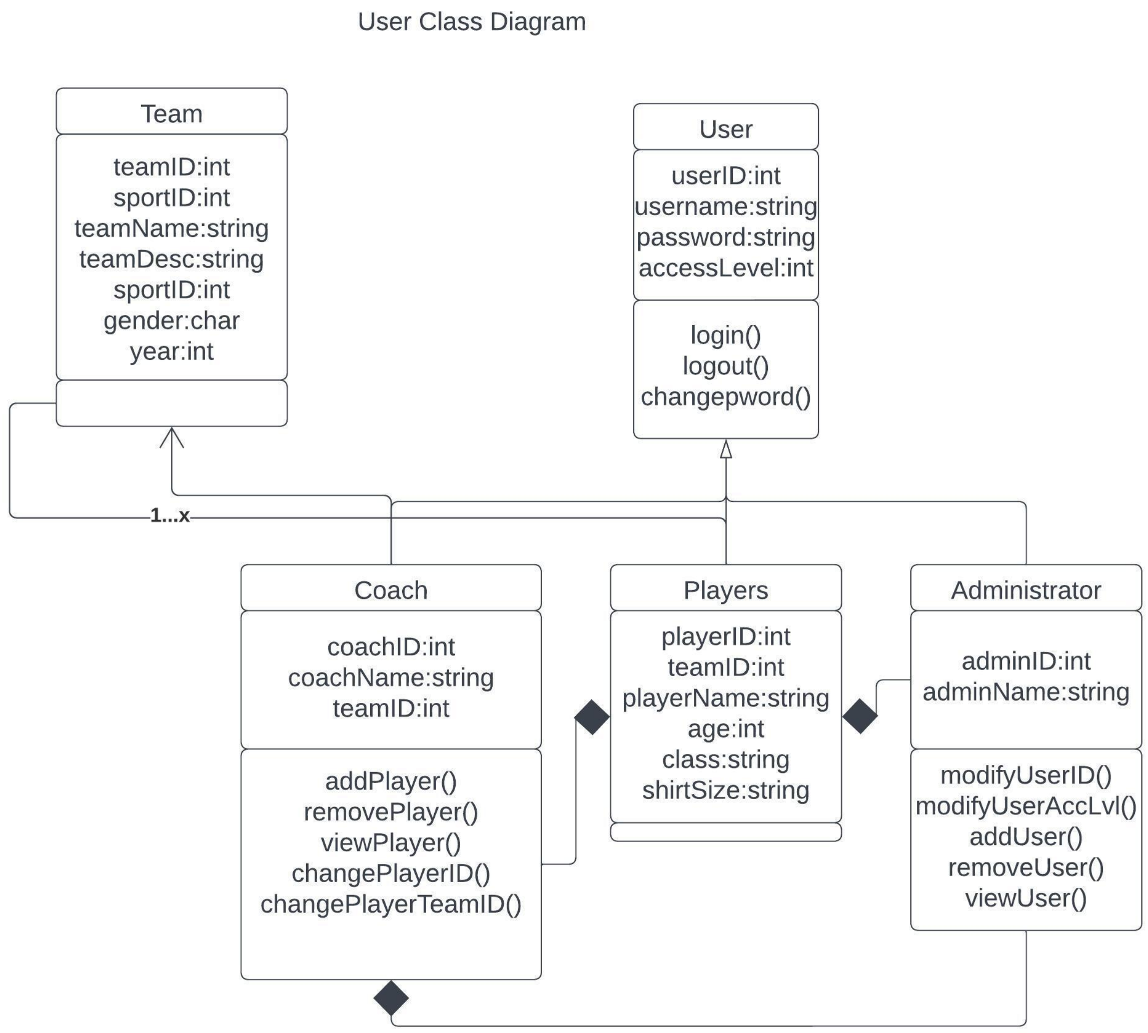


Figure 2. User Class Diagram

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 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 that needs modifying WITHIN the application. Password resets will be hosted by the University. The Team class will be separate, but both players and coaches will be able to view the team, with coaches being able to modify.

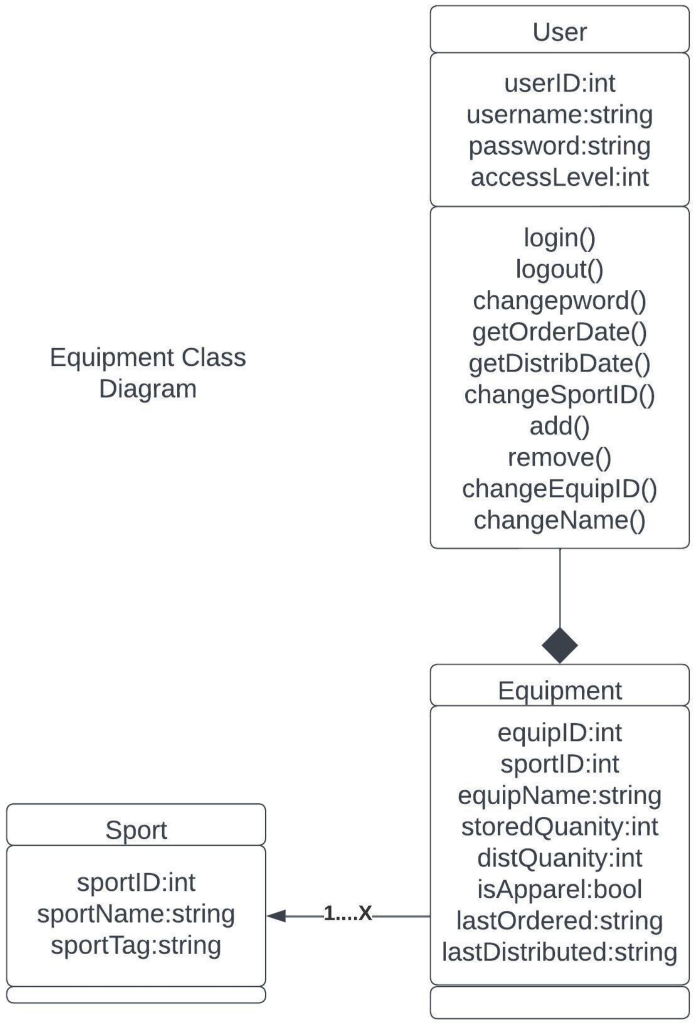


Figure 3. Equipment Class Diagram

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.

2.2.1 Design Pattern

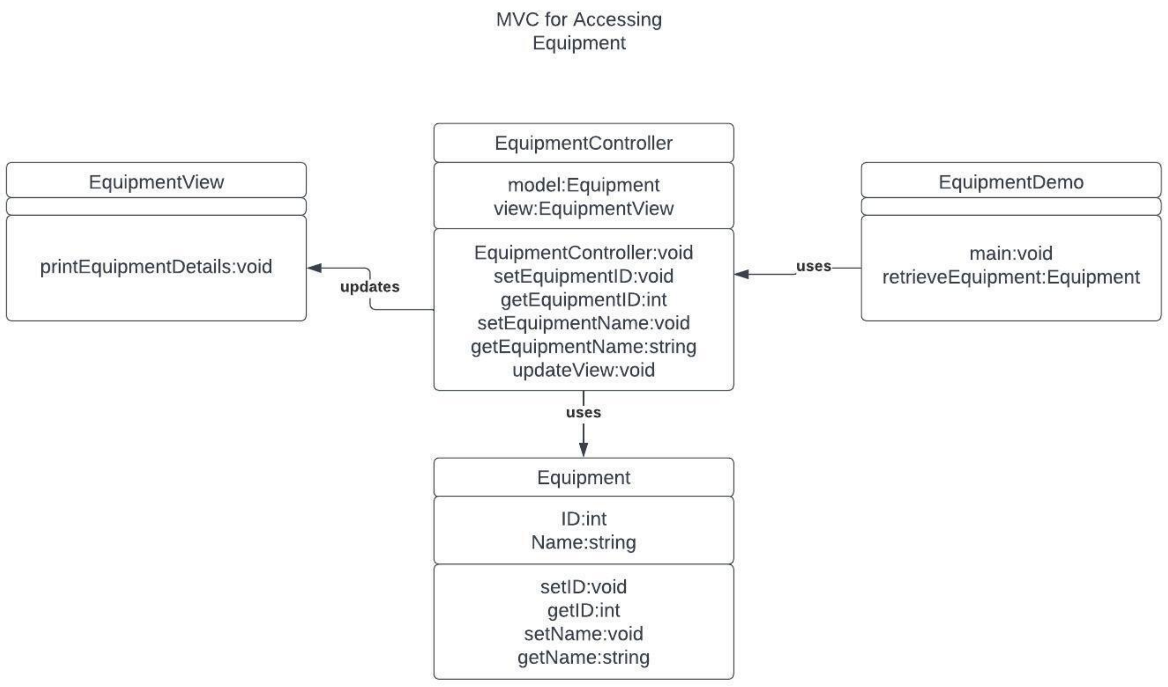


Figure 4. MVC Model for Equipment

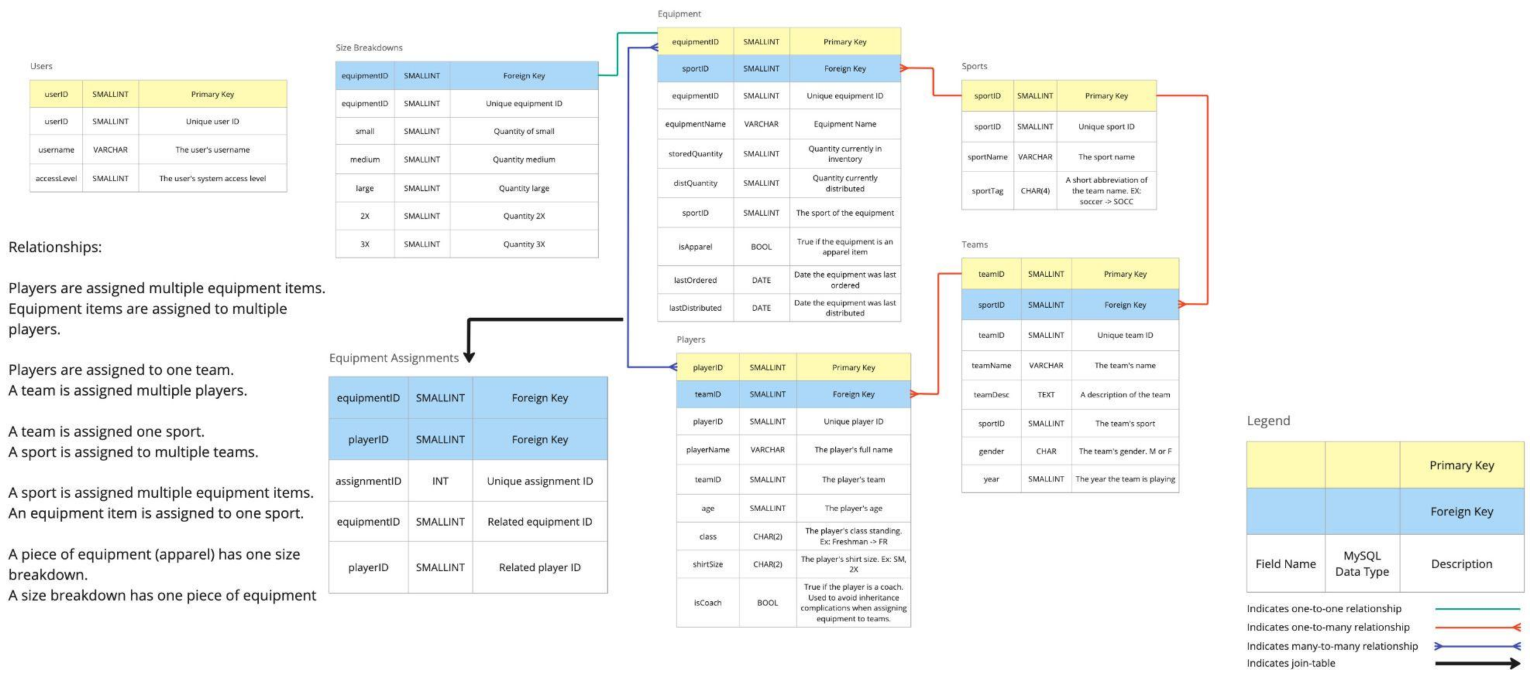
We will be applying the MVC (Model View Controller) design pattern for our product. (diagram pending) We believe this has the most accurate representation of our product in the final stages. A user will interact with the controller through the UI to add, modify or remove users or items. The controller 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.

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.

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.



Legend
Field Name
MySQL Data Type
Description
Primary Key
Foreign Key
Indicates one-to-one relationship
Indicates one-to-many relationship
Indicates many-to-many relationship
Indicates join-table

Users

userID	SMALLINT	Primary Key
userID	SMALLINT	Unique user ID
username	VARCHAR	The user's username
accessLevel	SMALLINT	The user's system access level

Size Breakdowns

equipmentID	SMALLINT	Foreign Key
equipmentID	SMALLINT	Unique equipment ID
small	SMALLINT	Quantity of small
medium	SMALLINT	Quantity medium
large	SMALLINT	Quantity large
2X	SMALLINT	Quantity 2X
3X	SMALLINT	Quantity 3X

Equipment

equipmentID	SMALLINT	Primary Key
sportID	SMALLINT	Foreign Key
equipmentID	SMALLINT	Unique equipment ID
equipmentName	VARCHAR	Equipment Name
storedQuantity	SMALLINT	Quantity currently in inventory
distQuantity	SMALLINT	Quantity currently distributed
sportID	SMALLINT	The sport of the equipment
isApparel	BOOL	True if the equipment is an apparel item
lastOrdered	DATE	Date the equipment was last ordered
lastDistributed	DATE	Date the equipment was last distributed

Sports

sportID	SMALLINT	Primary Key
sportID	SMALLINT	Unique sport ID
sportName	VARCHAR	The sport name
sportTag	CHAR(4)	A short abbreviation of the team name. EX: soccer -> SOCC

Teams

teamID	SMALLINT	Primary Key
sportID	SMALLINT	Foreign Key
teamID	SMALLINT	Unique team ID
teamName	VARCHAR	The team's name
teamDesc	TEXT	A description of the team
sportID	SMALLINT	The team's sport
gender	CHAR	The team's gender. M or F
year	SMALLINT	The year the team is playing

Players

playerID	SMALLINT	Primary Key
teamID	SMALLINT	Foreign Key
playerID	SMALLINT	Unique player ID
playerName	VARCHAR	The player's full name
teamID	SMALLINT	The player's team
age	SMALLINT	The player's age
class	CHAR(2)	The player's class standing. Ex: Freshman -> FR
shirtSize	CHAR(2)	The player's shirt size. Ex: SM, 2X
isCoach	BOOL	True if the player is a coach. Used to avoid inheritance complications when assigning equipment to teams.

Equipment Assignments

equipmentID	SMALLINT	Foreign Key
playerID	SMALLINT	Foreign Key
assignmentID	INT	Unique assignment ID
equipmentID	SMALLINT	Related equipment ID
playerID	SMALLINT	Related player ID

Relationships:

Players are assigned multiple equipment items.
Equipment items are assigned to multiple players.

Players are assigned to one team.
A team is assigned multiple players.

A team is assigned one sport.
A sport is assigned to multiple teams.

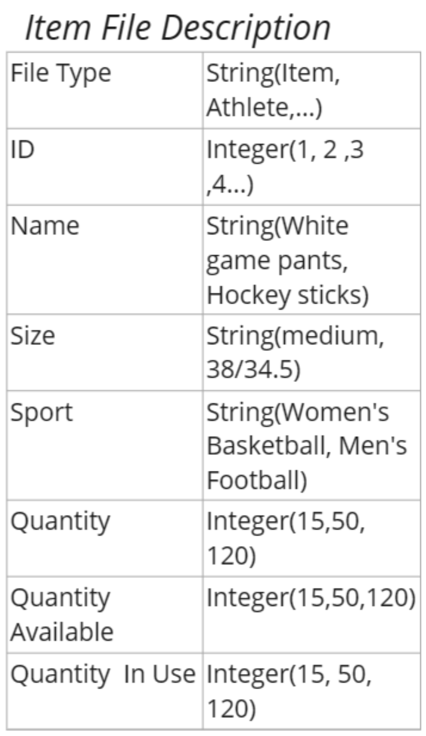
A sport is assigned multiple equipment items.
An equipment item is assigned to one sport.

A piece of equipment (apparel) has one size breakdown.
A size breakdown has one piece of equipment

Figure ???: Database Schema

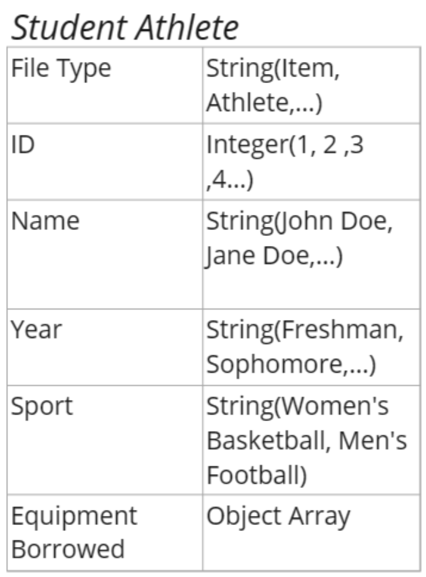
3.2 File Descriptions

Describe the file(s) used by the system. Include a diagram of the file structure. For each field in a file, give its name, data type (*e.g.*, int, double), size (*e.g.*, strings), and description of what it represents. Basically, give all of the information that a programmer must have to implement the file. If no files are used, simply state so. Supplement your description with a sample file(s). [Length is whatever it takes]



Item File Description

File Type	String(Item, Athlete,...)
ID	Integer(1, 2 ,3 ,4...)
Name	String(White game pants, Hockey sticks)
Size	String(medium, 38/34.5)
Sport	String(Women's Basketball, Men's Football)
Quantity	Integer(15,50, 120)
Quantity Available	Integer(15,50,120)
Quantity In Use	Integer(15, 50, 120)



Student Athlete

File Type	String(Item, Athlete,...)
ID	Integer(1, 2 ,3 ,4...)
Name	String(John Doe, Jane Doe,...)
Year	String(Freshman, Sophomore,...)
Sport	String(Women's Basketball, Men's Football)
Equipment Borrowed	Object Array

Above are diagrams of the structure of the files containing information on items in the inventory and student athletes. The item file will contain a file type telling the system that is an item file, a unique ID for the item, a name for the item, a size for the item which can be categorical or quantitative, a sport the item is associated with, the total quantity of the item obtained, the quantity of the item available for players, and the quantity currently available. The Student Athlete file contains a file type telling the system it is a Student Athlete File, a unique ID for the athlete, the athlete's name, the college year of the athlete, the sport the student athlete participates in, and the equipment the individual athlete is using. A definite programming language has not been chosen for this project yet so we will assume files are js based.

4 Requirements Matrix

Remember, every major section should have a section introduction.

4.1 Requirements Matrix Diagrams

https://docs.google.com/document/d/1nuFaMVKTzo3QIPwzl_FUahxrW1HdYcSa6JM5UPZa5OY/edit

Appendix A – Agreement Between Customer and Contractor

https://docs.google.com/document/d/1pIBvuRZvminvUjNsrFE4vpi2wPDUT0X2qUjera0f8rc/edit

Appendix B – Team Review Sign-off

https://docs.google.com/document/d/1aolYaI6lMkqoAG7fTMEgXDBIqEvvyQWsJ3o6D69pYg4/edit

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
Graham Bridges	11/6/23	File Descriptions	0.1
Gabriel Poulin	11/6/23	UML Diagrams and Design Pattern	0.1
Sean Radel	11/		
Gabriel Poulin	11/15/23	Appendix A and Appendix B	
Brennan Poitras	11/14/23	Additions to the DB schema	
