**Yahtzee**

**Group Project**

CIS-17B

Danielle F, Logan O,

Ismael P, Amir J

Date: 04/24/23

# **Table of Context**

Introduction………………………………………………………………….2

Summary………………………………………………………….………….2

Description……………………………………………………….…………..2

GitHub Link………………………………………………….………………2

UML Chart…………………………………………………………………...3

Flow Chart……………………………………………………………………4

Gantt Chart……………………………………………………………...……5

Yahtzee\_Dice\_V1…………………………………………………………….6

Yahtzee\_ScoreCard\_V2…………………………………………………..…..6

Yahtzee\_ScoreCard+Dice\_V3……………………………………….…….…6

Yahtzee\_Game\_V4……………………………………………………….…..7

Yahtzee\_Game\_V4.1………………………………………………………....7

Yahtzee\_Game\_V4.2…………………………………………………..……..7

Yahtzee\_Game\_V4.3…………………………………………………….…...7

Yahtzee User/Admin Versions Timeline Table ……………….…………..…8

Summary of Pivotal User/Admin Versions ………………………….…….…9

Reference ...…………………………………………………………………11

Pseudo Code ...……………………………………………………………...11

# **Introduction**

Title: Yahtzee

Choose a starting player by any method (oldest player, youngest player, highest roll of the dice, etc.)

Beginning with the starting player, players will take turns one at a time in clockwise order. The game consists of thirteen rounds, and at the end of the thirteenth round, the game will end. (All the categories on the players’ scorecards will be completely filled in at that point.)The dice can be scored after any of the rolls, but scoring the dice ends the player’s turn. Setting dice aside after one roll does not prevent one or more of them from being rolled again on any subsequent roll if the player so chooses.

Each player’s goal is to try and score as high as possible in one of the thirteen categories shown on their scorecard.

To score the dice, the player selects one of the categories on their scorecard and writes the score into it. Each category can be scored only once per game (except for the Yahtzee category). Categories can be filled in any order. A player must score the dice on their turn even if it turns out that there are no good categories remaining to score in. Once a category is filled it may not be changed.

A player may write a score of zero in any category if they have rolled no point-generating results or simply choose to do so. After marking their score on their scorecard, the player’s turn ends, and play proceeds to the player on their left.

**Summary**

Project Size: ~2485 lines

The entire User and Admin class was written by Danielle with around 28 versions of the code. In the final version of the code, the User class allows you to sign in and saves your high score to a binary file. While the Admin class gives the admin the ability to rewrite a specific user's high score, allows the admin to see all profiles created, can pick a specific profile to look at, etc. Her program also allows the player to play as a guest if they didn’t want to sign in. The Yahtzee portion of the project was written by Ismael with some help from Danielle to fix some bugs and clean up the code to make it a little more readable. The Yahtzee program consists of two helper classes, the scorecard class, and the dice class. When first starting the game it rolls 5 random dice and allows the user to pick a category from the scorecard if they wish. If not, it allows the user to keep or reroll specific dice in order to get the desired outcome for the scorecard.

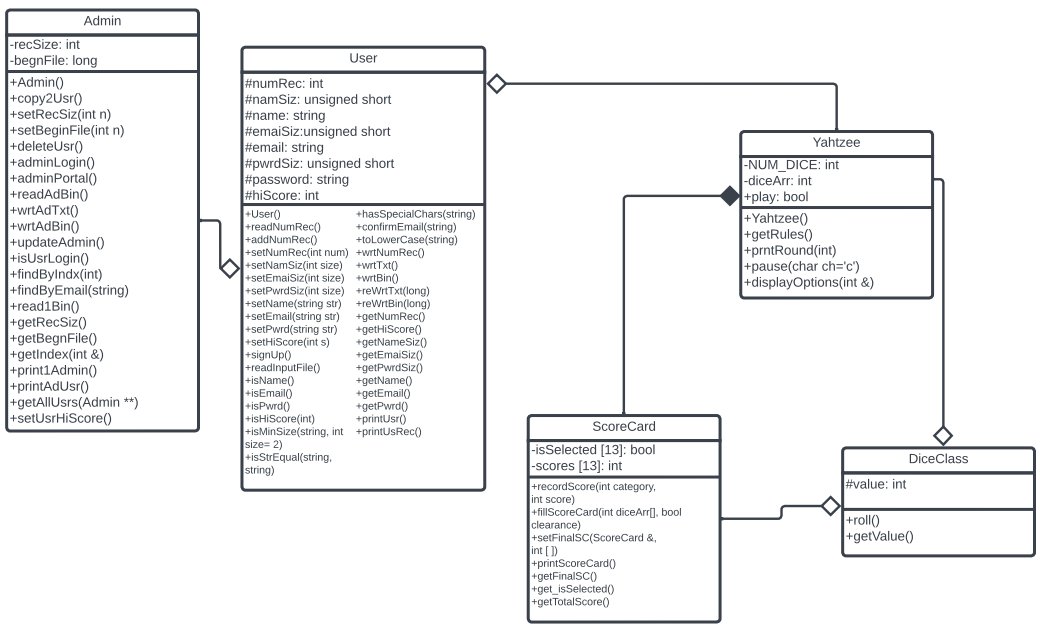
# **Description**

This is a Yahtzee Program, it allows you to play Yahtzee by yourself or play against a friend. Whoever gets the higher score wins, and if the person logged in gets a new personal high score, that score will be saved to a binary file.

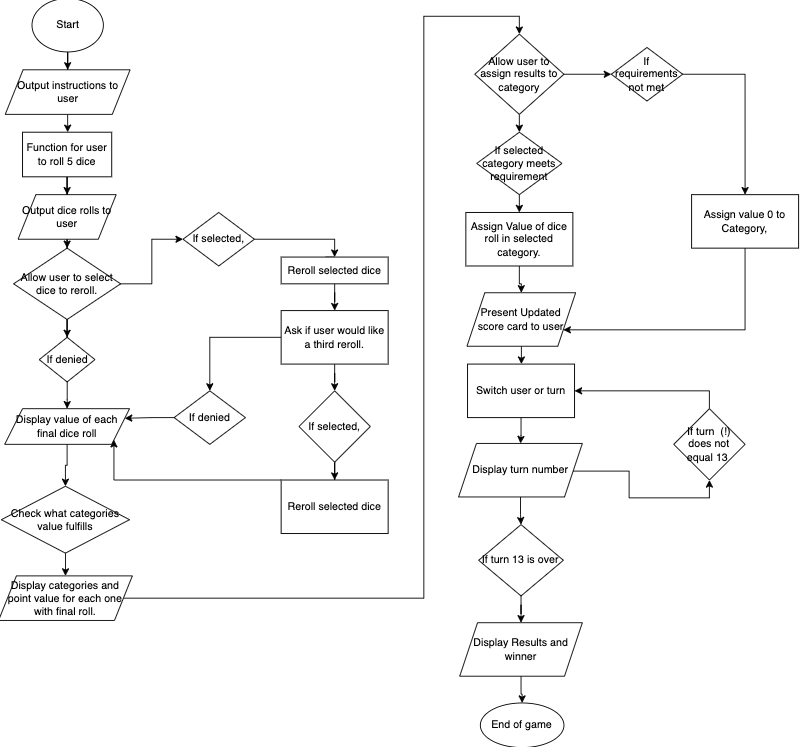
# **GitHub Links**

* Danielle’s Github: <https://github.com/koa2019/yahtzee>
* Ismael’s Github: <https://github.com/Error1417/Yahtzee_Project>

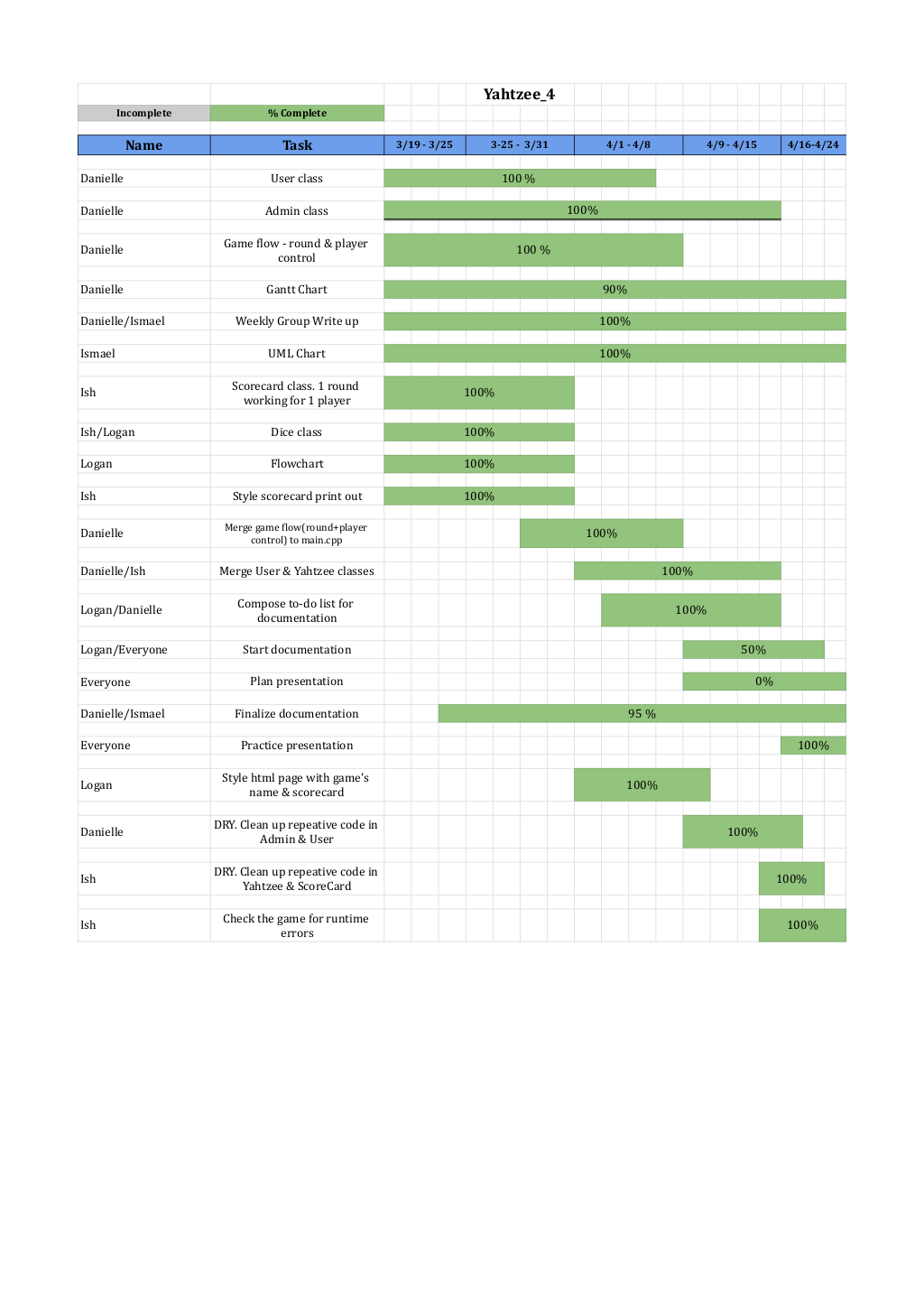
# **UML Chart**

****

## **FlowChart**



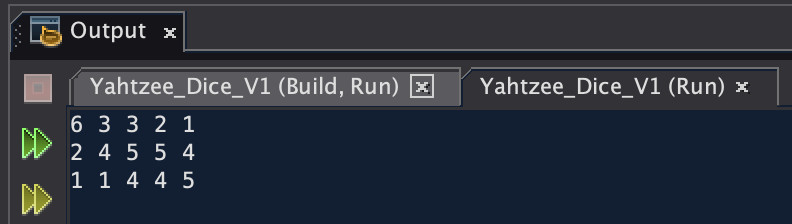
**Gantt Chart**

****

**Yahtzee Class Summaries**

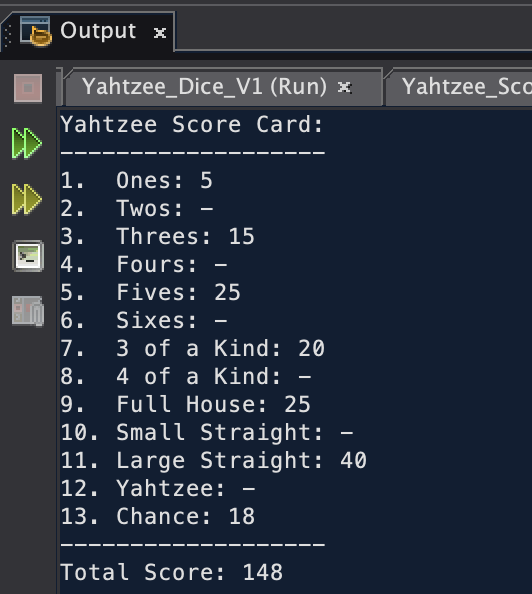
[**Yahtzee\_Dice\_V1**](https://github.com/Error1417/Yahtzee_Project/tree/main/Yahtzee_Dice)

-In this version of the Yahtzee game, I create a Dice class that creates 5 random dice and shows them to the user. I output these 5 random dice three times to test their randomness.



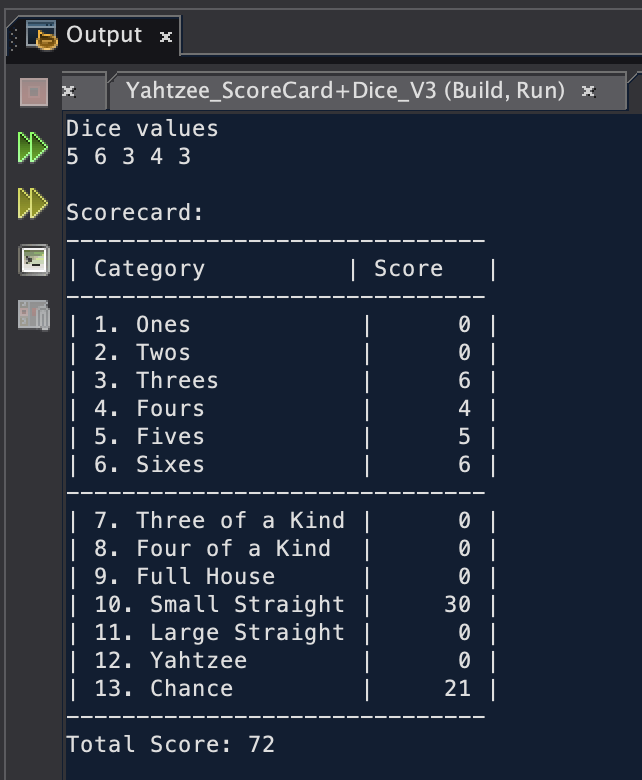
[**Yahtzee\_ScoreCard\_V2**](https://github.com/Error1417/Yahtzee_Project/tree/main/Yahtzee_ScoreCard)

-The next part of making this game for me was creating the scorecard class on which the values of the dice would be displayed. This class created a scorecard and for the moment only outputs random numbers into the places of the categories.

****

[**Yahtzee\_ScoreCard+Dice\_V3**](https://github.com/Error1417/Yahtzee_Project/tree/main/Yahtzee_ScoreCard%2BDice)

-In this version of the game, I combined the ScoreCard and Dice class in order to create an output to the scorecard that wasn’t just random. The scorecard takes in the random values of the dice and tests them to see if they fulfill the requirements for any of the categories.

****

[**Yahtzee\_Game\_V4**](https://github.com/Error1417/Yahtzee_Project/tree/main/Yahtzee_Game_V4)

-This version of the game was when I added the Yahtzee class, and it started looking more like a playable game. I added the ability to pick which dice you wanted to keep and which dice you wanted to reroll. I also added the ability to pick a category, but that feature wasn’t working yet. There are some bugs in this version of the code such as not being able to enter letters when the input asks for a number. This would cause an infinite loop. Also, this version allowed the user to break the game by having more than 5 dice. I can’t show the output of this one or any other version from here on out because it would be too large.

[**Yahtzee\_Game\_V4.1**](https://github.com/Error1417/Yahtzee_Project/tree/main/Yahtzee_Game_V4.1)

-This version of the game moves around some of the outputs of the program in hopes to make it like the real game. The only problem is that I was outputting the options for the game before actually rolling the dice for the user, so the user was allowed to pick a category before the dice were rolled. This version also added input verification.

[**Yahtzee\_Game\_V4.2**](https://github.com/Error1417/Yahtzee_Project/tree/main/Yahtzee_Game_V4.2)

-This version was more focused on fixing the selection system for the category, and fixing a bug I was having with the parameters of the fillscorecard function. I decided to move the category selection system into this function because it fixed the problem I was having with the parameter problems. In order to fix it, I did need to pass in a bool that said if the player was allowed to pick a category or not.

[**Yahtzee\_Game\_V4.3**](https://github.com/Error1417/Yahtzee_Project/tree/main/Yahtzee_Game_V4.3)

-This last version of my code was mainly for me to test out what would work when trying to save the category a user had chosen. My attempts were unsuccessful, but thankfully my partner Danielle was able to fix the problems with the category selection system and was able to make the code look and run much cleaner.

# 

# **Yahtzee Version Timeline Table**

# 

# **Summary of Pivotal User/Admin Versions**

[yahtzee\_v5\_userClass\_read3rdBinaryRec](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v5_userClass_read3rdBinaryRec):

I created a User class. The program can read in three strings (name, email, password) and validate them. Each string is validated before it is saved to an instance of User class. Then it writes the User object’s values to a text file as well as a binary file. The program opens the User’s binary file and looks for the third record. Once it locates the record, it reads the data in and saves it to a User object.

[yahtzee\_v6\_userClass\_readRandBinaryRec](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v6_userClass_readRandBinaryRec) :

* Added int hiScore=0 to User class.
  + Added getHiScore(), setHiScore(int score).
  + Wrote hiScore to text and binary files.
  + Confirmed hiScore is being read from binary file.
* Read 1 record from User’s binary file and saved it to an instance of User class.

[yahtzee\_v11\_adminClass](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v11_adminClass) :

Added Admin class & moved all of User class to it, so the User class inherits the Admin class.

[yahtzee\_v12\_adminClass\_staticMmbr](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v12_adminClass_staticMmbr) :

* Adding static int ttlRecs and static member() to Admin class and set its value by reading a text file each time one of Admin’s constructors were called. This allowed ttlRec to hold its value in between runs.
  + Added ttlRec to wrtTxt(), wrtBin(), readBin().
* Moved signUp() & readInputFile() to User class.
* Created adminPortal() with a menu inside of the admin.cpp. You can only access this menu if Admin is signed in.
* Add max retries in user validation functions
* Added overloaded readBin() that accepts an email.
* Added wrt1Record() and called it in both readBin()
* Added wrtAdminTxt() & wrtAdminBin() as private Admin members
* Added char arguments to wrtTxt()
* Added a switch menu in main() with options to sign up, user login, admin login.
* Fixed User overwriting ttlRec & ID by passing the values from Admin to User in main().
* Changed bool readBin() to long findEmail(), so I can pass the cursor when I need to rewrite a file in binary. If email is not found in binary, then it sets the cursor to -99.

[yahtzee\_v14\_adminClass](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v14_adminClass) :

Added cis17A\_project2\_yahtzee\_v8’s Player class to main(). Player::yahtzee() runs for 13 rounds and play() handles each player’s turn.

[yahtzee\_v18\_admin\_aggregateUser](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v18_admin_aggregateUser) :

Admin aggregate an instance of User instead of inheriting it. My group thought the Admin class should aggregate an instance of User class instead of using inheritance. Aggregating requires a lot more code than inheritance.

[yahtzee\_v19\_addYahtzeeClass](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v19_addYahtzeeClass) :

* Added Ismael's Yahtzee, Score\_Card, and Dice classes to main().
* Added setUsrHiScore(score) & called it main(), so Admin can rewrite the User's binary and text files.

[yahtzee\_v20\_yahtzee](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v20_yahtzee) :

* Moved wrtTxt(), wrtBin(), reWrtBin() & reWrtTxt() to User.cpp and passed beginFile to them.
* In main(), I made User rewrite their hiScore directly instead of having admin do it.
* Created an array of Users, so the game can run with 2 players.
  + Altered Yahtzee::play(), so it accepts a double pointer of Users.
  + Play() also returns if player 1 is the winner or not.

[ish\_df\_yahtzee\_v4.6](https://github.com/koa2019/yahtzee/tree/main/ish_df_yahtzee_v4.6) :

* I reorganized and cleaned up the flow of play() by creating prntRound(), rules(), setFinalSC().
* Created a 2nd instance of ScoreCard to hold the player's actual scores.
* Added a conditional if numRolls==3, then force switch case 2 else displayOptions();
* Added a pause(), so it makes the game more readable between rolls
* Created bool isSelected[13] to prevent already selected categories from being displayed in possible points scoreboard. Called it inside of recordScore().
* Reset isSelected() flags in setFinalSC() to stop categories from being overwritten.
* Changed play() to bool play(int &userHiScore).
  + If player 1 is winner AND their current score is larger than user’s high score, then
    - Reassign userHiScore's value and return true.

[yahtzee\_v22\_updateYahtzeeWithIsmaels](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v22_updateYahtzeeWithIsmaels) :

Added Ismael's updated classes from ismael\_yahtzee\_v4.6

[yahtzee\_v23.1\_addPlayrArray](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v23.1_addPlayrArray)

* Aggregated 2 instances of ScoreCard in Yahtzee's private members. Made them static arrays.
* Moved \*\*players Yahtzee's default constructor. This way I can access player[ ] in other Yahtzee functions.
  + Added for( pIndx<nPlayer) to control each player's turn
  + Reset values inside switch() default so it could jump out of all the loops and print final scorecards.
* Added getUpLowSums() to calculate the totals from the scorecard’s upper and lower sections.
* Created a specific function to print the player’s actual scorecard rather than using the function that prints their possible points.

[yahtzee\_v24\_startGameFunc](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v24_startGameFunc) :

Cleaned up play() by creating startGame() to handle the number of rounds loop

and play() handles each player's turn individually.

[yahtzee\_v26\_moveDice2ScoreCard](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v26_moveDice2ScoreCard):

* Moved the creation of Dice dice[ ] and int diceArr[ ] and their code from Yahtzee to ScoreCard.
* Cleaned up play() by adding selectDice(), selectCategory(), and isNewHiScore().
* Moved constant variables to the top of Classes h file

[yahtzee\_v27\_adminDoublePtrArray](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v27_adminDoublePtrArray):

I eliminated redundant code by having Admin::getAllUsr() be the only function that reads the User’s binary file. Now getAllUsr() returns a double pointer array with all the user's records read from the binary file. Initially findByEmail(), findByIndx(), and getAllUsr() read the User’s binary file.

[yahtzee\_v28.1\_doublePtrAdminConstructor](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v28.1_doublePtrAdminConstructor):

Added Admin \*\*usrArr as a private member & allocated memory for it in Admin’s default constructor. Now there's an array Admin can reference anywhere in its class.

[yahtzee\_v28.2\_changeFindByFuncs](https://github.com/koa2019/yahtzee/tree/main/yahtzee_v28.2_changeFindByFuncs):

* findByEmail() and findByIndx() can search through an array instead of searching a binary file.
* startGame() passes record's location to Yahtzee, so it can rewrite the user's record inside of it rather than in main().
* Cleaned up adminPortal() by putting each case in its own function.
* Updated Admin with survey\_v5's Admin class

# **References**

1. Gaddis, Tony. Starting out with C++: From Control Structures through Objects. 9th ed., Pearson, 2018.
2. Lehr, Mark. “2023\_Spring\_CSC\_CIS\_17B · ml1150258/2023\_spring\_csc\_cis\_17b.” GitHub, 2023,<https://github.com/ml1150258/2023_Spring_CIS_CSC_17B>.

# **Pseudo Code**

1. **Parent: Admin class**
   1. Dilemma. Option 1: User class will be the parent and then the Admin and Player classes will inherit User?
      1. Admin & Player both use User class’s variables & functions, but how will the Player ask Admin to test their hiScore variable?
   2. Dilemma. Option 2: Admin is the parent, list wrtBinary() as private and then User inherits Admin?
      1. Admin doesn’t need hiScore when it creates a new object for itself, but it needs to be able to reassign hiScore.
   3. Admin object:
      1. Private members:
         1. Static int num records /\* Doesn’t hold value between runs \*/
         2. unique ID
         3. name
         4. password
         5. email
         6. hiScore ? Should this be in User or Admin?
         7. readBinary()
         8. checkHiScore()
         9. setHiScore()
   4. Read inputs for 1 Admin record:
      1. Confirm inputs
      2. Save 1 admin record to object
      3. Write the object to text and binary files.
      4. Print message if successful or not
   5. ONLY Admin is allowed to read binary files
   6. Admin login()
      1. Read first record in Admin’s binary file
      2. Save it to an object
      3. Test login
      4. Print message if login was correct or not
      5. If login is correct, then allow them access to admin only functions
   7. Read User’s binary file:
      1. User login()
         1. Find a record by email
         2. Save it to an object
         3. Test login
         4. Print message if login was correct or not
         5. If login is correct, then allow them to view their profile and play game
      2. Be able to delete/edit a record or member’s of record
         1. Accept an ID or record as an object AND score from User, test if it’s bigger than their current hiSCore and update it accordingly
2. **Child: User class**
   1. Create User Class. User and game classes will have to be combined at some point.
   2. User object:
      1. Private members:
         1. Record #
         2. Unique ID
         3. name
         4. password
         5. email
         6. hiScore ? Should this be in User or Admin?
         7. scorecard[ ]
         8. dice array or vector
   3. Sign up for a new account. Read in name, email, password
      1. Confirm user inputs before saving their info to object
   4. Write and append each User to a binary file
      1. ONLY has permission to write to binary file
   5. Write and append each User’s to text file
      1. ONLY has permission to write to text file
   6. Play game as guest or as logged in user
   7. When they win they should send their ID or their record as an object AND their current score to Admin, so then Admin can test if it’s bigger than the hiScore saved to their record
3. **Game Play**
   1. **Dice** 
      1. Function returns a random num between 1 and 6
      2. Function that saves up to 5 dice in an array or vector to represent which dice the player is keeping towards their score
         1. Needs to add and remove a dice between rolls.
      3. Print the value of 1-5 different dice
         1. Test dice against 13 categories and print a score for categories player can potential get points from
            1. Ask user if they want to:

Keep one of the potential points

If yes, then turn ends and save points to scorecard

or keep 1-4 dice and roll again

If yes, get which dice they want to keep and roll the remaining dice again.

Needs to keep count of how many dice the player is keeping on each roll and the value of each dice.

* + 1. Allow user to remove dice if they still rolls remaining
       1. Consider using vectors for which dice the player is keeping between rolls?
  1. **Scorecard class:** 13 scoring categories + 3 different sums
     1. Upper Section categories
        1. 6 categories that summing each side of dice
           1. Sum function that accepts x number of dice and adds their face value?
        2. Total score from all 6 sides of a dice
        3. Bonus – if score is 63 or over, then +35 pts
        4. Total of upper section
     2. Lower Section categories: has to check if the dice meet the conditions for each of these categories. If they do, then it needs to return points.
        1. pts = three of a kind ? sum of all 3 dice : 0;
        2. pts = four of a kind ? sum of all 4 dice : 0;
        3. pts = full House (threeKind + twoKind) ? 25 pts : 0;
        4. pts = small Straight ? 30 pts : 0;
        5. pts = large Straight ? 40 pts : 0;
        6. pts = Yahtzee (5 of kind) ? 50 pts : 0;
           1. pts = Yahtzee bonus? ? 100 pts : 0;
        7. pts = chance pts = sum of all 5 dice : 0;
     3. Total of lower section
     4. Grand total = total of lower + upper section