* Danielle F. **Project Leader**.
* Ismael Perez Saavedra
* Logan Orlando
* Amir

**Important links:**

* Admin & User: Danielle’s github: <https://github.com/koa2019/yahtzee>
* Scorecard & Dice: Ismael’s Github: <https://github.com/Error1417/Yahtzee_Project>
* Gantt Chart <https://docs.google.com/spreadsheets/d/1jXmEnUfsL37aRnG8E6Z-XX4FZrRRHcmFdeqjoNkU2AE/edit?usp=sharing>

**List the meeting times each week, how/when you are meeting, etc......**

* **How are we going to meet?**
  + Danielle doesn’t mind meeting at RCC.
  + Ismael doesn’t mind meeting at RCC.
  + Logan prefers to meet via Zoom.
  + Amir prefers to meet via Zoom
* **Availability**
  + Danielle: Usually anytime
  + Ismael: Mondays before/after class or Fridays @10am
  + Logan: Mondays before class. Possibly Fridays via zoom?
  + Amir: Fridays
* **Next meetup?**
  + March 19: Coundron coffee shop in Riverside @11am
  + March 24 @Riverside Computer Lab @10am
  + March 27 @Riverside Computer Lab @12pm
  + March 31st @Albertos @10am
  + April 3rd before class [CCC Lab](https://www.google.com/maps/place/CCC+Lab/data=!4m2!3m1!19sChIJEWvmsXrPADsRQhhQ39r5Beg)
  + April

**Timeline. What do we need to get done each week?**

* **March 19-25:**
  + Have a basic working game and try plan how we’re going to merge it with User+Admin classes (in progress)
* **March 26-31:** 
  + Finish the Admin and User classes
  + Have a working scorecard class
  + Dice class to push and pull.
* **April 1-8**:
  + Merge User and Game classes
  + Merge scorecard & dice
* **April 9-15:** 
  + Finish game
  + Start documentation
  + Plan our presentation
* **April 16-24:**
  + Finalize game. Clean up code.
  + Finalize Documentation: UML, flowchart,
  + Finalize presentation

**Meeting Summary for March 27:** Met with Logan in person before class and we explained what we need the dice class to do. Had him download Ismael’s code from github. Showed Logan where the print possible points function needs to be in the flowchart.

**Meeting Summary for March 31:**

* Danielle & Ismael attended the meeting @Albertos. Updated Logan via Discord. Amir no show.
* Showed the progress of each of our classes.
* Assigned responsibilities of each team member for the following week.
* Updated Gantt chart.
* **Ismael’s comments:** 
  + # of lines coded/modified: 158
  + I finished the scorecard for the yahtzee game.
  + Next week I plan on having the full game complete.
* **Danielle’s comments:**
  + # of lines coded/modified: 500
  + Versions created:
    - cis17A\_project2\_battleshipYahtzee\_v7
    - cis17A\_project2\_yahtzee\_v8
    - yahtzee\_v14\_AdminUserBattleshipPlayer
    - yahtzee\_v15\_UserBaseAdminDerived
    - yahtzee\_v16\_reWrtTextFile
  + Problem: Unable to rewrite hiScore in a binary file.
  + Solution suggested? ios::ate
  + Problem: How to delete a record in a binary file?
  + Solution suggested? Write records to a temp file, stop when I find the targeted record, skip it somehow and continue writing to temp.dat.
  + Add Player class to main( ) which contains the game flow. Delete old binary and rename temp binary,
    - Created a double pointer to hold 2 player’s scorecards
    - Created play( ) that tracks # of rolls during each player’s turn/
    - Created 2 arrays: dice & keep. Will replace it with Logan’s Dice class.
    - Reads in which dice player wants to keep and removes it before they roll again.
  + Found record by email, reset hiScore, rewrote that record in binary & text files.
  + Restructured hierarchy for Admin & User classes. User is base class and Admin inherits User.
* **Logan’s comments:** 
  + # of lines coded/modified: ?
* **Amir comments:** 
  + # of lines coded/modified: ?

**Weekly Assignments for March 31 to April 7:**

* Danielle:
  + Delete & edit a record from binary & text files.
  + Restructure hierarchy for Admin & User classes.
* Ismael: Finish the Yahtzee Game
* Logan:
  + Make a to-do list for our documentation. Submit a document.
  + Start styling the user interface(HTML and CSS)
    - Yahtzee title
    - ScoreCard
* Amir: Help Danielle with delete/edit binary file or help logan with styling our html page

**Simple UML Chart** - 3/24/23



**Pseudo:**

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