Connect4™ Specifications (w/ timeline)

Project Description:
6x7 board layout (extra board sizes?), search feature, account saving and multiplayer. The objective is to place four consecutive tokens, vertically, horizontally, or diagonally. Authenticated users will be able to play with another user of their choice and will have access to special connect-four tokens they can use while playing. Unauthenticated users will only be able to play with random users, and will only be able to use standard red and yellow connect-four tokens.

Technologies Overview (data structures, languages, host location): JavaFX? + CSS frontend, Java backend, localhost (cloud host?), thread-safe arrays, SQL database, normal array for tokens.

- Token Class (1 hour + skin time)
    - Appearance
    - User it belongs to

- GameBoard Class (3 hour) - **Matthew**
    - Lock and condition represents turns
    - Private List<List<Token>> board = new CopyOnWriteArrayList<List<Token>>()
    - Method to check if valid place to place token and on which row the token lands
        - Int isValid() returns lowest valid row
    - Checking for four in a row (winner) method
        - Prints winner message
        - Ends game
    - Update GameBoard method

- Player Class (extends Thread) (8 hours) **- Anh**
    - Registered boolean
        - Registered players have usernames & can search/be searched
        - boolean, 0 = unregistered, 1 = registered
    - Preferred token variable (default for unregistered players)
        - Akin to private int ID = 10;
    - Boolean insert(int col) [Can't insert -> return false -> prompt new col]
        - Get lock (for gameboard)
        - Calls "isValid()" in GameBoard
        - Set value of GameBoard[isValid()][col] = true;
        - Call Update GameBoard
        - Release lock

    - Boolean to represent if the player's already in a game or not
        - isPlaying
        - Called in server or clientmain class, checks if another player can request to play with this player
    - **run()**: while(true) {
        - If (condition notified)
            - Prompt insert where: insert(scanner.getLine()) //check if its int
                - Scanner for now, but should be front end
            - if(winner) { break;}

- ServerMain (4 hours)- **YJ Lee, (SQL database schema from Maia)**
    - Saving login data (stores all usernames and passwords)
        - SQL database to store data
    - findPlayer()
        - Locates registered player to connect with and forms connection if they're available

- ClientMain [Scanner scanner = new Scanner(System.in)] (5 hours) - **Saleem Bekkali**
    - Pulls data from player class and allows individual user to login to the server to play
    - Contact player system
        - Player inputs which player they want to play with
        - if(isPlaying){ System.out.println("Player is busy.") }

- Front end (10 hours) - **Dylan (write high-level descriptions of pages), Maia**
    - Login page: input fields for user and pass, create an account, or enter as guest
    - Home page: start search (is the only guest functionality), invite player, token skins (, board skins?), favorite token (and board?)
    - Board page: where you actually play the game
    - Victory/Defeat page: displays after win condition for game reached (player earns in-game currency after each victory, currency can be spent to unlock new skins; daily quest/challenges (store variable in database to track current day???))

Test files:
Set up users (user setup, search function), anonymous users test, insert to full token stack test