CS3520 Programming in C++
Final Project

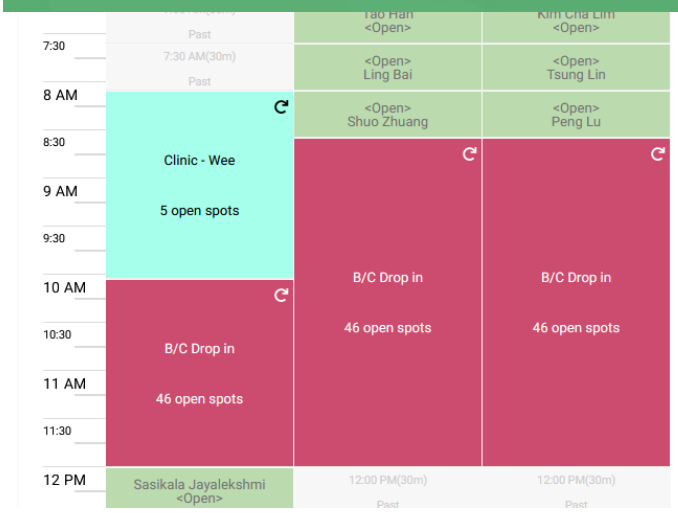
Objective:
Design, develop and implement a working software system in C++. All code must be maintained on GitHub. Please be sure to review:

- Problem Statement
- FAQs
- Final Project Rubric

A Word of Caution!
*You have tremendous amount of freedom in designing this application. You are welcome to add any additional functi… …ctual developer of the application. My sugg… …quired features {listed in this document **and***

Problem Stater
Congrats! City of Ne… …be used to manage reservations for thei… …may be reserved by members. Since cou… …ers (for individual or group games), we n… …chedule. This semi-visual system will h… …eady being used by someone. For this c… …ffice but you should feel free to use any …

Here is a sample vie… …ub. Other examples include clublocker.c…



Here are some important details regarding our system:

- **Courts:** The facility has 3 available courts, and the facility opens from 6am to midnight. Coaches are allowed to hold training sessions or lessons only during 9am-Noon and 3-6pm on weekdays and on Sundays. Club officers may schedule open-play times on one or more courts for a specific category of members on any day from 6-9pm. If the courts are not being used for open-play or by coaches, then individual members may reserve the court for themselves.
- **Reservations**: Each reservation slot is 30 minutes long. A player may not reserve more than 30 minutes per day and may not reserve more than 1 hour of court time per week. Each time slot may have a maximum of two players (where second player can join the reservation of the first player or a reservation may be held by a single member). If you join a reservation as second player, it will be counted as 1 reservation for the day (and the week). Reservations may be deleted only by the players on that reservation (or by officers upon request).
- **Users**: There are three kinds of users - club members, club officers and club coaches. Members wil es). Club officers are also conside
    - **Club** individual lessons or gr hours only. Coac
    - **Club** eball during open-hours s a 3-scale rating syste ed 'A', average playe ro'.
    - **Club** sponsibility of servi y hours during desig ions from players or co hem or delete their reser
- **Reservatior** to the members depending o s to the number / time of reser vance. Any user should be ab above) where each 30-minute sl pen-play sessions are utilized b nd they are not counted towards member's reservation limits. If a court is not reserved (by coaches for training or officers for open-play) atleast 2 days in advance, it will become available for other members to reserve.

System Features:

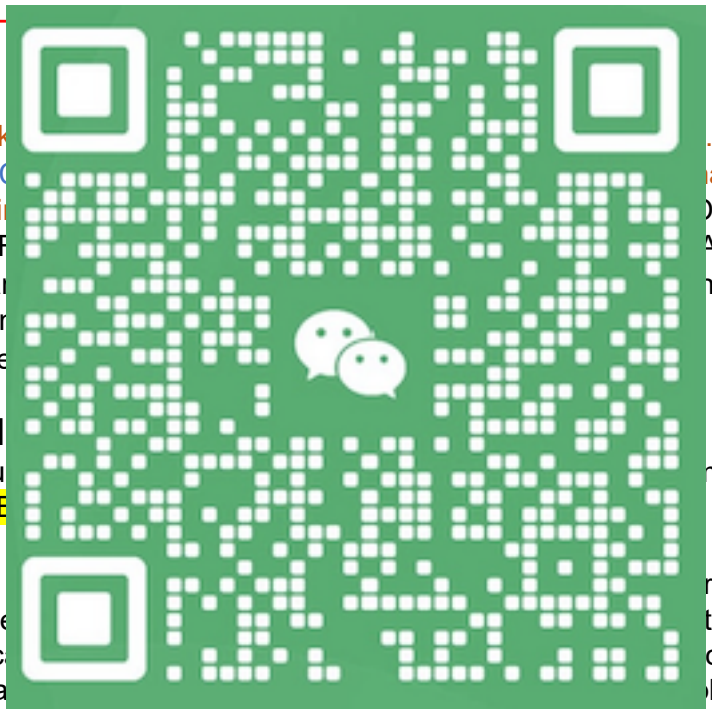
The software you are designing is intended to function as if it was actually going to be used by the City of Newton, thus the user experience you create must be realistic.

Here are some required functionalities:

- Each user should have a unique username (or id) and a specific membership type in the system. At the start, user is presented with a set of menu style options for different features such as view today's schedule (all users), send a message to an officer to request cancellation (members and coaches), reserve a court (all users, where each user may see different available time-slots depending on their role), cancel my reservation (all users), cancel others' reservations (club officers only), and other features as appropriate

- The system should be user-friendly, so there should never be any confusion about what a certain action will do and what input is expected from the user. This means you will probably have to print instructions to the user anytime they are expected to provide any user input. Provide sufficient guidelines and help to use the system (For example: if your system is expecting any input in a specific format, be sure to specify that in the instructions as well as the readme file for your project submission).
- Your system must have **state persistence** by saving the necessary data to files. State persistence allows your system to outlive the process that created it. For example, if you add a member and/or reservation to the system when you initially run it and then close the execution before running the system again, the system must be able to remember the previous reservations and its state. This will allow you to use the previously entered information without having to create all users, reservations, etc. again and again. You can complete the entire project by storing data in files or any other persistent storage medium

***Note**: The design of a Graphical User Interface (GUI) is not needed and is optional. You should develop a command-*



Guidelines:

- You may work . You **MUST** Self-join a group from ating with your group partner or joi T CREATE A NEW STUDENT GR ABLE GROUPS.
- If you have a ntact the instructor to have them re
- See the delive

Design & Devel

The final project inclu nentation. You MUST design the software E

Design: The design roject and will include the list of all classe their relationships & collaboration (CRC c ce diagrams), as well as data structures/da lementation. You will start by analyzing the requirements of the project and by identifying the classes required along with attributes and functionalities.

Make sure that your design follows good object-oriented design principles such as **SOLID, DRY, KISS, YAGNI**, and so on. A doc/docx document listing the design along with above listed diagrams must be submitted by the deadline on Canvas. Please include the following **in your report**:

- Class Diagrams with all classes (with member attributes and member functions), and CRC Cards
- What are class hierarchies and relationships? (UML Diagrams)
- Sequence diagrams (for most important features)
- List of all other data structures or files to be used

After the first version of the initial design is completed, it would be beneficial for you to analyze your design thoroughly and try answering some of the following questions:

- How many classes will you have and how will they interact?
- How will you store user information?
- How will you use hierarchy for different user types?
- How will you represent the project/task selection options and the menu?
- How will you deal with weekly limits on reservations?
- What information is needed to make your system state-persistent?

Initial version of the design will be due on Canvas by **Tuesday 4/18**. Y*ou will have an opportunity to discuss your initial design with TAs during the lecture before the deadline.* You can update the initial design to incorporate any feedback before starting your implementation.

Now imagine yourself as the user of the system (as club member and/or coach) and observe the behavior of the system (and see how it changes). Reflect on the efficiency of your choices. Review your design and make the necessary changes to remove any unnecessary attributes and functions. You may have to add some more functionality that you might have missed in your initial design. Your **Final R** nal design.

Implementation: C epository (on Khoury github) and start imp e user-friendly menu options that would al hat the menu options only work when corre idually to ensure that all available functiona shared amongst team members via github. ators.

Testing: Test each o ber, coach and office. Please ensure that th on. Use the rubric as a check-list (if neede erall testing:

- Your software The most up-to-date schedule shou
- You should b reservations, remove yourself from tion and so on.
- Your software y time and can re-run it without losi atus).

The final implementa day 4/25. Be sure to review the **checklist** given below:

Deliverables: The following items must be submitted:

- **All source and header files** related to the system implementation (on Khoury github).
- A **README** file with any information regarding compilation and testing that we may need to know, in order to compile and run your software successfully. Include **any other files** needed to compile or test or any default username and passwords.
- **Source code must be properly organized, readable, and must use proper best coding practices.**
- **The report** with the final design (UML diagrams showing class relationships & hierarches, and sequence diagrams), CRC Cards and brief details of testing activities. Feel free to show additional testing you performed in the report
- {*Optional*} Video Demo (12 minutes long max) showing all features of your working system. Voice over is not mandatory

Be sure to use github properly i.e., to store all updates of your work throughout the project. Make frequent commits to github to show your progress and avoid using github for a single, final commit only. After the work is completed, commit the final version of the code to github, then download it again (on a new computer) to ensure that it compiles and runs as expected.

Additional Requirements:

- Properly implement all constructors and destructors (Rule of 5). Use RAII design principle.
- Use STL for any data structures, and algorithms that you might need.
- Prefer using smart pointers over raw pointers.
- If using raw pointers, use dynamic memory allocation as much as possible and make sure that your software does not have any memory leaks (check with valgrind to confirm and provide screen grab)

Submission Checklist:

You can use the following checklist to ensure that you have submitted everything required for the project:

- {Before submi ion of your code from Khoury github
- Did you subm e file?
- Does your su d the names of your partners?
- Did you add th ithub repository?
- Did you subm

FAQs:

Here are some common questions

- **Q**: Who adds the members or coaches to the system? How many members should there be?
  A: At startup, you can add few users as needed. At the minimum, you should setup your system with 5+ users, 2+ officers and 2+ coaches.
- **Q**: When do we create users? How do you switch users? Do they need to login?
  A: There are two options that are typically suggested: one is to create a few users (members, club officers and coaches) at the start and the other is to create them on the fly. Use whatever works for you. In either case, their information should be preserved. It would be advisable to implement a login screen/menu where users can provide their username and password. It would make sense to add those as options to main Menu.
- **Q:** How many days should a user be able to view the schedule for?
  A: User should be able to see the schedule for the entire week. You may also provide an option to show the daily schedule or schedule for 3 days. You should populate data for atleast 2 week ummy names.
- **Q**: Do we disp
  A: Yes, you sh You can start the week from Mo
- **Q**: What happ servation?
  A: To keep thi request. This request may be comi Similarly, officers can make a reserv
- **Q**: What are o
  A: Open play ers of similar skill set. Players gathe loes not track who is playing and f ards daily or weekly reservation lin
- **Q**: What happ arlier?
  A: If the coac es available for other players to rese ithin 48 hrs only if the court remains

Initial Design of Final Project: Reservation System - (25 pts)

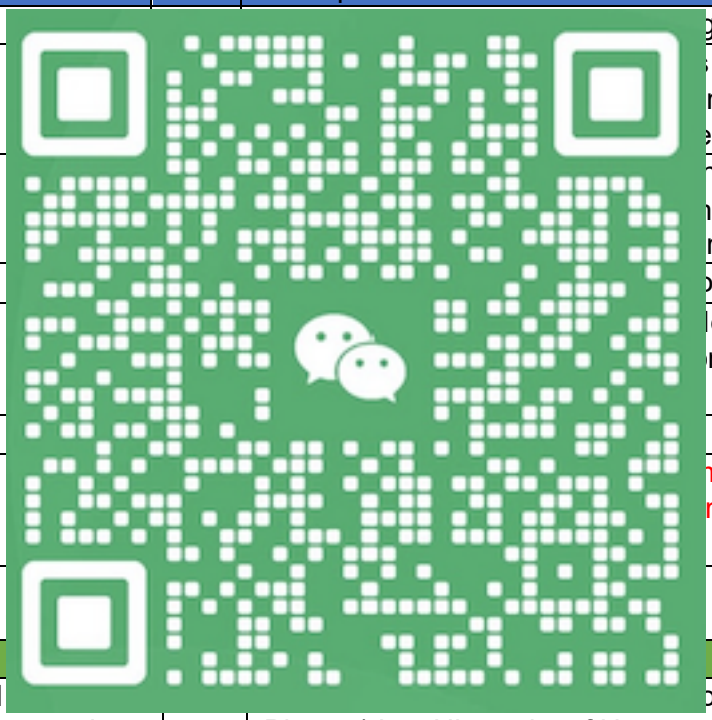
General – 25 pts

Criteria	Pts	Description
Initial Design	10	-10: UML diagrams [Class Diagrams showing relationships and Sequence diagrams], CRC Cards, and brief discussion -5: Some diagrams are missing, or they don't include important information
Quality of Design	15	-15: Quality of Initial design (following Design Principles) -5: Minor design flaws

Final Project Rubric (150 pts) – Reservation System

General – 40 pts

Criteria	Pts	Description
README file		gram is missing
Final Report		s showing relationships rds missing ed
Quality of Design		ng Design Principles) ns showing relationships rds, and brief discussion
Video Demo		orking system
Coding Style		er block, missing n, and variable names,
File Organization		
Makefile		num) if the code doesn't redit by visually



Project specific –

Criteria		
members types and info (unique username and details).		ot supported (i.e., Coach / Players) i.e., Hierarchy of Users -2: The program does not prevent duplicate usernames and does not provide appropriate error messages
view daily schedule and reservations	17	-7: Court availability is not maintained properly. -5: The program does not neatly display updated schedule for all courts, showing the name(s) of people on each reservation. -5: If users can't check and track their reservations for the day or week.
Members can reserve a court	23	-3: Club coaches should not be able to reserve any court for individual lessons or group training sessions other than designated coaching hours. -3: Club members should not be able to reserve courts other than their designated reservation times.

		-3: Club officers are not able to reserve courts for open play during designated time slots. -2: Club officers are not able to reserve courts for themselves. -2: 3-scale rating system (ABC) for players' skills is not implemented properly. -3: Officers are not able to modify reservations from players or coaches upon their requests. -2: Member can't join another reservation. -3: Deletion of reservation is not proper by all members. -2: members can make reservation during restricted hours (open-hours or coaches training slots) if they were not reserved 48 hrs in advanced
System enforces reservation rules	16	-2: Reservation slot length is overlapping. -2: Maximum reservation time per day is not maintained. week is not maintained. er time slot are not ation. embers reservation limit. to 7 days in advance for ed. to 2 days in advance for ot booked by any user" to 2 days in advance for s have not booked the
cancel others' rese		llation of their o the officers. request for each user
Overall Quality of I		ptions r -2: If user is not given proper guidance or useful prompts for inputs (Usability) -5: If user experience is not delightful or fun.
Data persistence	15	-5: for each instance of lack of data persistence. (All info for players (skills, weekly/Daily booking should be retained). -10: User should be able to successfully save the state and load it back without losing any data.
Memory Allocation & Using smart C++.	13	-3 Implementation of RAII is not proper. -3. Rule of 5 is not implemented. -3: STL Not used in the code. -3 All Memory not released (if raw pointers are used), try to use smart pointers. -1 If screen grab of *valgrind* not provided
Compile and Run		-50% **if program doesn't compile or crashes.** We will test it on Khoury server only.

Change Log:

- March 26th (7pm): Initial Version