****

**Hackathon Scheduler**

**Course:** BSCYSev-III-B  
**Subject:** Data Structures and Algorithms  
**Course Code:** CS-213

**Team Members:**

* Zain Iqbal Khan (232144)
* Shaheer Ali (232078)
* Imad Ul Islam (232082)

**Instructor:** Dr. Muhammad Imran

**Overview**

Our **Hackathon Scheduler** is an application designed to efficiently manage and organize cybersecurity hackathons. The application facilitates the scheduling of matches, simulation of progress across rounds, and seamless tracking of match results. Developed using C++ and Windows Forms, our Hackathon Scheduler offers a user-friendly interface to oversee all aspects of a hackathon, ensuring a smooth and engaging experience for organizers and participants alike.

**Overall Flow of the Program**

1. **Startup**
   * The application initializes with the MyForm class.
   * Displays the main interface, including a welcome message, hackathon details, and a **"Start"** button.
2. **Add Teams**
   * Clicking the **"Start"** button navigates to the AddTeams form.
   * Users can input and manage the teams participating in the hackathon.
3. **Manage Rounds**
   * After adding teams, users are directed to the ManageRounds form.
   * This section allows for scheduling, simulating, and viewing the progress and results of matches across different rounds.
4. **Simulation**
   * Rounds are simulated in the background using a BackgroundWorker.
   * Ensures the user interface remains responsive during the simulation process.
5. **Completion**
   * Once all rounds are concluded, the application displays the winner of the hackathon.

**Major Components**

* **MyForm**
  + **Description:** The main form that greets the user and provides information about the hackathon.
* **AddTeams**
  + **Description:** A form where users can input and manage the teams participating in the hackathon.
* **ManageRounds**
  + **Description:** A form dedicated to scheduling, simulating rounds, and displaying match progress and results.
* **Scheduler**
  + **Description:** A class responsible for match scheduling, tracking progress, and managing results.

**Classes**

**1. MyForm**

* **Purpose:** Serves as the main interface for the application.
* **Key Methods:**
  + **startButton\_Click():** Handles the click event of the "Start" button to open the AddTeams form.

**2. AddTeams**

* **Purpose:** Provides the interface for adding teams to the hackathon.
* **Key Methods:**
  + **addButton\_Click():** Adds a new team to the hackathon.
  + **doneButton\_Click():** Proceeds to the ManageRounds form after team addition is complete.

**3. ManageRounds**

* **Purpose:** Manages the scheduling of matches, simulates rounds, and displays match progress and results.
* **Key Methods:**
  + **OnScheduleRoundClick():** Schedules a new round of matches.
  + **OnSimulateRoundClick():** Initiates the simulation of the current round.
  + **OnSimulateRoundWorker():** Background worker method responsible for simulating matches.
  + **OnSimulationProgress():** Updates the user interface with the progress of the simulation.
  + **OnBackClick():** Handles navigation back to the previous form.

**4. Scheduler**

* **Purpose:** Oversees the scheduling and tracking of matches and rounds.
* **Key Methods:**
  + **scheduleFirstRound():** Schedules the initial round of matches.
  + **scheduleNextRound():** Schedules subsequent rounds based on previous results.
  + **reset():** Resets the scheduler to prepare for a new hackathon.
  + **getCurrentRound():** Retrieves the current round number.
  + **getWinnerTeam():** Determines and returns the winner of the hackathon.

**5. TeamQueue**

* **Purpose:** Manages a queue of teams, used for scheduling matches in subsequent rounds.
* **Key Methods:**
  + **enqueue(const Team& team): Adds a team to the queue.**
  + **dequeue(): Removes and returns the team at the front of the queue.**
  + **isEmpty(): Checks if the queue is empty.**

**Data Structures**

**1. Arrays**

* **Fixed-size Arrays:**
  + **teams[MAX\_TEAMS] in the Scheduler class:** 
    - Stores the list of teams participating in the hackathon.
  + **newTeam.members[MAX\_MEMBERS] in the loadTeamsFromFile method:** 
    - Stores the names of the team members.
  + **date[20] in the scheduleFirstRound method:** 
    - Stores the formatted date string for scheduling matches.
  + **nextRoundTeams[8] in the scheduleNextRound method:** 
    - Temporarily stores teams advancing to the next round.

**2. Pointers**

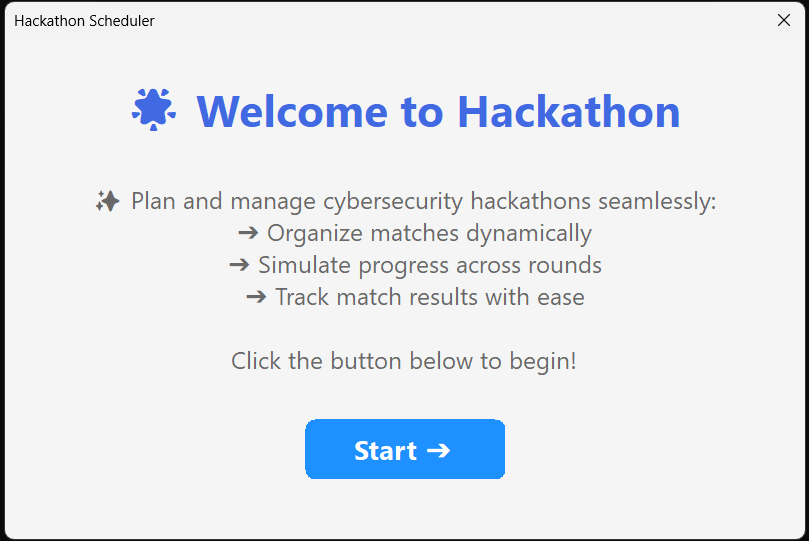
* **Pointer to Object:**
  + **Match currentMatches in the simulateRound method:** 
    - Points to the current match being simulated.
  + **Match prevMatch in the scheduleNextRound method:** 
    - Points to the previous match in the list.
  + **Match finalMatch in the getWinnerTeam method:** 
    - Points to the final match in the list.

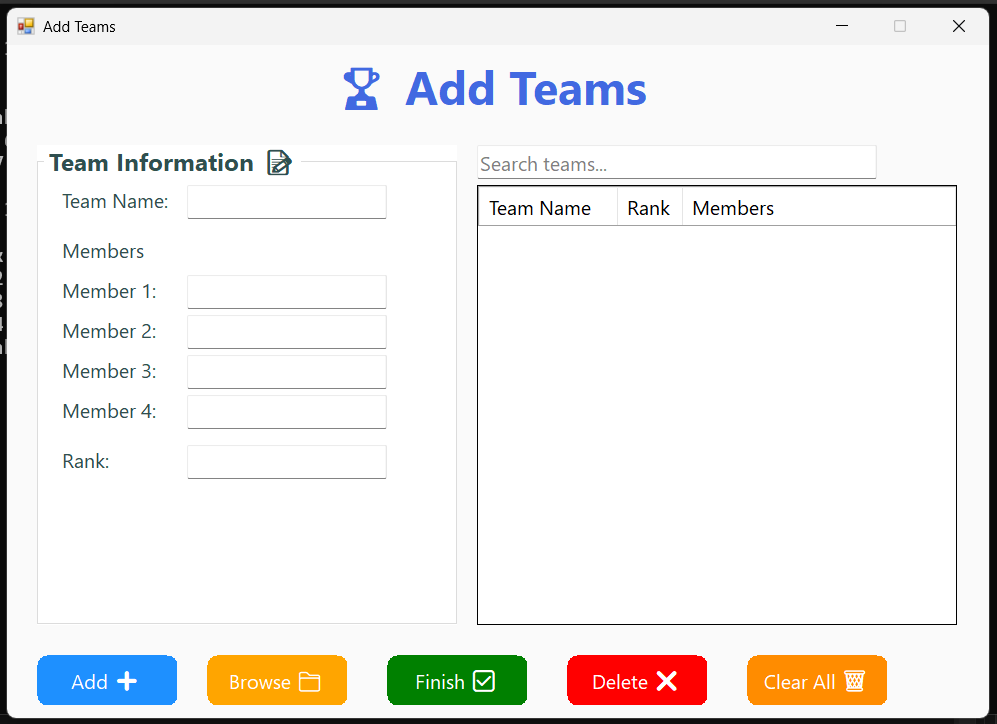
**3. Structures**

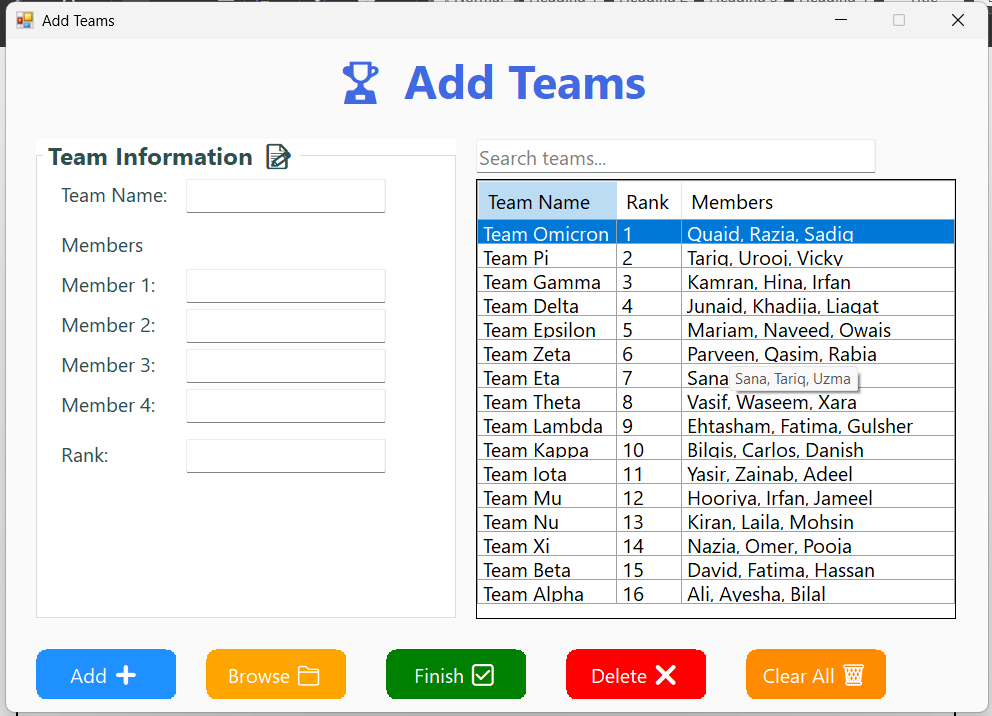
* **Team**
  + **Purpose:** Represents a team participating in the hackathon.
  + **Attributes:**
    - **char teamName[50]:** Name of the team.
    - **int memberCount:** Number of members in the team.
    - **int rank:** Rank of the team.
    - **char members[MAX\_MEMBERS][MEMBER\_NAME\_SIZE]:** Names of the team members.
* **Match**
  + **Purpose:** Represents a match between two teams.
  + **Attributes:**
    - **char team1[50]:** Name of the first team.
    - **char team2[50]:** Name of the second team.
    - **char date[20]:** Date of the match.
    - **char time[10]:** Time of the match.
    - **bool completed:** Indicates whether the match is completed.
    - **char winner[50]:** Name of the winning team.
    - **Match next:** Pointer to the next match in the list.
* **MatchList**
  + **Purpose:** Represents a list of matches in a round.
  + **Attributes:**
    - **Match head:** Pointer to the first match in the list.
    - **Match tail:** Pointer to the last match in the list.

**Summary**

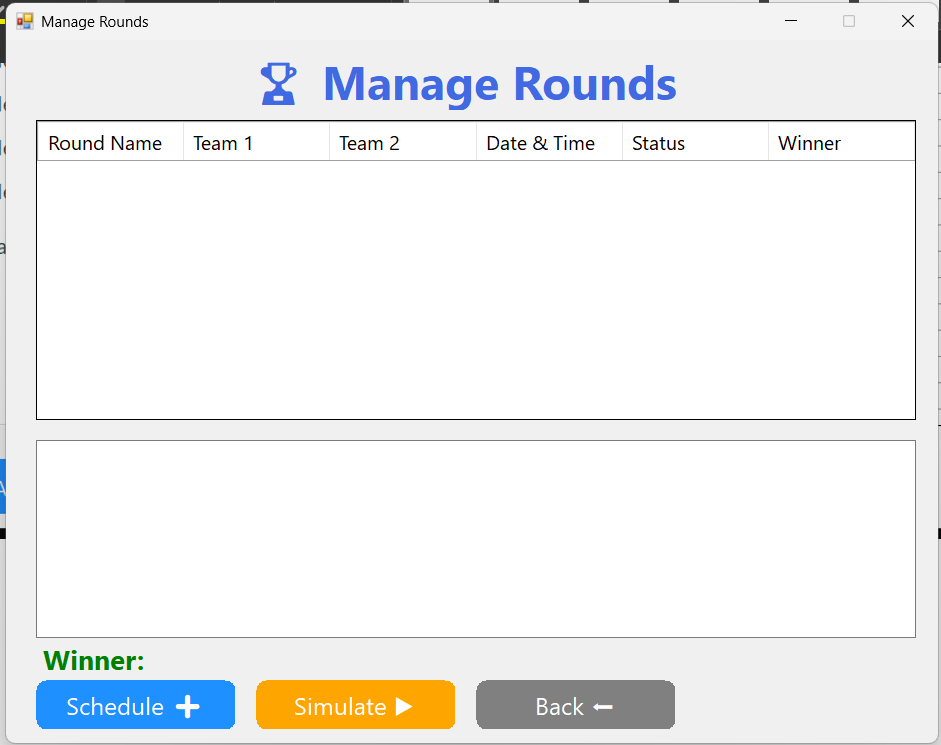
* **Arrays:** Utilized to store teams, team members, and temporary data such as match scheduling information.
* **Pointers:** Essential for dynamic memory management and for traversing linked lists of matches, facilitating efficient data handling and manipulation.
* **Structures:** Define and organize data representations for teams, matches, and lists of matches, ensuring structured and maintainable code.
* **TeamQueue:** Manages a queue of teams, aiding in the scheduling of matches for subsequent rounds.

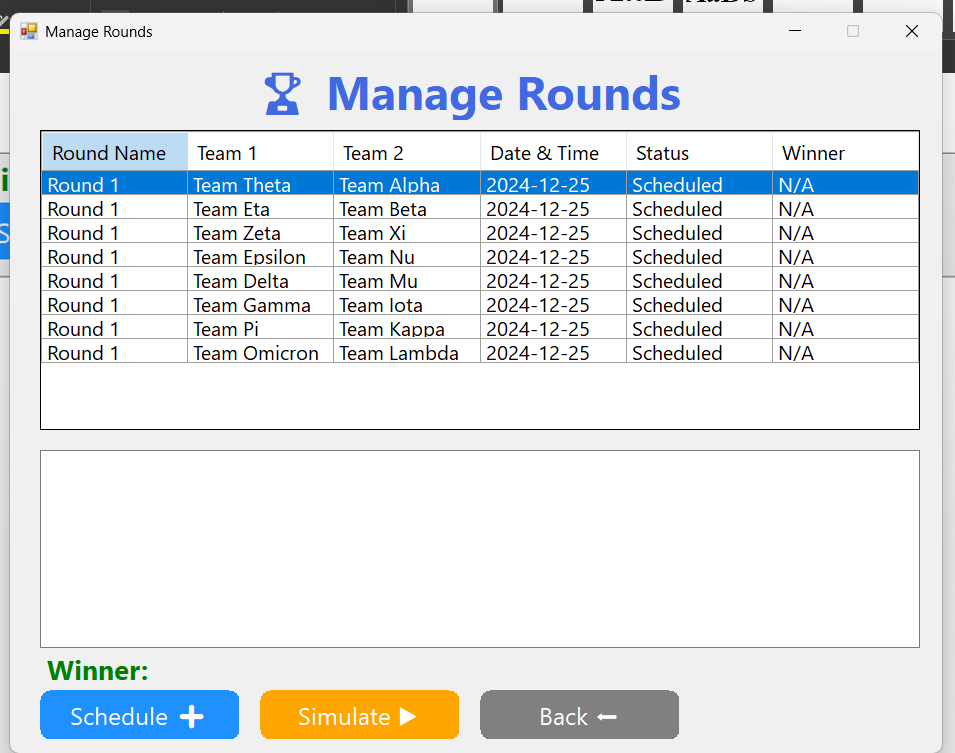
**Screenshots:**Welcome Screen:  


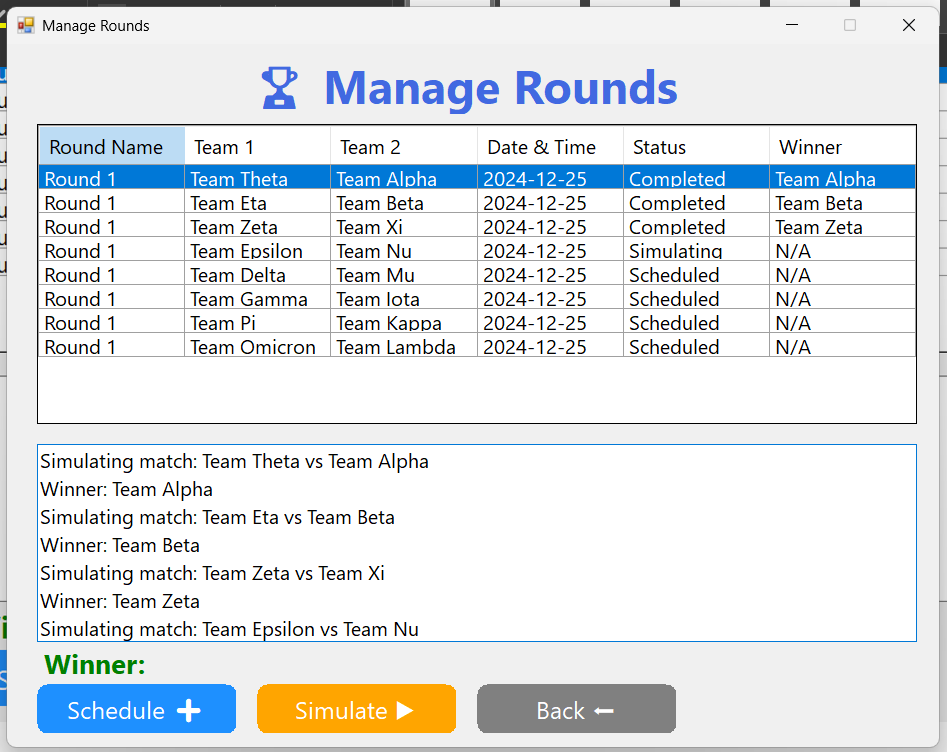
Form to Add and manage teams:  


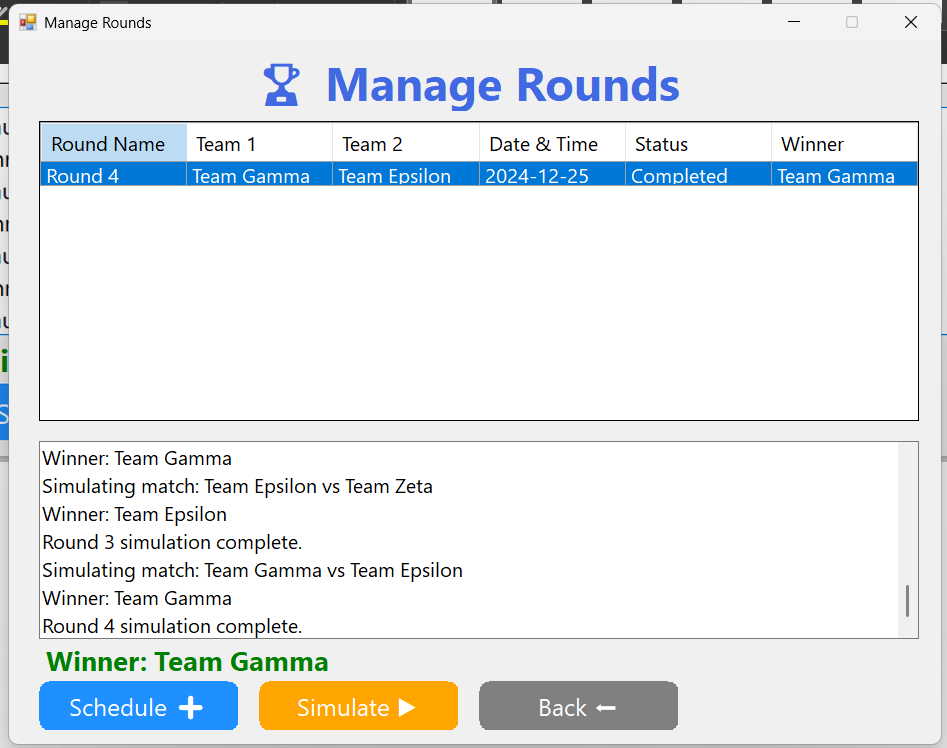


Form to Schedule and Simulate Rounds:









**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***