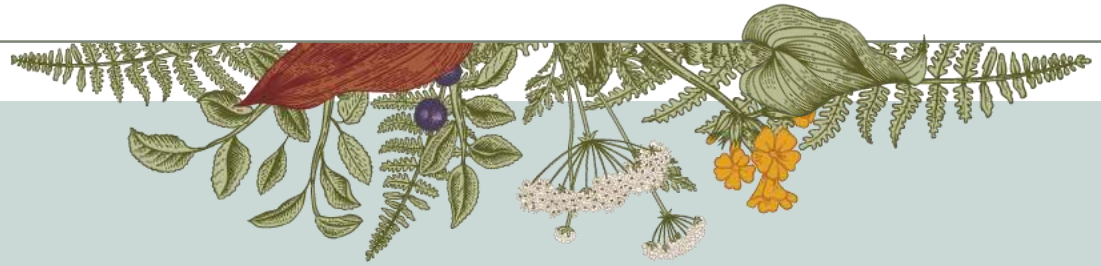
GEU BADMINTON TOURNAMENT

By group c

Project Overview

About Badminton Tournament (by group c)



CONTENT

- Introduction of topic
- Objectives
- Slides
- Team
- Prototype
- Summary

Introduction

Creating a complete software solution for a badminton tournament involves writing a substantial amount of code, and it's beyond the scope of a single response. However, we can provide you with a simplified example focusing on the administration work , entries of individual players , teams like double, and mixed doubles, sponsors and scheduling part of the system.

Objectives

EVENT SETUP

- Define tournament details (dates, venue, rules).
- Specify categories (singles, doubles, mixed doubles).

REGISTRATION MANAGEMENT AND SCHEDULE GENERATOR

1. Registration Management:

1. Individual player registration.
2. Team registration for doubles and mixed doubles.
3. Collect necessary details (names, contact information, skill levels).

2. Schedule Generator:

1. Automatically generate match schedules for 20 days.
2. Consider court availability, player preferences, and rest periods.

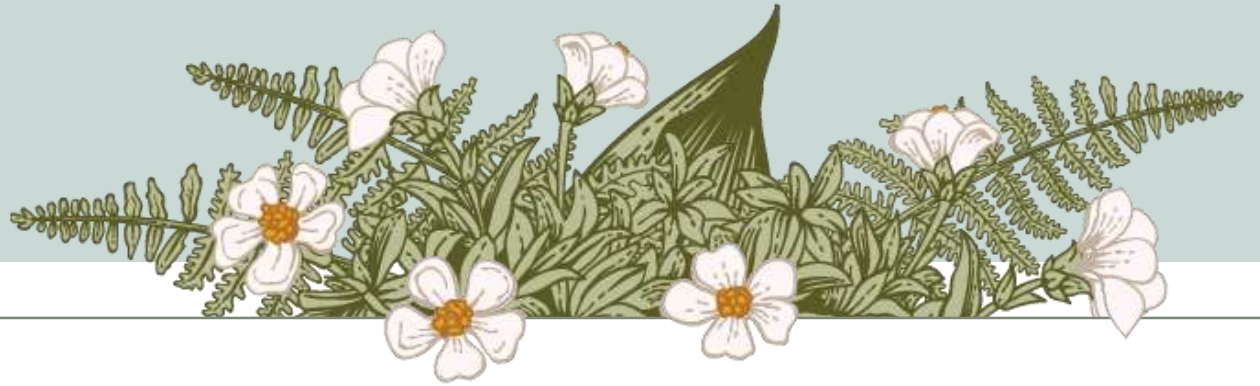
RESULTING TRACKING AND COMMUNICATION HUB

1. Results Tracking:

1. Record match results.
2. Update rankings and standings.

2. Communication Hub:

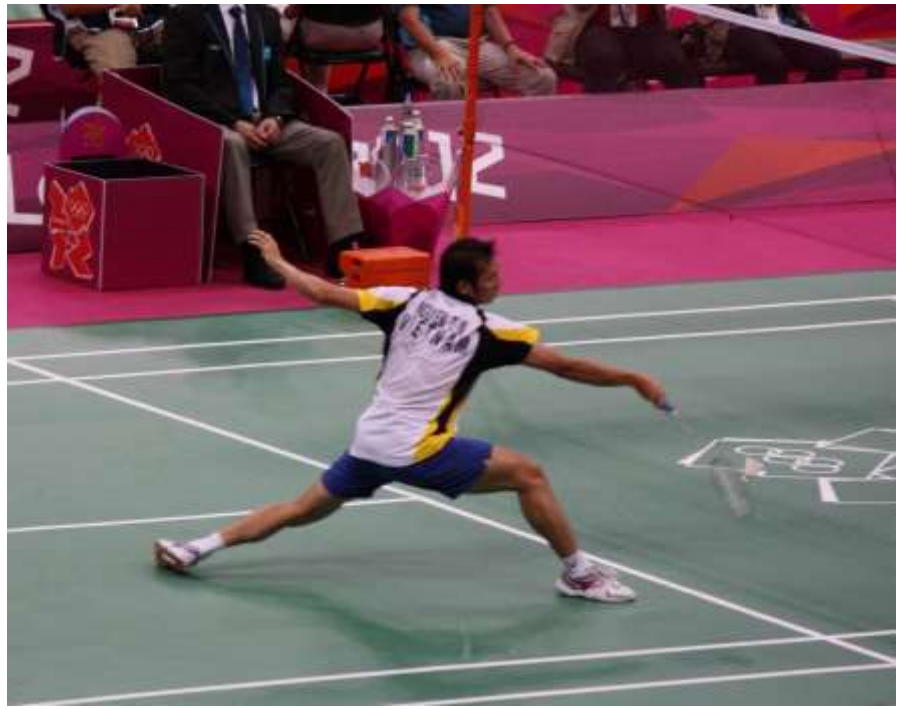
1. Send notifications to participants.
2. Provide updates on match schedules, results, and other announcements.



“Badminton is like ballet dancing. It requires a lot of control, strength, mind play and measured
movement”

Slide 1: Introduction

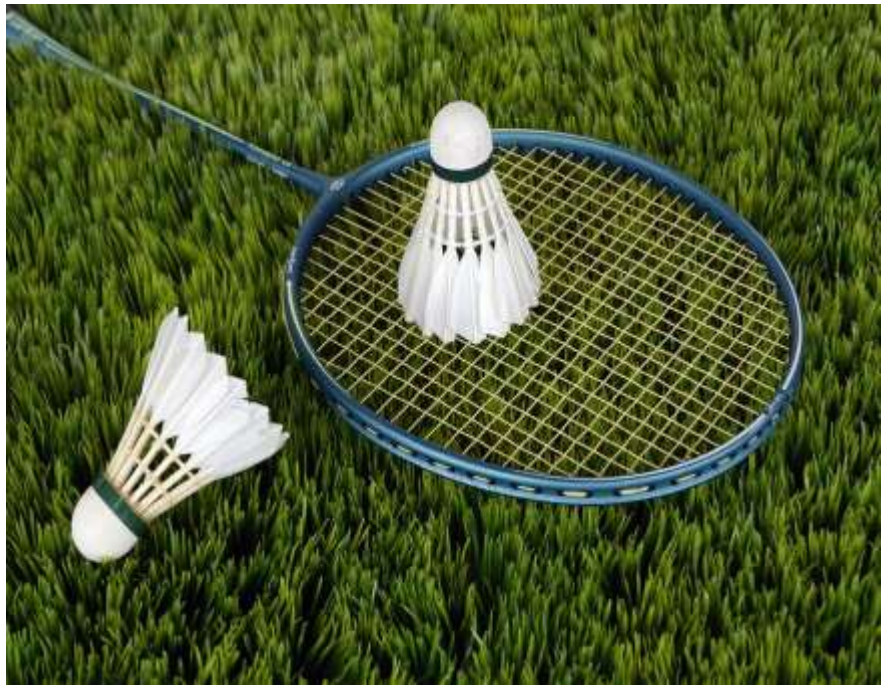
- Brief overview of GEU Sports Ltd. Badminton Tournament.-
- Highlighting the importance of efficient administration and sponsor engagement.



Slide 2: Player Entries (cont)*

- *- Criteria for player entries (age, gender, nationality).*
- *Total participants (60), specifying male and female participants.*
- *Entry fees and document verification process.*
- *Details on double and mixed doubles entries.*

- *Clarification on common participation in single, double, and mixed categories*

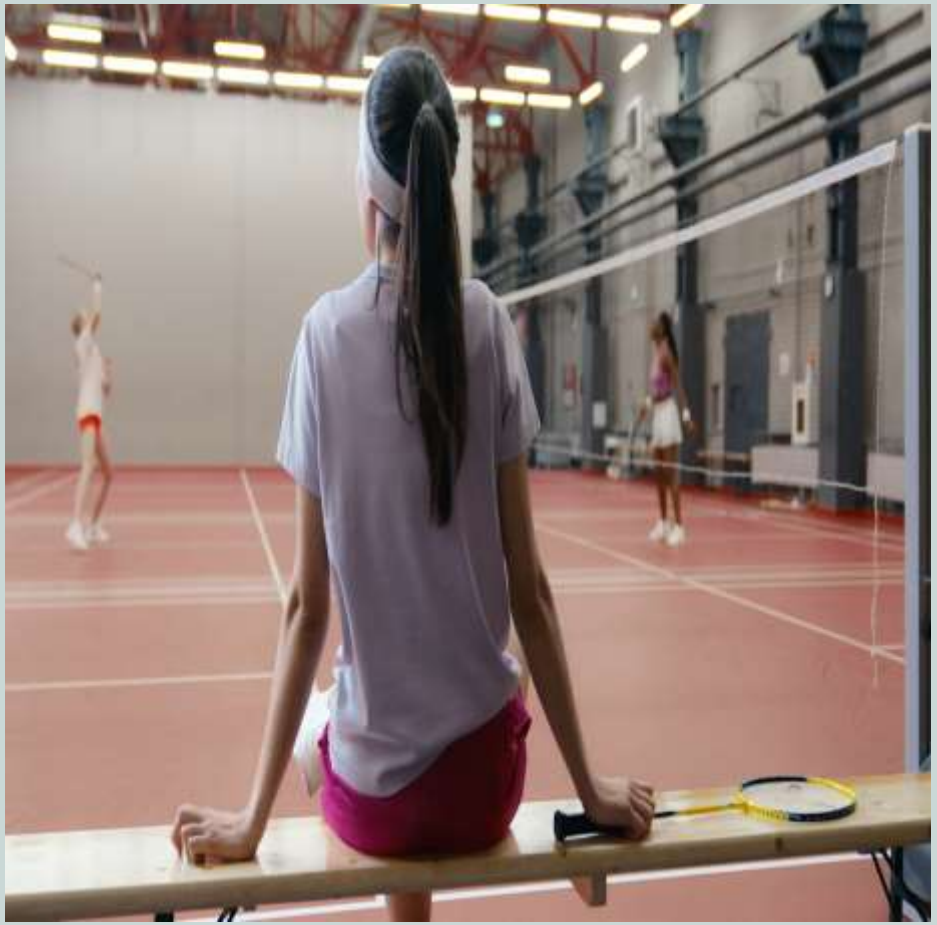


```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

// Structure to represent a match
struct Match {
    int courtNumber;
    int player1Index;
    int player2Index;
    // Additional match details can be added here
};

// Function to generate fixture on a random basis
void generateFixture(struct Player players[], int numPlayers, struct Match matches[], int numMatches) {
    // Logic to generate random matches, ensuring no clashes for a player
    // Example: (Generate matches randomly)
    srand(time(0));
    for (int i = 0; i < numMatches; ++i) {
        matches[i].courtNumber = (rand() % 10) + 1; // Randomly assign a court number
        matches[i].player1Index = rand() % numPlayers;
        matches[i].player2Index = rand() % numPlayers;
        // Ensure players don't play more than one match in a day (for demonstration purposes)
        // Actual implementation needs more sophisticated logic considering schedule and time
        while (matches[i].player1Index == matches[i].player2Index) {
            matches[i].player2Index = rand() % numPlayers;
        }
    }
}

// Function to calculate sports material requirement per match and overall total requirement
void calculateMaterialRequirement(struct Match matches[], int numMatches) {
    // Logic to calculate sports material requirement per match and overall
    // Example: (for demonstration purposes)
    int shuttlecockPerMatch = 6; // Assuming 6 shuttlecocks per match
    int totalShuttlecocks = shuttlecockPerMatch * numMatches;
```



Slide 3: Player Entries (cont.)

- Details on double and mixed doubles entries.

- Clarification on common participation in single, double, and mixed categories

```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>

// Structure to represent a player
struct Player {
    char name[50];
    int age;
    char gender[10];
    char location[50];
    bool international;
    bool singlesParticipant;
    bool doublesParticipant;
    bool mixedDoublesParticipant;
};

// Function to validate player's age
bool validateAge(int age) {
    return (age >= 18 && age <= 25);
}

// Function to validate player's eligibility based on location and international status
bool validateEligibility(const char *location, bool international) {
    // Implement logic to check if the player is from India or living in India for more than a year
    // Return true if eligible, false otherwise
    // Example: (logic to check location and international status)
    return (strcmp(location, "India") == 0 || international);
}

// Function to handle player entries
void enterPlayers(struct Player players[], int numPlayers) {
    for (int i = 0; i < numPlayers; ++i) {
        printf("\nEnter details for Player %d:\n", i + 1);
        printf("Name: ");
        scanf("%s", players[i].name);
        printf("Age: ");
```

*Slide 4 : Administration Work AND Administration Work (cont.)

- **Administration work**
- Random fixture generation through software.
- Limitation on the number of matches per player Per day.
- Availability of 10 courts and sports material requirements.

- **Administration Work (cont.)**
- Dining arrangements for players over 20 days.
- Showcase overall total sports material requirements.

```c
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>

// Structure to represent a team under administrator
struct Team {
    int player1Index;
    int player2Index;
    // Additional team details can be added here
};

// Function to update fixtures regularly by the administrator
void updateFixtures(struct Match matches[], int numMatches) {
    // Logic to update fixtures based on the administrator's input
    // Example: (for demonstration purposes, assuming new matches are entered manually)
    printf("Please enter updated fixtures:\n");
    for (int i = 0; i < numMatches; ++i) {
        printf("Enter details for Match %d - Court: ", i + 1);
        scanf("%d", &matches[i].courtNumber);
        printf("Enter Player 1 Index: ");
        scanf("%d", &matches[i].player1Index);
        printf("Enter Player 2 Index: ");
        scanf("%d", &matches[i].player2Index);

    }
updated fixtures
void selectTeams(struct Match matches[], int numMatches, struct Player players[], int numPlayers) {
    // Logic to select teams for quarter-finals, semi-finals, and finals based on updated fixtures
    // Example: (for demonstration purposes, assuming simple selection based on match results)
    printf("\nSelecting teams for quarter-finals, semi-finals, and finals:\n");

    // Assume winners from matches progress to the next round
    // (In a real scenario, results would be evaluated)
    struct Team quarterFinalsTeams[4];
    struct Team semiFinalsTeams[2];
```

Slide 5: Sponsors and Sponsors (cont.)*

Sponsors

- ❑ Invitation for sponsor entries.-
- ❑ Inclusion of publicity points in sponsorship packages.

Sponsors (cont.)

- ❑ Save and organize sponsor entries and materials on the software.
- ❑ Detailed report format for sponsors' entries.

```c
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>

// Structure to represent a sponsor
struct Sponsor {
    char name[50];
    float contribution; // Amount contributed by the sponsor
    int publicityPoints;
};

// Function to handle sponsor entries
void sponsorEntries(struct Sponsor sponsors[], int numSponsors) {
    for (int i = 0; i < numSponsors; ++i) {
        printf("\nEnter details for Sponsor %d:\n", i + 1);
        printf("Name: ");
        scanf("%s", sponsors[i].name);
        printf("Contribution amount (in Rs.): ");
        scanf("%f", &sponsors[i].contribution);
        // Assigning publicity points based on the contribution (for demonstration purposes)
        // Actual logic for assigning points can be based on specific criteria
        sponsors[i].publicityPoints = sponsors[i].contribution / 1000; // Assuming 1000 Rs. = 1 publicity point
    }
}

// Function to save sponsor details and generate a detailed report
void saveSponsorDetails(struct Sponsor sponsors[], int numSponsors, struct Match matches[], int numMatches) {
    // Logic to save sponsor details and generate a detailed report
    // Example: (for demonstration purposes)
    printf("\nDetailed Report:\n");
    printf("Sponsor Entries and Publicity Points:\n");
    for (int i = 0; i < numSponsors; ++i) {
        printf("Sponsor Name: %s, Contribution: Rs. %.2f, Publicity Points: %d\n",
               sponsors[i].name, sponsors[i].contribution, sponsors[i].publicityPoints);
    }

    printf("\nSchedule with Match Details:\n");
```

Slide 6: Schedule , Prize Distribution *and Fixture Updates*

Schedule*- Submission of the tournament schedule with all details.- Highlighting the importance of regularly updated fixtures.

Prize Distribution*- Explanation of prize distribution (half of the total sponsors' money).- Breakdown of prizes for winners, silver, bronze, and doubles.

Fixture Updates*- Regular updates on fixtures for quarter-finals, semi-finals, and finals.- Team selection based on updated fixtures.

Prototype

About fixture

```c
// Function to update fixtures regularly by the administrator
void updateFixtures(struct Match matches[], int numMatches) {
    // Logic to update fixtures based on the administrator's input
    // Example: (for demonstration purposes, assuming new matches are entered manually)
    printf("Please enter updated fixtures:\n");
    for (int i = 0; i < numMatches; ++i) {
        printf("Enter details for Match %d - Court: ", i + 1);
        scanf("%d", &matches[i].courtNumber);
        printf("Enter Player 1 Index: ");
        scanf("%d", &matches[i].player1Index);
        printf("Enter Player 2 Index: ");
        scanf("%d", &matches[i].player2Index);
```

Prize distribution

```c
// Calculate prize distribution based on the total sponsors' money
float totalSponsorsMoney = 0;
for (int i = 0; i < numSponsors; ++i) {
    totalSponsorsMoney += sponsors[i].contribution;
}

float totalPrizeMoney = totalSponsorsMoney / 2;
printf("\nPrize Distribution:\n");
printf("Total Prize Money: Rs. %.2f\n", totalPrizeMoney);
printf("Winner: Rs. %.2f\n", totalPrizeMoney * 0.6); // Assuming 60% for the winner
printf("Silver: Rs. %.2f\n", totalPrizeMoney * 0.3); // Assuming 30% for the silver
printf("Bronze: Rs. %.2f\n", totalPrizeMoney * 0.1); // Assuming 10% for the bronze
// Similarly, calculate and print prizes for doubles and other categories
}

// Main function
int main() {
    // Assuming sponsors' details are already entered and available in the 'allSponsors' array
    const int totalSponsors = 5; // Assuming 5 sponsors
    struct Sponsor allSponsors[totalSponsors];
```

Schedule.. Importance of fixture updating

```c
// Function to generate fixture on a random basis
void generateFixture(struct Player players[], int numPlayers, struct Match matches[], int numM
    // Logic to generate random matches, ensuring no clashes for a player
    // Example: (Generate matches randomly)
    srand(time(0));
    for (int i = 0; i < numMatches; ++i) {
        matches[i].courtNumber = (rand() % 10) + 1; // Randomly assign a court number
        matches[i].player1Index = rand() % numPlayers;
        matches[i].player2Index = rand() % numPlayers;
        // Ensure players don't play more than one match in a day (for demonstration purposes)
        // Actual implementation needs more sophisticated logic considering schedule and time
        while (matches[i].player1Index == matches[i].player2Index) {
            matches[i].player2Index = rand() % numPlayers;
        }
    }
}
```

Summary

This outlines the basic structure for registering players, teams, and sponsors, as well as generating a simplified schedule of geu badminton tournamnent
Tournament Administration Modul
Player/Team Module:
Sponsorship Module:
Administrative module

Thank you

Shivam

Maletha

Chirag Paliwal

Rohan Maurya

Sanjesh Koli

Kanishka

(group c)