1. Functional Requirements

System Overview

AuraC² (Aura Contest Control) is an open-source contest management system developed to provide universities and organizations with a secure, offline-first environment for hosting programming contests. The system manages contest creation, participant registration, automated code judging, real-time scoreboards, and security monitoring. It aims to replace dependency on online platforms such as Codeforces or PC² by offering a self-hosted, modular, and reliable alternative that operates efficiently within LAN environments.

Functional Requirements List

FR1: Contest Management

The system shall allow administrators to create, configure, and manage programming contests.
This includes defining contest metadata (title, start time, duration), uploading problem sets, and managing participant teams. The system shall also allow editing or deleting contests before they begin.

FR2: Automated Submission and Judging

The system shall allow registered teams to submit solutions to programming problems during the contest.
Each submission shall be automatically evaluated using the Judge0 API (in the initial version) and return verdicts such as *Accepted*, *Wrong Answer*, or *Time Limit Exceeded*.

FR3: Real-Time Scoreboard

The system shall display a real-time scoreboard showing all participating teams, their ranks, number of solved problems, and penalty times.
The scoreboard shall automatically update whenever a new submission is judged, maintaining synchronization across all connected clients.

FR4: Team Authentication and Security

The system shall provide a secure login mechanism for both administrators and contestants using JWT-based authentication.
It shall restrict unauthorized access, monitor connected devices during contests, and alert administrators if suspicious hardware (e.g., USB modems or mobile hotspots) is detected.

FR5: Offline and LAN Mode Operation

The system shall support full functionality without requiring an active internet connection during contests.
All submissions, judgments, and scoreboard updates shall operate within the local network

---

FR6: Clarification Messaging System

The system shall allow teams to send clarification requests to contest administrators and receive replies during the contest.
Each request shall include the related problem and message body. Administrators can reply privately to one team or publicly to all teams.

2. System Analysis

2.1 System Description

AuraC² (**Aura Contest Control**) is a secure, offline-first contest management system designed to host and monitor programming contests within a LAN environment.
It allows administrators to create and manage contests, handle team registration, monitor submissions, and ensure fair competition without depending on any external internet service.

Submissions are processed asynchronously to maintain high performance and reliability.
When a team uploads its code, the backend forwards the submission to an internal **message queue (RabbitMQ)**, which dispatches it to a **Submission Service Worker** for evaluation.
The worker sends the code to the **Judge0** automated judging system and receives the final result through a **callback (webhook)**.
This asynchronous architecture eliminates blocking requests, improves scalability, and keeps the system responsive even under heavy contest load.

Teams can view real-time verdicts and live scoreboards, while administrators can manage problems, monitor contests, and track suspicious activities through the **Security Monitor module**, which detects device or network anomalies (such as USB or hotspot connections) to preserve contest integrity.

AuraC² is implemented using **Spring Boot** for backend services, **React** for the frontend interface, **PostgreSQL** for persistent storage, and **RabbitMQ** for reliable communication between services.

Primary users include:

- **Administrators**, who configure and control contests, manage teams and problems, and monitor the system.

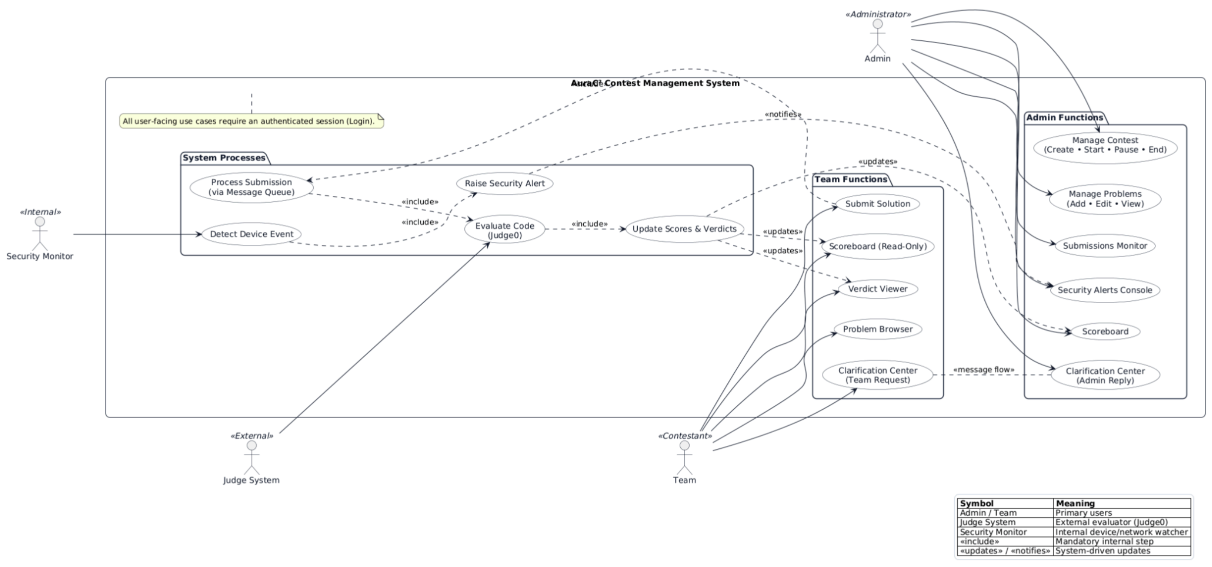
- **Teams**, who participate in contests, submit solutions, and view results in real time.

```
[Frontend (Team)]
   ↓
HTTP Request → /api/submissions
   ↓
[Spring Boot Controller]
   ↓
Publish message → [RabbitMQ Queue: "submissions"]
   ↓
[Submission Service Worker]
   ↓
Sends to Judge0 → Waits for callback
   ↓
[Callback Handler] → Update Database + Notify UI
```

2.2 Use Case Diagram

The following diagram illustrates the main actors (Administrator, Team, Security Monitor, and Judge System) and their interactions with the AuraC² Contest Management System.

Figure 1: AuraC² Use Case Diagram



Symbol	Meaning
Admin / Team	Primary users
Judge System	External evaluator (Judge0)
Security Monitor	Internal device/network watcher
«include»	Mandatory internal step
«updates» / «notifies»	System-driven updates

This diagram highlights the relationships between actors and system functions, showing how administrators, teams, and internal services interact through core modules such as Contest Management, Submissions, Scoreboard, and Security Monitoring.

2.3 Use Case Descriptions

Use Case 1 — Submit Solution

Attribute	Description
Use Case Name:	Submit Solution
Actors:	Team (Contestant)
Preconditions:	The team must be logged in and the contest must be active.
Main Flow:	1. The team selects a problem from the contest list .2. The team uploads their solution file. 3. The system forwards the submission for evaluation. 4. The result is processed and displayed to the team.
Alternative Flows:	- If the contest is paused, submission is rejected .- If the file format is invalid, an error is displayed.
Postconditions:	Submission is recorded and verdict is displayed to both team and admin.

Use Case 2 — Manage Contest

Attribute	Description
Use Case Name:	Manage Contest
Actors:	Administrator
Preconditions:	Admin must be logged in.
Main Flow:	1. Admin creates or modifies a contest. 2. Admin can start, pause, or end the contest. 3. System updates all connected clients.
Alternative Flows:	- Attempting to start two contests simultaneously is prevented. - Invalid configuration triggers an error message.
Postconditions:	Contest state is updated (Running, Paused, Ended).

Use Case 3 — Clarification Center

Attribute	Description
Use Case Name:	Clarification Center
Actors:	Team (Requester), Administrator (Responder)
Preconditions:	Both actors are logged in; contest is active.
Main Flow:	1. Team submits a clarification question. 2. Admin reviews and replies. 3. System notifies the team of the response.
Alternative Flows:	- Admin can close irrelevant or duplicate clarifications.
Postconditions:	Clarification is saved with “Answered” or “Pending” status.

Use Case 4 — Security Alerts Console

Attribute	Description
Use Case Name:	Security Alerts Console
Actors:	Administrator, Security Monitor (System Process)
Preconditions:	Contest is active; monitoring service is running.
Main Flow:	1. Security Monitor detects suspicious activity. 2. The system raises a security alert. 3. Admin reviews and takes appropriate action.
Alternative Flows:	- False alerts can be dismissed manually. - If monitor fails, alerts are logged for later review.
Postconditions:	Security alert is stored and displayed in the admin console.

Use Case 5 — View Scoreboard

Attribute	Description
Use Case Name:	View Scoreboard
Actors:	Administrator, Team
Preconditions:	Contest must be running or recently finished.
Main Flow:	1. User opens the scoreboard interface. 2. System displays ranked results based on accepted problems .3. View updates automatically when new results arrive.
Alternative Flows:	- If no submissions are present, an empty scoreboard is shown. - During pause, updates are frozen.
Postconditions:	Scoreboard reflects the latest standings accurately.

2.4 Summary of Use Cases

Use Case	Actor(s)	Core Function
Submit Solution	Team	Submit code for judging
Manage Contest	Admin	Control contest lifecycle
Clarification Center	Team, Admin	Communicate contest questions
Security Alerts Console	Admin, System	Monitor suspicious activities
View Scoreboard	Admin, Team	Display rankings and results

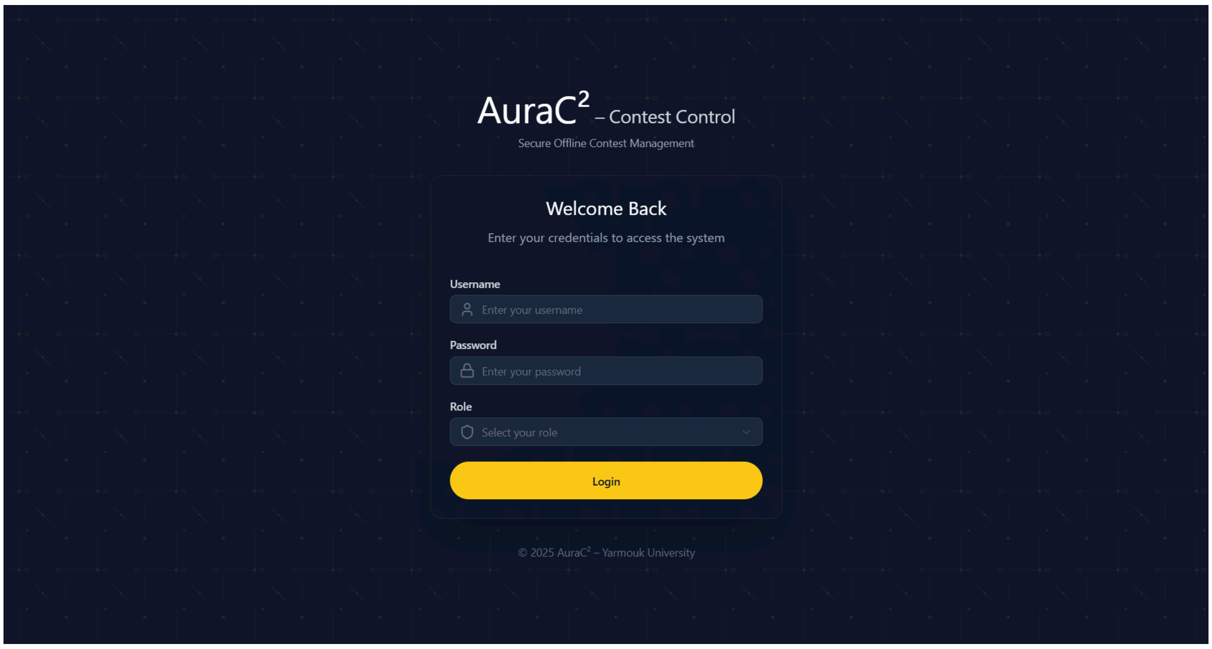
3. Initial User Interface Design

This section presents the main user interface layouts for the AuraC² Contest Control system. Each screen represents a key functional area as described in the system analysis and functional requirements.

Figure 3.1 – Login Interface

Description:
This screen allows users (Admins and Teams) to securely log in to the system.
It includes input fields for username and password, a role selector (Team/Admin), and a gold "Login" button on a dark blue background.
The design emphasizes clarity, accessibility, and brand consistency.

Screenshot:


AuraC^2 – Contest Control
Secure Offline Contest Management

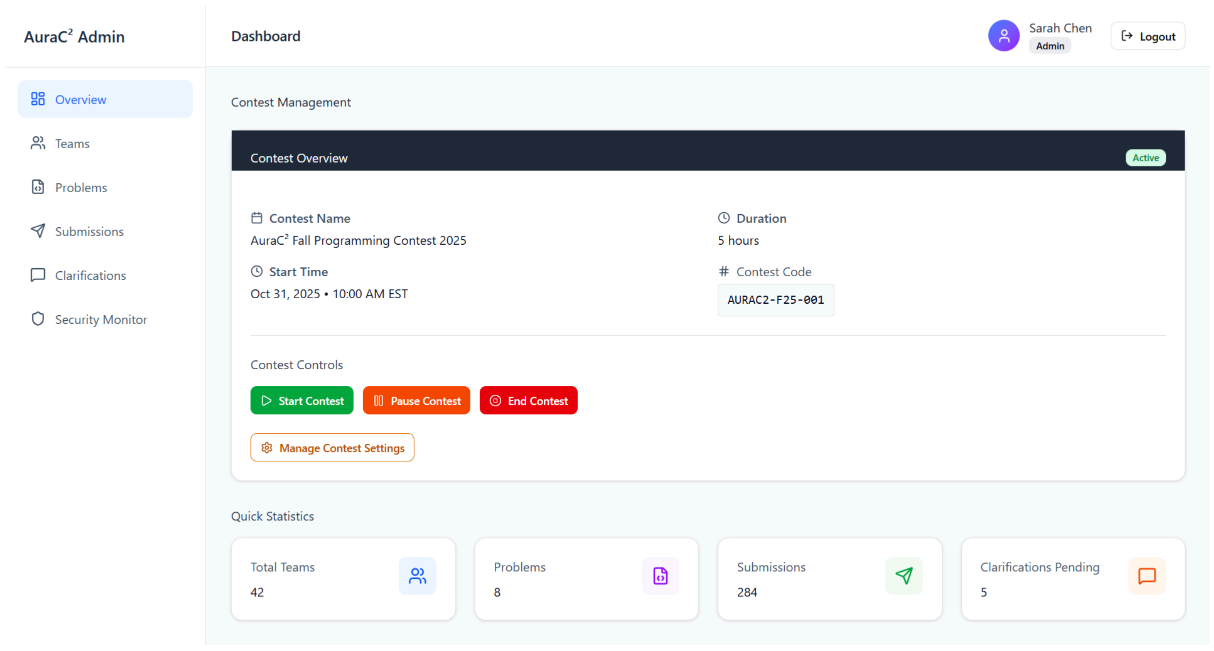
Welcome Back
Enter your credentials to access the system

© 2025 AuraC² – Yarmouk University

Figure 3.2 – Admin Dashboard

Description:
The Admin Dashboard provides an overview of contest status, teams, problems, and submissions.
A sidebar enables navigation, while quick stats and control buttons (Start, Pause, End Contest) appear at the top.
The layout follows modern admin dashboard principles with white cards and blue-gray headers.

Screenshot:



Contest Name
AuraC² Fall Programming Contest 2025

Duration
5 hours

Start Time
Oct 31, 2025 • 10:00 AM EST

Contest Code
AURAC2-F25-001

Total Teams 42

Problems 8

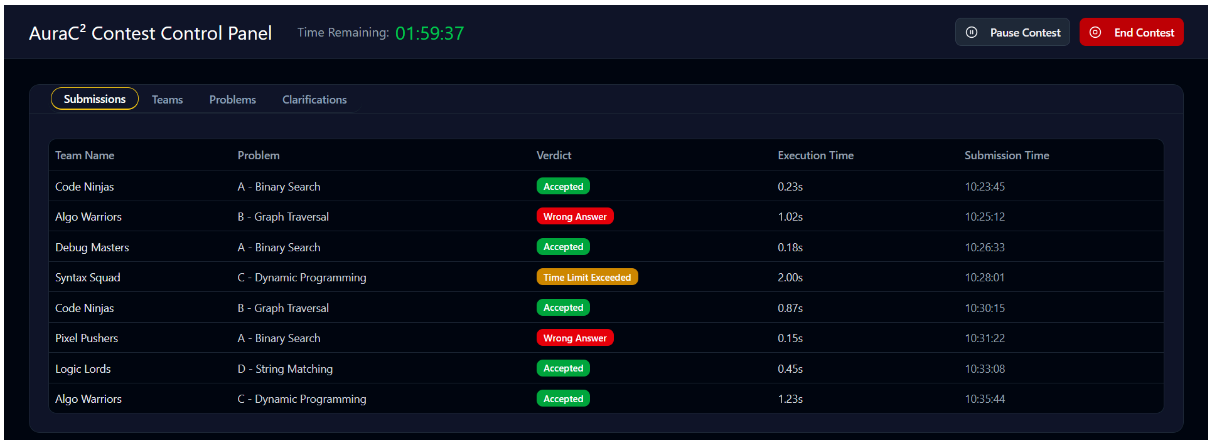
Submissions 284

Clarifications Pending 5

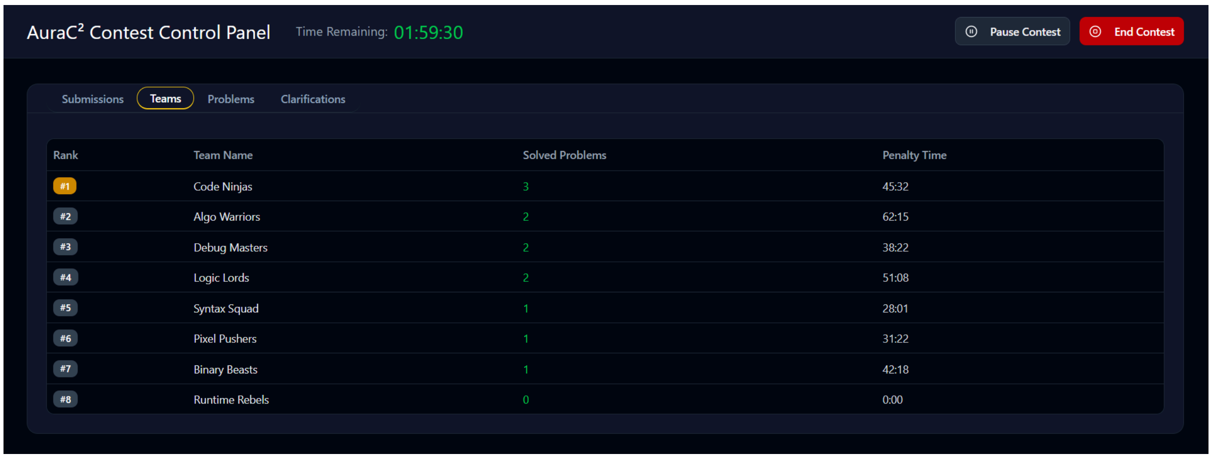
Figure 3.3 – Contest Control Panel

Description:
This screen enables the admin to monitor live submissions, teams, and clarifications during the contest.
It includes tabs (Submissions, Teams, Problems, Clarifications), a countdown timer, and control buttons.
The design is optimized for clarity during live judging.

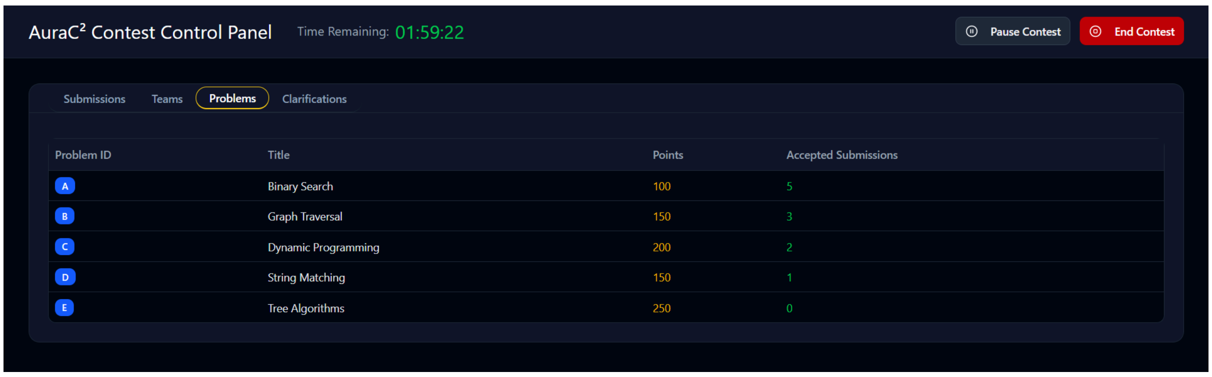
Screenshot:



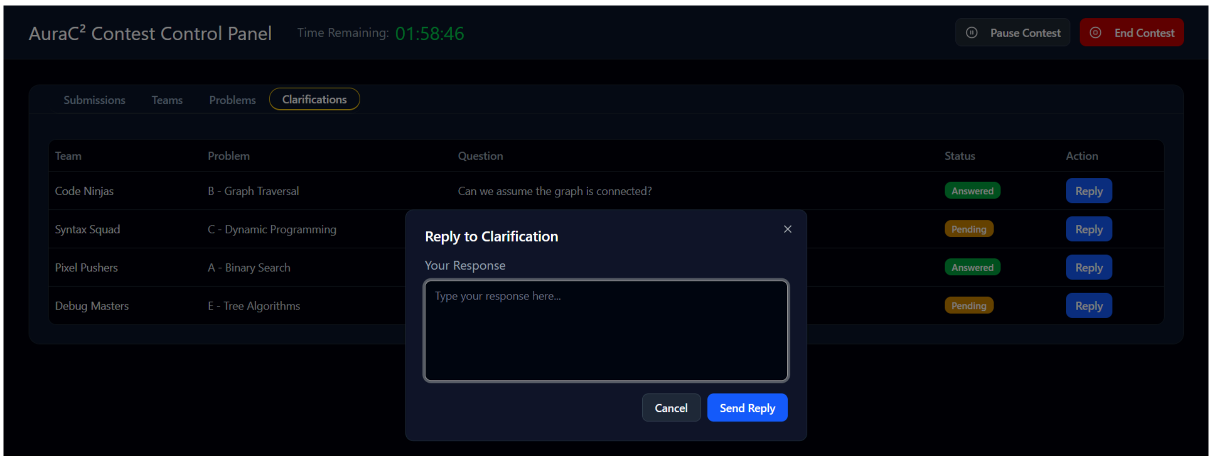
Team Name	Problem	Verdict	Execution Time	Submission Time
Code Ninjas	A - Binary Search	Accepted	0.23s	10:23:45
Algo Warriors	B - Graph Traversal	Wrong Answer	1.02s	10:25:12
Debug Masters	A - Binary Search	Accepted	0.18s	10:26:33
Syntax Squad	C - Dynamic Programming	Time Limit Exceeded	2.00s	10:28:01
Code Ninjas	B - Graph Traversal	Accepted	0.87s	10:30:15
Pixel Pushers	A - Binary Search	Wrong Answer	0.15s	10:31:22
Logic Lords	D - String Matching	Accepted	0.45s	10:33:08
Algo Warriors	C - Dynamic Programming	Accepted	1.23s	10:35:44



Rank	Team Name	Solved Problems	Penalty Time
#1	Code Ninjas	3	45:32
#2	Algo Warriors	2	62:15
#3	Debug Masters	2	38:22
#4	Logic Lords	2	51:08
#5	Syntax Squad	1	28:01
#6	Pixel Pushers	1	31:22
#7	Binary Beasts	1	42:18
#8	Runtime Rebels	0	0:00


AuraC^2 Contest Control Panel Time Remaining: 01:59:22

Problem ID	Title	Points	Accepted Submissions
A	Binary Search	100	5
B	Graph Traversal	150	3
C	Dynamic Programming	200	2
D	String Matching	150	1
E	Tree Algorithms	250	0


AuraC^2 Contest Control Panel Time Remaining: 01:58:46

Team	Problem	Question	Status	Action
Code Ninjas	B - Graph Traversal	Can we assume the graph is connected?	Answered	Reply
Syntax Squad	C - Dynamic Programming		Pending	Reply
Pixel Pushers	A - Binary Search		Answered	Reply
Debug Masters	E - Tree Algorithms		Pending	Reply

Reply to Clarification

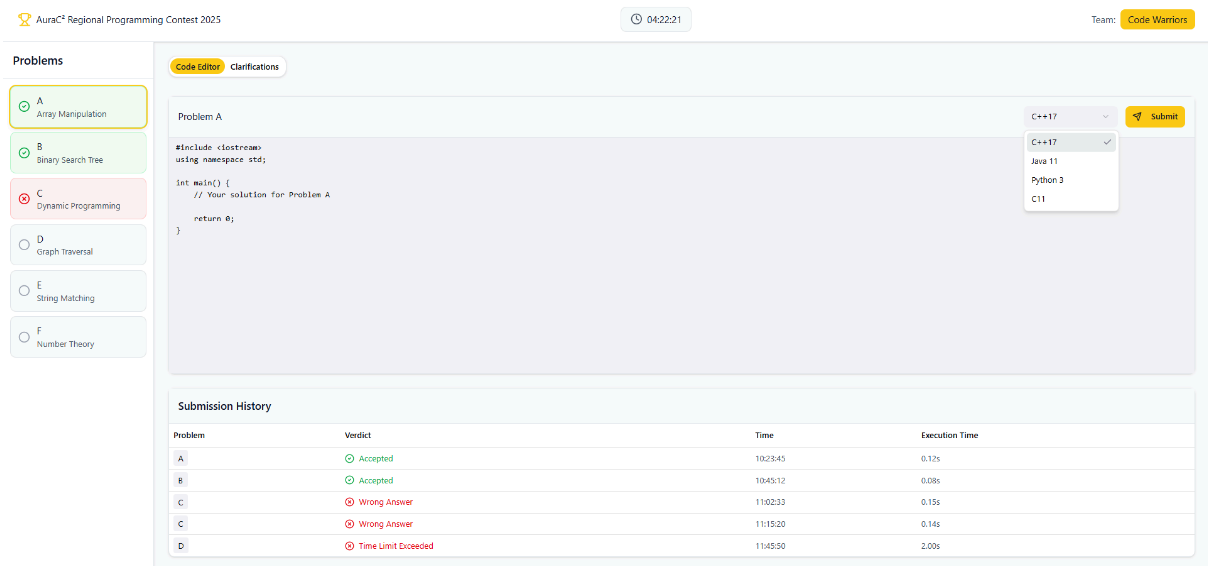
Your Response

Type your response here...

Figure 3.4 – Team Dashboard

Description:
The Team Dashboard replicates a real competitive programming environment, similar to PC² or Codeforces.
It includes a problem list, code editor area, submission history, and result feedback section.
The interface uses minimal distractions and efficient spacing for fast problem-solving.

Screenshot:



A	Array Manipulation
B	Binary Search Tree
C	Dynamic Programming
D	Graph Traversal
E	String Matching
F	Number Theory

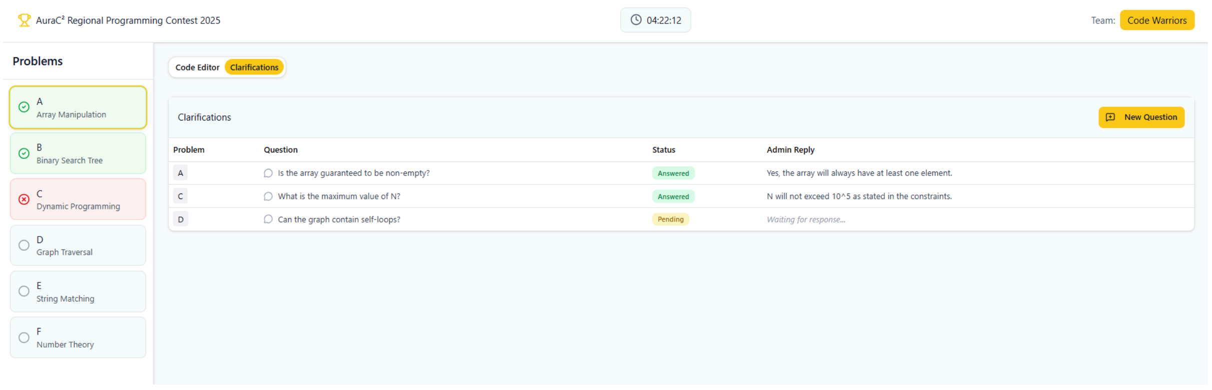
```
#include <iostream>
using namespace std;

int main() {
    // Your solution for Problem A

    return 0;
}
```

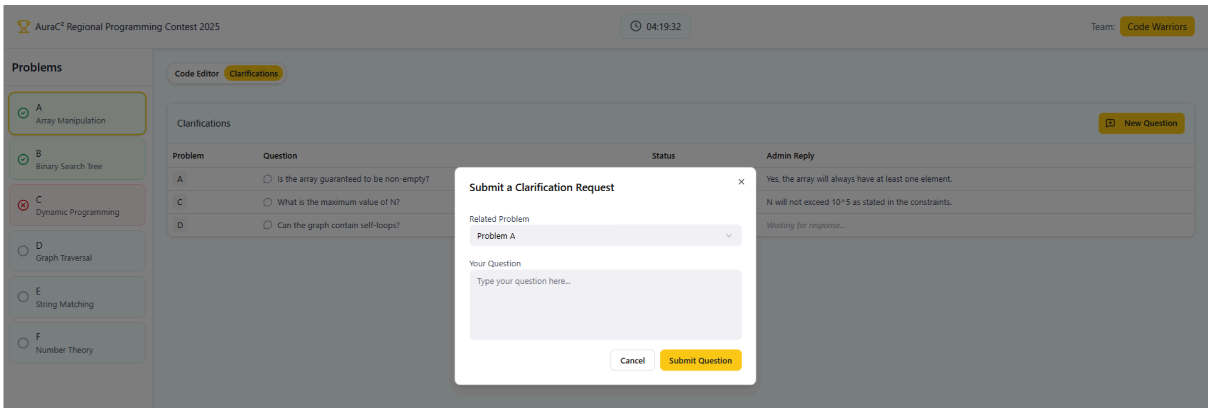
Submission History

Problem	Verdict	Time	Execution Time
A	Accepted	10:23:45	0.12s
B	Accepted	10:45:12	0.08s
C	Wrong Answer	11:02:33	0.15s
C	Wrong Answer	11:15:20	0.14s
D	Time Limit Exceeded	11:45:50	2.00s



Clarifications

Problem	Question	Status	Admin Reply
A	Is the array guaranteed to be non-empty?	Answered	Yes, the array will always have at least one element.
C	What is the maximum value of N?	Answered	N will not exceed 10^5 as stated in the constraints.
D	Can the graph contain self-loops?	Pending	*Waiting for response...*



Submit a Clarification Request

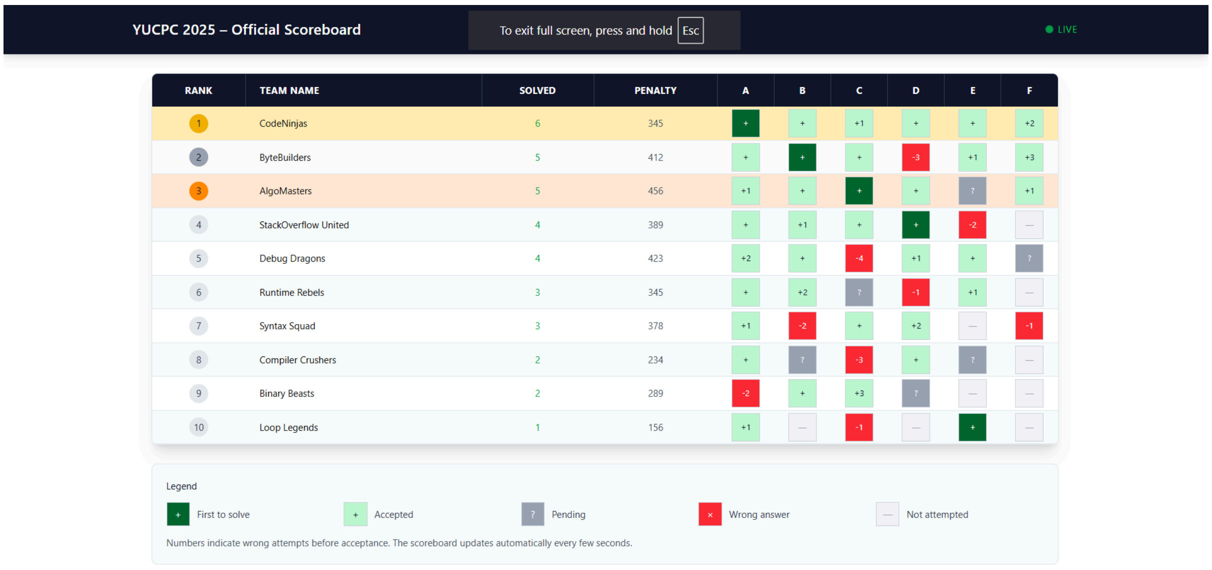
Related Problem: Problem A

Your Question: Type your question here...

Figure 3.5 – Real-Time Scoreboard

Description:
This scoreboard displays real-time rankings with problem-based verdict colors:
Dark green for first accepted, light green for accepted, red for wrong answer, gray for pending, and light gray for unattempted.
The design highlights the top three teams (gold, silver, bronze) and features a live update indicator.

Screenshot:



YUCPC 2025 – Official Scoreboard

To exit full screen, press and hold Esc

● LIVE

RANK	TEAM NAME	SOLVED	PENALTY	A	B	C	D	E	F
1	CodeNinjas	6	345	+	+	+1	+	+	+2
2	ByteBuilders	5	412	+	+	+	-3	+1	+3
3	AlgoMasters	5	456	+1	+	+	+	?	+1
4	StackOverflow United	4	389	+	+1	+	+	-2	—
5	Debug Dragons	4	423	+2	+	-4	+1	+	?
6	Runtime Rebels	3	345	+	+2	?	-1	+1	—
7	Syntax Squad	3	378	+1	-2	+	+2	—	-1
8	Compiler Crushers	2	234	+	?	-3	+	?	—
9	Binary Beasts	2	289	-2	+	+3	?	—	—
10	Loop Legends	1	156	+1	—	-1	—	+	—

Legend

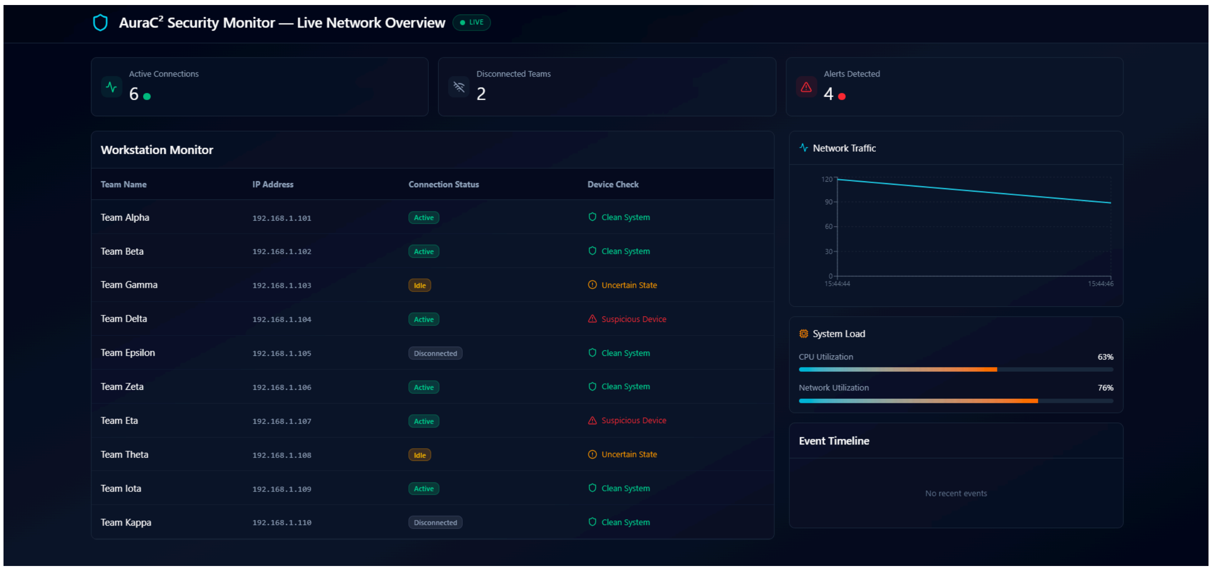
+	First to solve	+	Accepted	?	Pending	×	Wrong answer	—	Not attempted

Numbers indicate wrong attempts before acceptance. The scoreboard updates automatically every few seconds.

Figure 3.6 – Security Monitor Dashboard

Description:
This interface allows administrators to track network activity and connected workstations.
It displays a grid of teams with their IP addresses, connection status, and device alerts.
A side panel visualizes network traffic with real-time charts, while suspicious devices trigger red warning icons.

Screenshot:



AuraC² Security Monitor — Live Network Overview ● LIVE

Active Connections 6 ●

Disconnected Teams 2

Alerts Detected 4 ●

Workstation Monitor

Team Name	IP Address	Connection Status	Device Check
Team Alpha	192.168.1.101	Active	Clean System
Team Beta	192.168.1.102	Active	Clean System
Team Gamma	192.168.1.103	Idle	Uncertain State
Team Delta	192.168.1.104	Active	Suspicious Device
Team Epsilon	192.168.1.105	Disconnected	Clean System
Team Zeta	192.168.1.106	Active	Clean System
Team Eta	192.168.1.107	Active	Suspicious Device
Team Theta	192.168.1.108	Idle	Uncertain State
Team Iota	192.168.1.109	Active	Clean System
Team Kappa	192.168.1.110	Disconnected	Clean System

System Load

CPU Utilization 63%

Network Utilization 76%

Event Timeline

No recent events