**A blue and white logo

Description automatically generated**

Comsats University Islamabad

**Project Report: Capture The Flag (CTF) Management System**

January 16, 2025

Sadat ali

| FA23-BCT-034

Department of Computer Science

**Project Report: Capture The Flag (CTF) Management System**

**Table of Contents**

1. **Introduction**
   * Project Overview
   * Objectives
   * Scope
2. **System Design**
   * Architecture
   * Class Diagram
   * Use Case Diagram
3. **Features**
   * User Authentication
   * CTF Management
   * Challenge Management
   * Team Management
   * Scoreboard and Rankings
4. **Code Implementation**
   * Key Classes and Their Roles
   * Code Flow
   * Important Code Snippets
5. **Technologies Used**
   * Java
   * Swing (GUI)
   * Serialization (File I/O)
6. **Challenges Faced**
   * File Handling
   * Data Persistence
   * GUI Navigation
7. **Future Enhancements**
   * Database Integration
   * Advanced Security Features
   * Scalability
8. **Class Structures**
   * GUI Classes (Frontend)
   * Backend Classes (Model)
   * Utility Classes
   * Inner Classes
9. **Conclusion**
10. **ScreenShots**

**1. Introduction**

**Project Overview**

The **Capture The Flag (CTF) Management System** is a Java-based application designed to manage and organize CTF events. It allows users to register, create CTFs, upload challenges, form teams, and track scores. The system is built using **Object-Oriented Programming (OOP)** principles and features a **Graphical User Interface (GUI)** for ease of use.

**Objectives**

* To provide a platform for organizing and participating in CTF events.
* To manage challenges, teams, and scores efficiently.
* To ensure data persistence using file serialization.
* To create a user-friendly interface for seamless navigation.

**Scope**

The project is designed for CTF organizers and participants. It supports:

* User registration and authentication.
* Creation and management of CTFs.
* Uploading and solving challenges.
* Team formation and management.
* Real-time score tracking and rankings.

**2. System Design**

**Architecture**

The system follows the **Model-View-Controller (MVC)** architecture:

* **Model**: Backend classes (User, Player, Team, CTF, Challenge, Scoreboard).
* **View**: GUI panels (WelcomePanel, LoginPanel, CtfMenuPanel, etc.).
* **Controller**: AppController manages the flow between the view and model.

**3. Features**

**User Authentication**

* Users can register and log in using their email and password.
* Passwords are validated for strength.
* Users can log out, and their status is updated in the system.

**CTF Management**

* Organizers can create CTFs with a unique name and entry code.
* Participants can join CTFs using the entry code.
* CTFs can be deleted by the organizer.

**Challenge Management**

* Organizers can upload challenges with details like title, description, points, category, and flag.
* Participants can view and attempt challenges.
* Challenges can be downloaded for offline solving.

**Team Management**

* Players can form teams and add members.
* Team leaders can update team names and remove players.
* Teams can participate in CTFs and track their progress.

**Scoreboard and Rankings**

* The system maintains a scoreboard for each CTF.
* Teams are ranked based on their scores.
* Participants can view their rankings in real-time.

**4. Code Implementation**

**Key Classes and Their Roles**

1. **AppController**: Manages navigation between panels and handles interactions between the GUI and backend.
2. **User**: Represents a user in the system and handles authentication.
3. **Player**: Extends User and manages player-specific details like solved challenges and CTF participations.
4. **Team**: Represents a team and manages team members and their activities.
5. **CTF**: Represents a CTF event and manages challenges, teams, and the scoreboard.
6. **Challenge**: Represents a challenge in a CTF and handles file operations for challenge folders.
7. **Scoreboard**: Manages team scores and rankings for a CTF.

**Code Flow**

1. **User Registration**:
   * The user enters details in the RegistrationPanel.
   * The User class validates the input and saves the user to a file.
2. **User Login**:
   * The user enters credentials in the LoginPanel.
   * The User class verifies the credentials and logs the user in.
3. **CTF Creation**:
   * The organizer enters CTF details in the CtfRegisterPanel.
   * The CTF class creates the CTF and saves it to a file.
4. **Challenge Upload**:
   * The organizer uploads challenges in the CtfChallengesPanel.
   * The Challenge class saves the challenge and its folder.
5. **Team Formation**:
   * Players form teams in the TeamPanel.
   * The Team class manages team members and saves the team to a file.
6. **Score Tracking**:
   * Teams solve challenges, and their scores are updated in the Scoreboard.

**Important Code Snippets**

**User Registration**

public boolean register(String username, String email, String password) throws IOException, ClassNotFoundException {

if (validateCredentials(email)) {

return false; // Email already exists

}

this.userid = generateUserId();

setUsername(username);

setEmail(email);

setPassword(password);

setStatus(true);

return save(); // Save user to file

}

**CTF Creation**

public static CTF register(String CTFname, String EntryCode, User Organizer) {

CTF newCTF = new CTF(CTFname, EntryCode, Organizer);

ctfList.add(newCTF);

saveCTFList(ctfList); // Save CTF to file

return newCTF;

}

**Challenge Upload**

public boolean uploadChallenge() {

FolderManager manager = new FolderManager(CHALLENGE\_FOLDER\_PATH);

return manager.uploadFolder(this.getTitle()); // Upload challenge folder

}

**Team Formation**

public void addPlayer(Player player) throws IOException {

if (!players.contains(player)) {

players.add(player);

}

saveTeam(); // Save team to file

}

**Scoreboard Update**

public void updateTeamScore(Team team, int points) {

rankings.merge(team, points, Integer::sum); // Update team score

}

**5. Technologies Used**

**Java**

* The project is implemented in Java, leveraging its OOP features like inheritance, encapsulation, and polymorphism.

**Swing (GUI)**

* The GUI is built using Java Swing, providing a user-friendly interface for navigation.

**Serialization (File I/O)**

* Data persistence is achieved using Java serialization, allowing objects to be saved to and loaded from files.

**6. Challenges Faced**

**File Handling**

* Managing file operations for serialization and deserialization was challenging, especially for large datasets.

**Data Persistence**

* Ensuring data consistency across multiple files required careful handling of file I/O operations.

**GUI Navigation**

* Implementing smooth navigation between panels using CardLayout required careful design and testing.

**7. Future Enhancements**

**Database Integration**

* Replace file-based storage with a database (e.g., MySQL) for better scalability and performance.

**Advanced Security Features**

* Implement password hashing and encryption for secure storage of user credentials.

**Scalability**

* Add support for larger CTF events with more participants and challenges.

**8. Class Structures**

**GUI Classes (Frontend)**

These classes handle the **Graphical User Interface (GUI)** and user interactions:

1. **AppController.java**
   * Manages navigation between panels and controls the flow of the application.
2. **WelcomePanel.java**
   * Displays the welcome screen with options to log in or register.
3. **LoginPanel.java**
   * Handles user login functionality.
4. **RegistrationPanel.java**
   * Handles user registration functionality.
5. **CtfMenuPanel.java**
   * Displays the main menu for CTF-related actions (create, organize, play CTFs).
6. **CtfRegisterPanel.java**
   * Allows users to organize and register a new CTF.
7. **CtfChallengesPanel.java**
   * Allows organizers to upload challenges for a CTF.
8. **CtfCreatorPanel.java**
   * Allows users to create challenges for a CTF.
9. **CtfDeletePanel.java**
   * Allows organizers to delete a CTF.
10. **CtfScoreboaredPanel.java**
    * Displays the scoreboard for a CTF.
11. **CtfTeamsPanel.java**
    * Displays the teams participating in a CTF.
12. **PlayChallengePanel.java**
    * Displays available challenges for players to solve.
13. **PlayCtfPanel.java**
    * Displays available CTFs for players to join.
14. **CtfPlayChallengesPanel.java**
    * Displays challenges for a specific CTF that players can solve.
15. **CtfPlayScoreboaredPanel.java**
    * Displays the scoreboard for a specific CTF.
16. **CtfPlayTeamsPanel.java**
    * Displays the teams participating in a specific CTF.
17. **PlayerPanel.java**
    * Displays player information and provides navigation to CTFs, challenges, and team management.
18. **TeamPanel.java**
    * Manages team-related actions (changing team name, adding/removing players).
19. **PanelUtils.java**
    * Provides utility methods for creating styled components (buttons, labels, panels).

**Backend Classes (Model)**

These classes handle the **business logic** and **data management**:

1. **User.java**
   * Represents a user in the system and handles authentication (login, registration, logout).
2. **Player.java**
   * Extends User and represents a player with additional details like solved challenges and CTF participations.
3. **Team.java**
   * Represents a team and manages team members and their activities.
4. **CTF.java**
   * Represents a CTF event and manages challenges, teams, and the scoreboard.
5. **Challenge.java**
   * Represents a challenge in a CTF and handles file operations for challenge folders.
6. **Scoreboard.java**
   * Manages team scores and rankings for a CTF.
7. **FolderManager.java**
   * Handles file operations for uploading and downloading challenge folders.
8. **CreateDatabase.java**
   * Creates a MySQL database for the application (if needed).

**Utility Classes**

These classes provide **helper functionality**:

1. **FormPanel.java**
   * A reusable component for creating forms with labels and text fields.
2. **MainRunner.java**
   * The entry point of the application, initializes the main window and GUI.

**Inner Classes**

These are **nested classes** used within other classes:

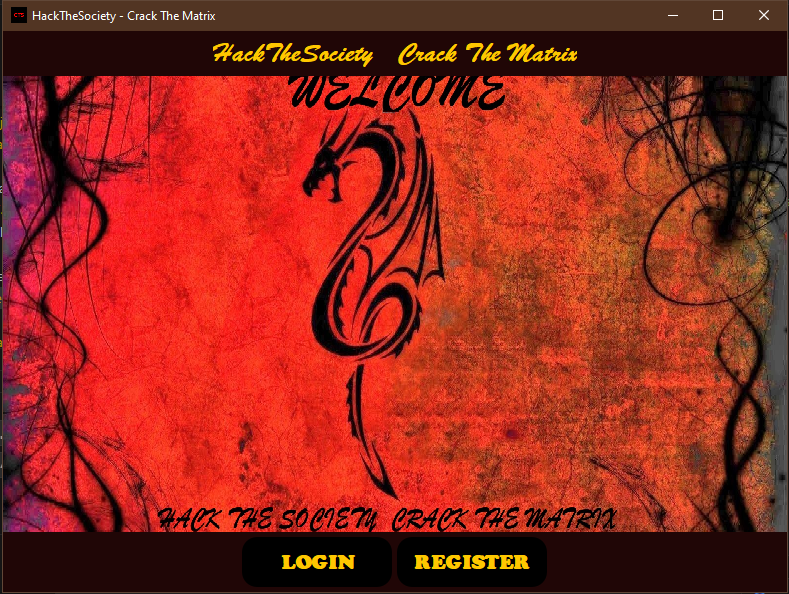
1. **Player.SolvedChallenge.java**
   * Represents a solved challenge by a player (used in Player.java).
2. **Player.CTFParticipation.java**
   * Represents a player's participation in a CTF (used in Player.java).

**9. Conclusion**

The **Capture The Flag (CTF) Management System** is a robust application that simplifies the organization and participation in CTF events. By leveraging Java's OOP features and Swing for the GUI, the project provides a seamless user experience. With future enhancements like database integration and advanced security features, the system can be further improved to meet the needs of larger and more complex CTF events.

1. **ScreenShot:**

**WELCOME SCREEN:**

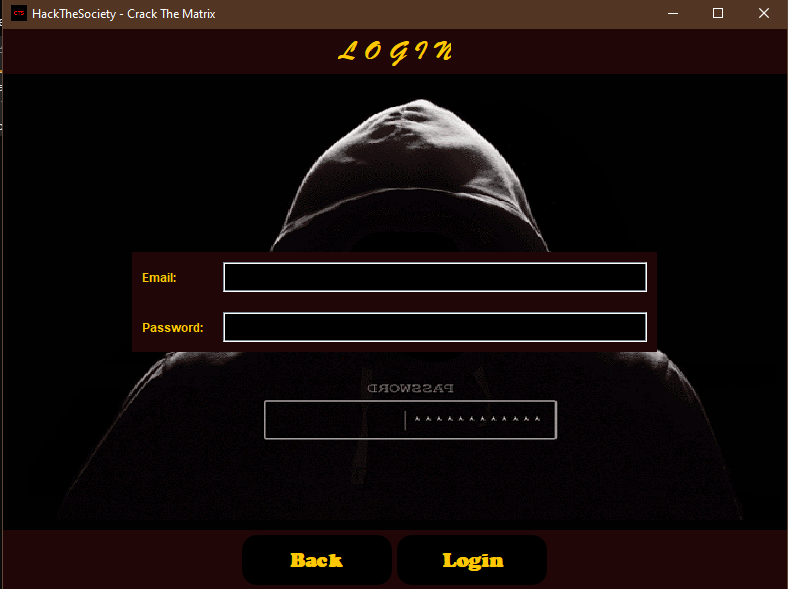
****

**REGISTRATION SCREEN:**

**A screenshot of a computer

Description automatically generated**

**LOGIN:**

****

**Home Screen:**

**A screenshot of a computer

Description automatically generated**

**CREATE CTF CHALLENGE AND UPLOAD:**

****

**A screenshot of a computer

Description automatically generated**

**PLAY SCREEN:**

****

**TEAM MANGEMENT SCREEN:**

**A screenshot of a computer

Description automatically generated**

**PLAY CHALLENGES INDIVIDUALLY:**

**A screenshot of a game

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**PLAY CTF COMPITITIONS AS TEAM:**

****

**A screenshot of a video game

Description automatically generated**

**A screenshot of a computer game

Description automatically generated**

**ORGANIZE CTF COMPITAION:**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**And there are many more..**