AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

Dept. of Computer Science
Faculty of Science and Technology

CSC2210: OBJECT ORIENTED PROGRAMMING 2

FALL 2025-2026

Section: [H]

Group No: 10

Project Report On

Project Name: **AIUB LOST AND FOUND MANAGEMENT**

Supervised by Md. Hasibul Hasan

Submitted By:

Name	ID
1. RAHUL DEV SHAHA	**24-58397-2**

Obtained Marks for CO2 and CO3 (Description given in the following page)					
Assessment Criteria	**Not Attended/ Incorrect (0)**	**Inadequate (1-2)**	**Average (3)**	**Good (4)**	**Excellent (5)**

Evaluation Criteria (CO2)	**Total =**	**Evaluation Criteria (CO3)**	**Total =**
Requirement fulfillment		Organization of the application	
Validation		Representation and Integration of Database	
Verification		Graphical User Interface	

CO2: Display and verify the mean of a real-life Project using the concepts of C# Graphical User Interface based environment with database integration to depict a desktop-based application.					
Assessment Criteria	**Not Attended/ Incorrect (0)**	**Inadequate (1-2)**	**Average (3)**	**Good (4)**	**Excellent (5)**
Evaluation Criteria	**Evaluation Definition**				
Requirement fulfillment	Fails to demonstrate any understanding of real-life scenario-based project development or functional requirement identification. There is no attempt to depict a project or identify functional requirements accurately.	Demonstrates limited understanding of real-life scenario-based project development and functional requirement identification. The project depicted lacks coherence or relevance to real-life scenarios, and functional requirements are inaccurately identified or insufficiently described.	Presents a basic depiction of a real-life scenario-based project and identifies some functional requirements. However, the project lacks depth or complexity, and some functional requirements may be vaguely defined or missing key details.	Effectively demonstrates a realistic scenario-based project and accurately identifies most functional requirements. The project is well-developed with appropriate complexity, and functional requirements are clearly articulated with relevant details.	Exhibits an exceptional understanding of real-life scenario-based project development and accurately identifies all functional requirements. The project is meticulously developed with thorough attention to detail, reflecting a comprehensive understanding of Object-Oriented Programming project development activities.
Validation	Fails to demonstrate any understanding or implementation of validation forms in their system. There is no attempt to deal with data validation, and validation requirements are completely ignored or incorrectly applied.	Demonstrates limited understanding of validation forms and data validation techniques. While some attempt may be made to implement validation, it is incomplete or poorly executed, leading to inadequate handling of data validation.	Shows a basic understanding of validation forms and data validation techniques. They attempt to implement validation, but some aspects may be missing or incorrectly implemented, resulting in partial or inconsistent handling of data validation.	Effectively demonstrates the use of validation forms and implements data validation techniques. Validation is mostly accurate and comprehensive, ensuring the proper handling of data input and verification in the system.	Exhibits an exceptional understanding and implementation of validation forms and data validation techniques. Validation is meticulously implemented with thorough attention to detail, ensuring robust data validation procedures and contributing to the overall reliability and integrity of the system.
Verification	Fails to demonstrate any attempt to verify the system data or functional requirements. There is no evidence of	Demonstrates limited understanding of verification processes and data flow in the system. Verification	Shows a basic understanding of verification processes and attempts to verify system data. However, verification	Identifies and verifies system data, ensuring proper functional requirements are met. Verification efforts are mostly accurate and	Exhibits an exceptional understanding of verification processes and meticulously verifies system data. Verification

	understanding or implementation of verification processes, and data flow is not considered.	attempts are incomplete or inaccurate, and there is insufficient consideration given to ensuring data integrity and functionality.	efforts may be inconsistent or lack thoroughness, and there may be gaps in ensuring proper functional requirements and data flow.	thorough, with attention to ensuring data integrity and appropriate data flow within the system.	efforts are comprehensive and precise, with a keen focus on ensuring all functional requirements are met and maintaining proper data flow throughout the system.

CO3: Prepare and Explain a real life desktop based application synthesizing several component of C# along with development tools to adhere the given requirements.					
Assessment Criteria	**Not Attended/ Incorrect (0)**	**Inadequate (1-2)**	**Average (3)**	**Good (4)**	**Excellent (5)**
Evaluation Criteria	**Evaluation Definition**				
Organization of the application	Fails to identify any suitable real time application or requirements for project development activities related to OOP.	Limited understanding about the project scopes and scenarios or identification of functional requirements.	Lacks depth or relevance to OOP project development activities and may contain inaccuracies. Real-life scenarios are mentioned, but the discussion lacks depth or clarity.	Consider and integrate the idea of several core aspects of the project along with relevance to real-life scenarios. Demonstrating a solid understanding of the application presentation.	Generalize and exhibits an exceptional understanding of project preparation according to a to real-life scenarios. Also contains proper and insightful identification of the system which is comprehensive and precise.
Representation and Integration of Database	Fails to identify and present any understanding or implementation of database. Also failed to integrate the data with the project itself.	Limited understanding of the database concepts or their proper way of using in a real time project. While some attempt may be made to implement but it is incomplete or poorly executed, leading to inadequate design.	Lacks depth or relevance to database integration with the application. Shows a basic understanding but some aspects may be missing or incorrectly implemented, resulting in partial or inconsistency. May lack proper normalization.	Integrate the database with the forms properly and implements it with proper validation which is mostly accurate and comprehensive, ensuring the proper handling of data input and verification along with general normalization.	Exhibits an exceptional understanding and implementation of database ensuring attention to detail, and robust data manipulation procedures and contributing to the overall clarity.
Graphical User Interface	Fails to present or prepare GUI based application interfaces. There is no evidence of creating or integrating such things according to their usefulness.	Limited understanding of graphical user interfaces. Lack of design knowledge. Very poor attempt to make such things which are currently obsolete	Shows a basic understanding of creating user interfaces. Most of them are interconnected but maybe some of them lack it. However, most of it can be	Effectively identifies and meet the consider the simplicity. Design related works are mostly accurate and taken proper attention to ensuring a user-	Exhibits an exceptional work design following a high standard of simple and elegant work. Several controls and mechanism has been organized in a

		or can’t be identified as coherent.	described as user friendly.	friendly coherent system.	preferred way according to the coherent usage .

Table of Contents: Page no.

Abstract

AIUB Lost & Found System is a Windows-based application designed to help students and staff to report and find lost items within the university campus. Users can easily create lost and found reports, then Employee of lost and found can search for items, and view possible matches.

The application provides secure login and role-based access for Admin and Staff to ensure proper system control. Admins can manage users and reports, while Staff can handle item records.

The system is developed using C#(.Net Framework) and MS SQL Server, ensuring fast performance, data security, and reliable data storage. This application saves time increasing the chances of recovering lost belongings.

Introduction

Losing personal items such as mobile phones, ID cards, books, or bags is a common problem in university environments. Traditional lost and found systems usually depend on notice boards or word-of-mouth, which are inefficient, time-consuming, and unreliable. Many lost items remain unclaimed due to poor communication and lack of proper tracking.

The AIUB Lost & Found System is a Windows-based software solution designed to solve these problems by providing a digital platform for reporting and managing lost and found items. The system allows users to submit item reports, search records, and find possible match based on item details.

This application supports secure user authentication and role-based access, ensuring that only authorized users can perform specific actions. Admins can manage users, monitor reports, and view confirmed matches, while staff members can create and update reports. The system also includes a smart matching feature that user can compares item details to find items.

Developed using C# and MS SQL Server, the AIUB Lost & Found System ensures reliable data storage, fast processing, and secure operations. The goal of this system is to make the lost and found process more organized, transparent, and efficient, helping students and staff recover their lost belongings easily.

Features

User Authentication
• Secure login with role-based access (Admin & Staff).

Lost & Found Reports
• Create, edit, and delete lost/found item reports.

Match System
• Possible matches Check based on item details.
• Matches are confirmed by Admin/Staff.

Search & Filter
• Search and filter reports by item and user details.

Status Tracking
• Report status updates automatically (Pending / Matched).

Match History
• Stores all confirmed match records.

Admin Control
• Admin manages users, reports, and system activities.

User Stories

Admin User Stories

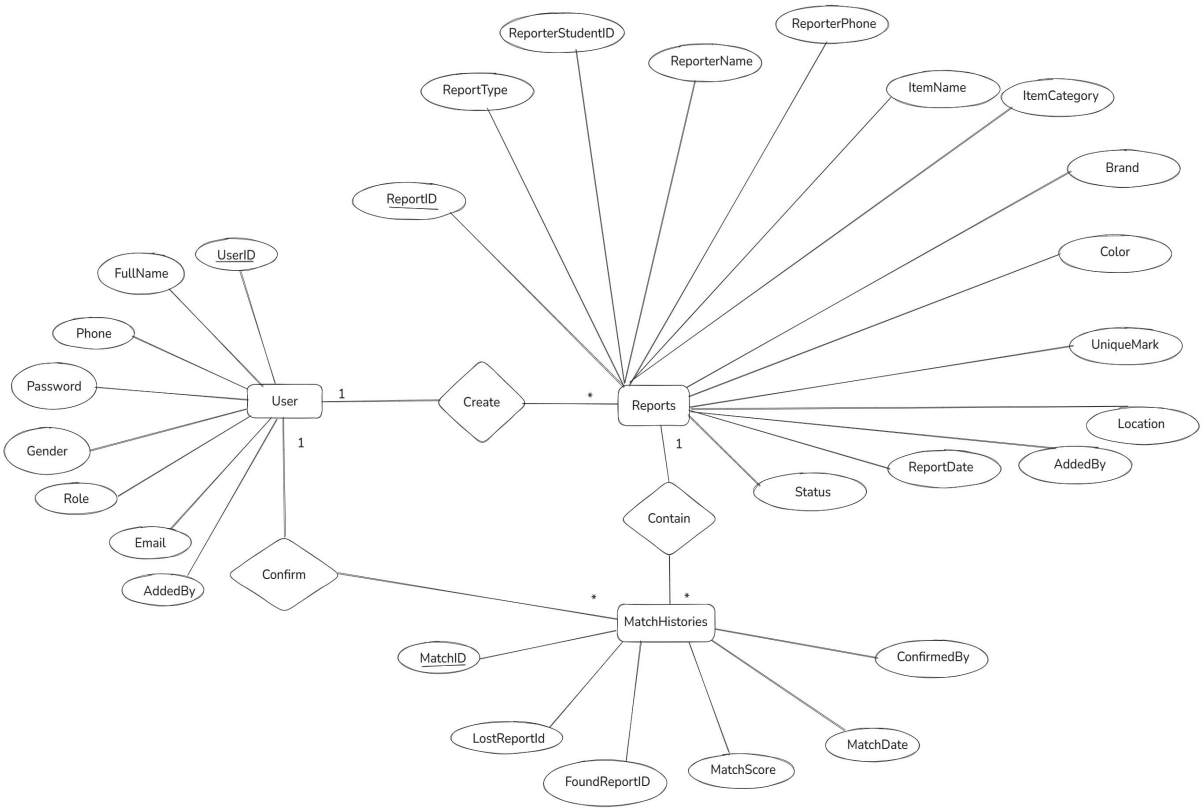
• As an admin, I can log in securely to access the system.
• As an admin, I can add, edit, and remove, search users.
• As an admin, I can view and search all lost and found reports.
• As an admin, I can confirm matches between lost and found items.
• As an admin, I can view all confirmed match history.
• As an admin, I can delete invalid or fake reports.
• As an admin, I can monitor overall system activities.

Staff User Stories

• As a staff member, I can log in securely.
• As a staff member, I can create lost and found reports.

• As a staff member, I can edit existing reports.
• As a staff member, I can delete reports if needed.
• As a staff member, I can search reports using filters.
• As a staff member, I can view possible matches if I need.
• As a staff member, I can help confirm item matches.

ER Diagram



Users - Reports (Create): One to Many

UNF
UserID, UserName, UserEmail, UserPhone, UserPassword, UserRole, UserGender, ReportType, ReporterName, ReporterStudentID, ReporterPhone, ItemName, ItemCategory, Color, brand, UniqueMark, ReportDate, Location, Description, Status, ReportID

1NF
UserID(pk), UserName, UserEmail, UserPhone, UserPassword, UserRole, UserGender, ReportType, ReporterName, ReporterStudentID, ReporterPhone,

ItemName, ItemCategory, Color, brand, UniqueMark, ReportDate, Location, Description, Status, ReportID(pk)

2NF
Users:
UserID (PK), FullName, Email, Phone, Password, Role, Gender

Reports:
ReportID (pk), ReportType, ReporterName, ReporterStudentID, ReporterPhone, ItemName, ItemCategory, Color, brand, UniqueMark, ReportDate, Location, Description, Status
AddedBy (UserID-FK)

3NF Result
Users - 3NF
Reports - 2NF (Because Reporters has no Direct access to System)

Reports - MatchHistories(Contain): One to Many

UNF
ReportID , ReportType, ReporterName, ReporterStudentID, ReporterPhone, ItemName, ItemCategory, Color, brand, UniqueMark, ReportDate, Location, Description, Status, ReportID, LostReportID, FoundReportID, MatchScore, MatchDate

1NF
MatchID (pk), ReportType, ReporterName, ReporterStudentID, ReporterPhone, ItemName, ItemCategory, Color, brand, UniqueMark, ReportDate, Location, Description, Status, ReportID(pk), MatchScore, MatchDate

2NF
Reports:
ReportType, ReporterName, ReporterStudentID, ReporterPhone, ItemName, ItemCategory, Color, brand, UniqueMark, ReportDate, Location, Description, Status, ReportID(pk)

MatchHistories:
MatchID (PK), LostReportID (FK), FoundReportID (FK), MatchScore, MatchDate

3NF Result:
Reports - 2NF (Because Reporters has no Direct access to System)
MatchHistories - 3NF

Users - MatchHistories (Confirm): One to Many

UNF
UserID, UserName, UserEmail, UserPhone, UserPassword, UserRole, UserGender, MatchScore, MatchDate

1NF
MatchID (PK), UserID(PK),UserName, UserEmail, UserPhone, UserPassword, UserRole, UserGender, MatchScore, MatchDate

2NF
Users:
UserID (PK), FullName, Email, Phone, Password, Role, Gender

MatchHistories:
MatchID (PK), LostReportID (FK-ReportID), FoundReportID (FK- ReportID), MatchScore, MatchDate, ConfirmedBy (FK-UserID)

3NF Result:
Users - 3NF
MatchHistories - 3NF

Duplicate Findout
Users → Appeared in Step 1 & Step 3 (Duplicate)
Reports → Appeared in Step 1 & Step 2 (Duplicate)
MatchHistories → Appeared in Step 2 & Step 3 (Duplicate)

Final Output Tables
Users:
UserID (PK), FullName, Email, Phone, Password, Role, Gender

Reports:
ReportID (PK), ReportType, ReporterName, ReporterStudentID, ReporterPhone, ItemName, ItemCategory, Color, brand, UniqueMark, ReportDate, Location, Description, Status

MatchHistories:
MatchID(PK), LostReportID (FK), FoundReportID(FK), MatchScore, MatchDate, ConfirmedBy (FK)

SQL Queries

1. LoginForm

```
string query = "SELECT * FROM Users WHERE Email = '" + email + "' AND Password = '" + password + "'";
```

2. Inner Join (AdminDashboardForm + StaffDashboardForm)

```
    string query = @"SELECT
l.ReporterName AS 'Lost Reporter Name',l.ReporterPhone AS 'Lost Reporter Phone',f.ReporterName AS 'Found Reporter Name',f.ReporterPhone AS 'Found Reporter Phone',l.ItemName AS 'Lost Item',l.UniqueMark AS 'Unique Mark',mh.MatchScore AS 'Match Score',mh.MatchDate AS 'Match Date',u.FullName AS 'Confirmed By'FROM MatchHistories mhJOIN Reports l ON mh.LostReportID = l.ReportIDJOIN Reports f ON mh.FoundReportID = f.ReportIDJOIN Users u ON mh.ConfirmedBy = u.UserIDORDER BY mh.MatchDate DESC;";
```

3. Search/Fiter matched data

```
string query = @"SELECT     mh.MatchID,    l.ReporterName AS LostReporter, l.ReporterPhone,    l.ItemName AS LostItem,    f.ItemName AS FoundItem, mh.MatchScore,    mh.MatchDate,    u.FullName AS ConfirmedByFROM MatchHistories mhJOIN Reports l ON mh.LostReportID = l.ReportIDJOIN Reports f ON mh.FoundReportID = f.ReportIDJOIN Users u ON mh.ConfirmedBy = u.UserIDWHERE 1=1";

        if (phone != "")
            query += $" AND l.ReporterPhone LIKE '%{phone}%'";
```

4. ManageReportForm (Delete)

```
query = $"DELETE FROM Reports WHERE ReportID = {selectedReportId}";
```

5. ReportManageForm (Filter/Search)

```
 string query = @" SELECT      r.ReportID AS 'Report Id',     r.ReportType AS 'Report Type',     r.ItemName AS 'Item Name',     r.ItemCategory AS Category,     r.Brand AS Brand,     r.Color AS Color,     r.UniqueMark AS 'Unique Mark',     r.Location AS Location,     r.ReportDate AS 'Report Date',     r.Status AS Staus, r.ReporterName AS 'Reporter Name',     r.ReporterStudentID 'Reporter ID', r.ReporterPhone 'Phone',     u.FullName AS AddedBy FROM Reports r, Users u WHERE r.AddedBy = u.UserID ";

         if (!string.IsNullOrEmpty(name))
             query += $" AND r.ReporterName LIKE '%{name}%'";

         if (!string.IsNullOrEmpty(sid))
             query += $" AND r.ReporterStudentID LIKE '%{sid}%'";

         if (!string.IsNullOrEmpty(phone))
```

```
            query += $" AND r.ReporterPhone LIKE '%{phone}%'";

        if (!string.IsNullOrEmpty(status) && status!= "All")
            query += $" AND r.Status= '{status}'";

        query += " ORDER BY r.ReportDate DESC";
```

6. Load Report

```
    string query = @"SELECT     r.ReportID,    r.ReportType,    r.ItemName,
r.ItemCategory,    r.Brand,    r.Color,    r.UniqueMark,    r.Location,
r.ReportDate,    r.Status,
    r.ReporterName,    r.ReporterStudentID,    r.ReporterPhone,
    u.FullName AS AddedByFROM Reports r, Users uWHERE r.AddedBy = u.UserIDORDER BY
r.ReportDate DESC";
```

7. Fetching For Edit

```
string query = $"SELECT * FROM Reports WHERE ReportID = {selectedReportId}";
```

8. Add New Report

```
 query = $@"INSERT INTO Reports(ReportType, ReporterName, ReporterStudentID,
ReporterPhone, ItemName, ItemCategory, Brand, Color, UniqueMark, Description,
Location, Status, AddedBy)VALUES('{reportType}', '{reporterName}',
'{reporterStudentID}', '{reporterPhone}', '{itemName}', '{itemCategory}', '{brand}',
'{color}', '{uniqueMark}', '{description}', '{location}', 'Pending',
{filledByUserId})";
```

9. Update Report

```
query = $@"UPDATE Reports SET    ReportType = '{reportType}',    ReporterName =
'{reporterName}',    ReporterStudentID = '{reporterStudentID}',    ReporterPhone =
'{reporterPhone}',    ItemName = '{itemName}',    ItemCategory = '{itemCategory}',
Brand = '{brand}',    Color = '{color}',    UniqueMark = '{uniqueMark}',
Description = '{description}',    Location = '{location}'WHERE ReportID =
{selectedReportId}";
```

10. Delete Report

```
string query = $"DELETE FROM Reports WHERE ReportID = {selectedReportId}";
```

11. Load Selected Report

```
 string query = $"SELECT * FROM Reports WHERE ReportID = {selectedReportId}";
```

12. Opposite Type Report

```
       string query = $"SELECT * FROM Reports WHERE ReportType = '{oppositeType}' AND
Status = 'Pending'";
```

13. Insert Match

```
 string insertQuery = $@"INSERT INTO MatchHistories(LostReportID, FoundReportID,
MatchScore, MatchDate, ConfirmedBy)VALUES({lostId}, {foundId}, {selectedMatchScore},
GETDATE(), {loggedInUserId})";
```

14. Update LostId

```
 string updateQuery1 = $"UPDATE Reports SET Status = 'Matched' WHERE ReportID =
{lostId}";
```

15. Update FoundId

```
 string updateQuery2 = $"UPDATE Reports SET Status = 'Matched' WHERE ReportID =
{foundId}";
```

16. Fetching Users

```
string query = @"SELECT u.UserID AS 'User ID',u.FullName AS 'Full Name',u.Email AS
'Email',u.Phone AS Phone,u.Role AS Role,u.Gender AS Gender,ISNULL(a.FullName,
'System') AS 'Added By'FROM Users uLEFT JOIN Users a ON u.AddedBy = a.UserID;";
```

16.Search/Filter (Conditional Query)

```
    string query = @"SELECT     u.UserID,    u.FullName,    u.Email,    u.Phone,
u.Role,    u.Gender,    ISNULL(a.FullName, 'System') AS 'Added By'FROM Users uLEFT
JOIN Users a ON u.AddedBy = a.UserIDWHERE 1=1";


    if (name != "")
        query += $" AND u.FullName LIKE '%{name}%'";


    if (email != "")
        query += $" AND u.Email LIKE '%{email}%'";


    if (role != "" && role != "Select")
        query += $" AND u.Role = '{role}'";


    if (gender != "" && gender != "Select")
        query += $" AND u.Gender = '{gender}'";
```

17. Adding Neew User

```
query = $@"INSERT INTO Users (FullName, Email, Phone, Password, Role, Gender, AddedBy)
VALUES ('{fullName}', '{email}', '{phone}', '{password}', '{role}', '{gender}',
{loggedInUserId})";
```

18. Update User without password

```
    query = $@"UPDATE UsersSET     FullName = '{fullName}',    Email = '{email}',
Phone = '{phone}',    Role = '{role}',    Gender = '{gender}'WHERE UserID =
{selectedUserId}";
```

19. Update user With Password

```
    query = $@" UPDATE UsersSET     FullName = '{fullName}',    Email = '{email}',
Phone = '{phone}',    Password = '{password}',    Role = '{role}',    Gender =
'{gender}'WHERE UserID = {selectedUserId}";
```

20. Fetching updatebale data

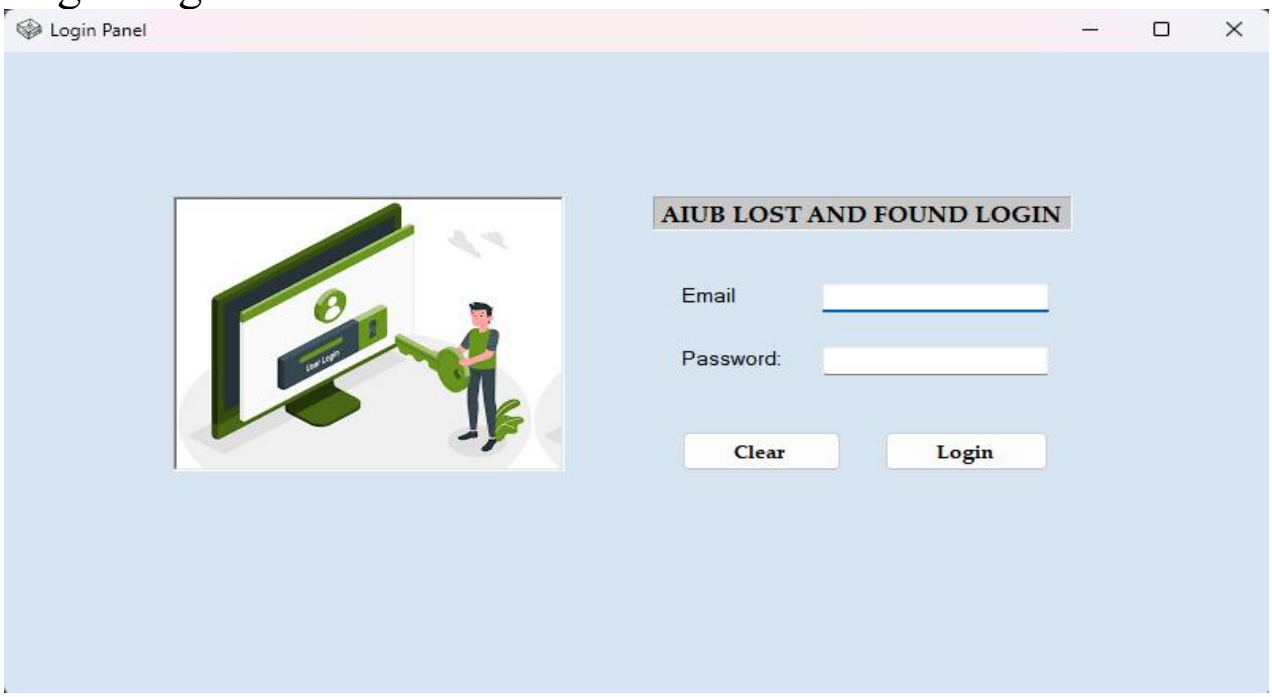
```
    string query = $"SELECT * FROM Users WHERE UserID = {selectedUserId}";
```

21. Removing User

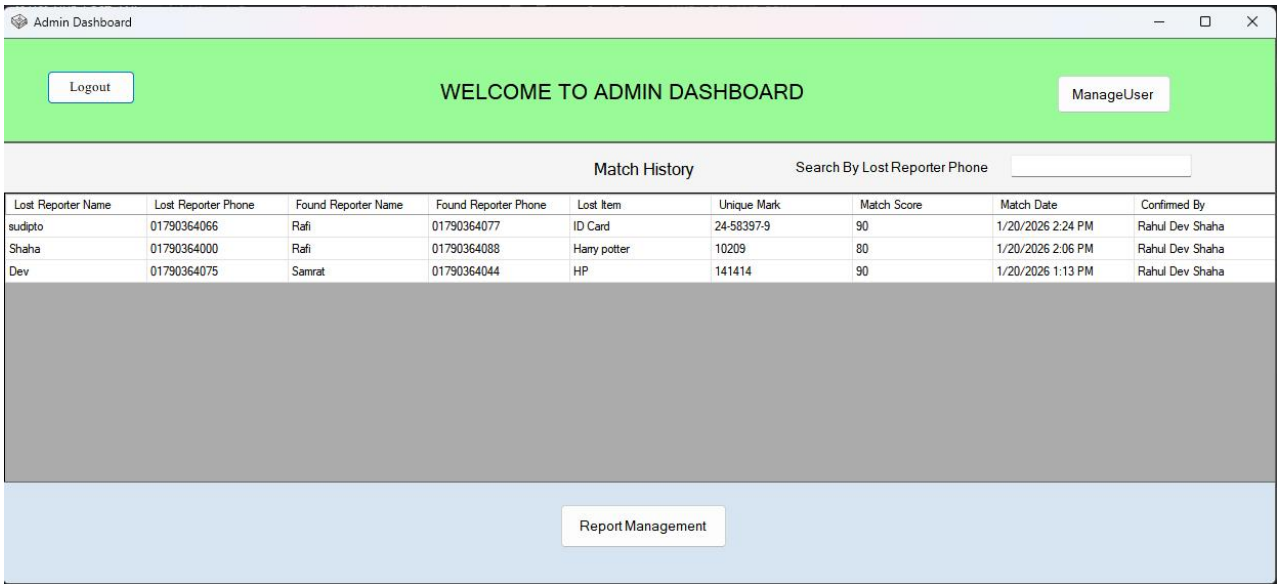
```
string query = $"DELETE FROM Users WHERE UserID = {selectedUserId}";
```

ScreenShots

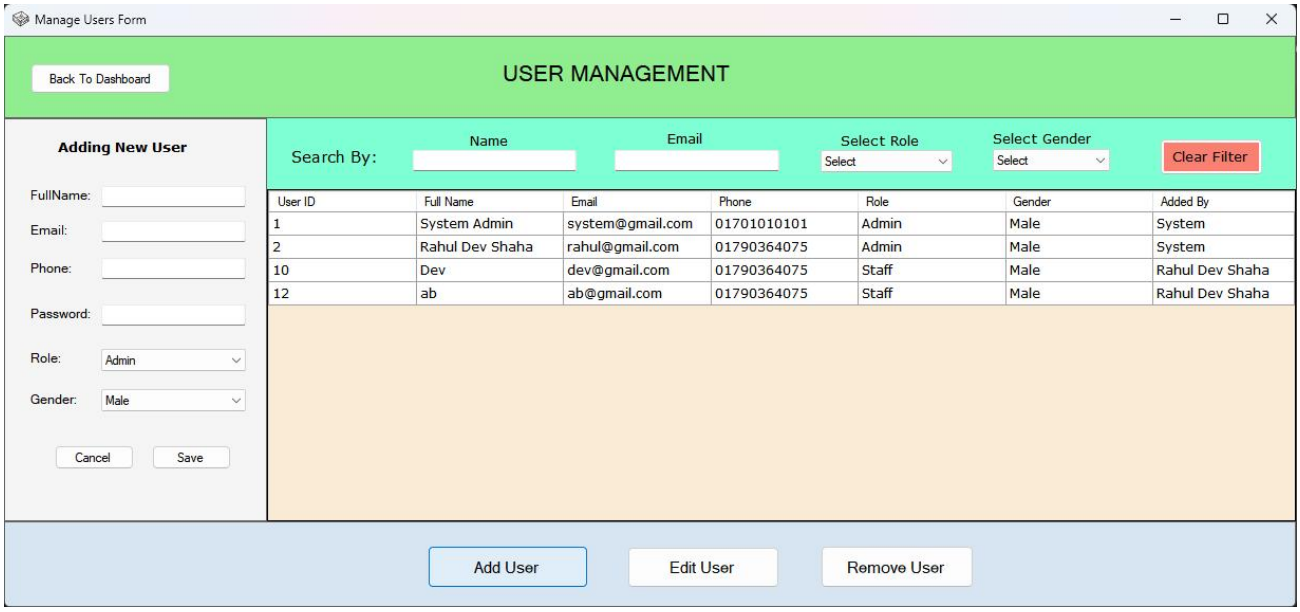
Login Page:



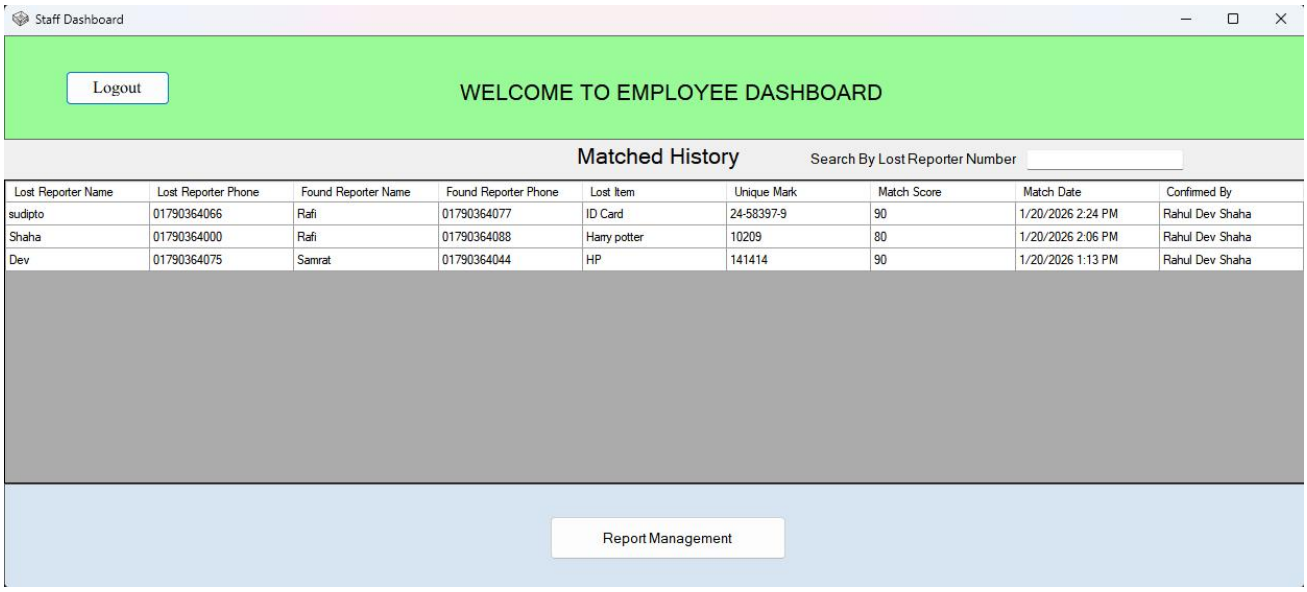
Admin Dashboard



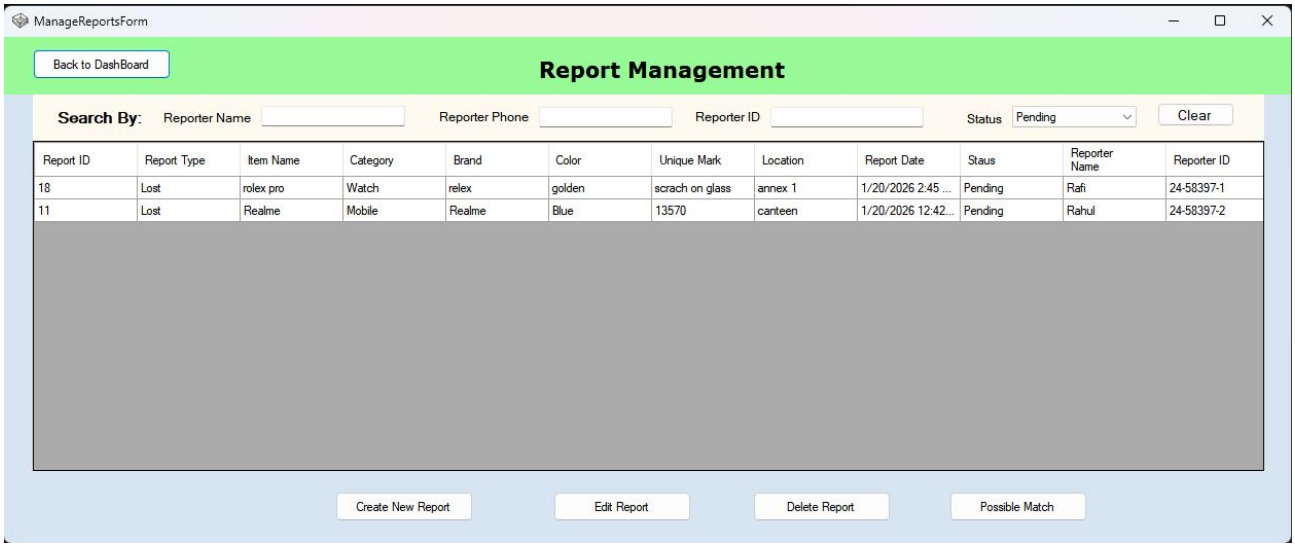
User Management



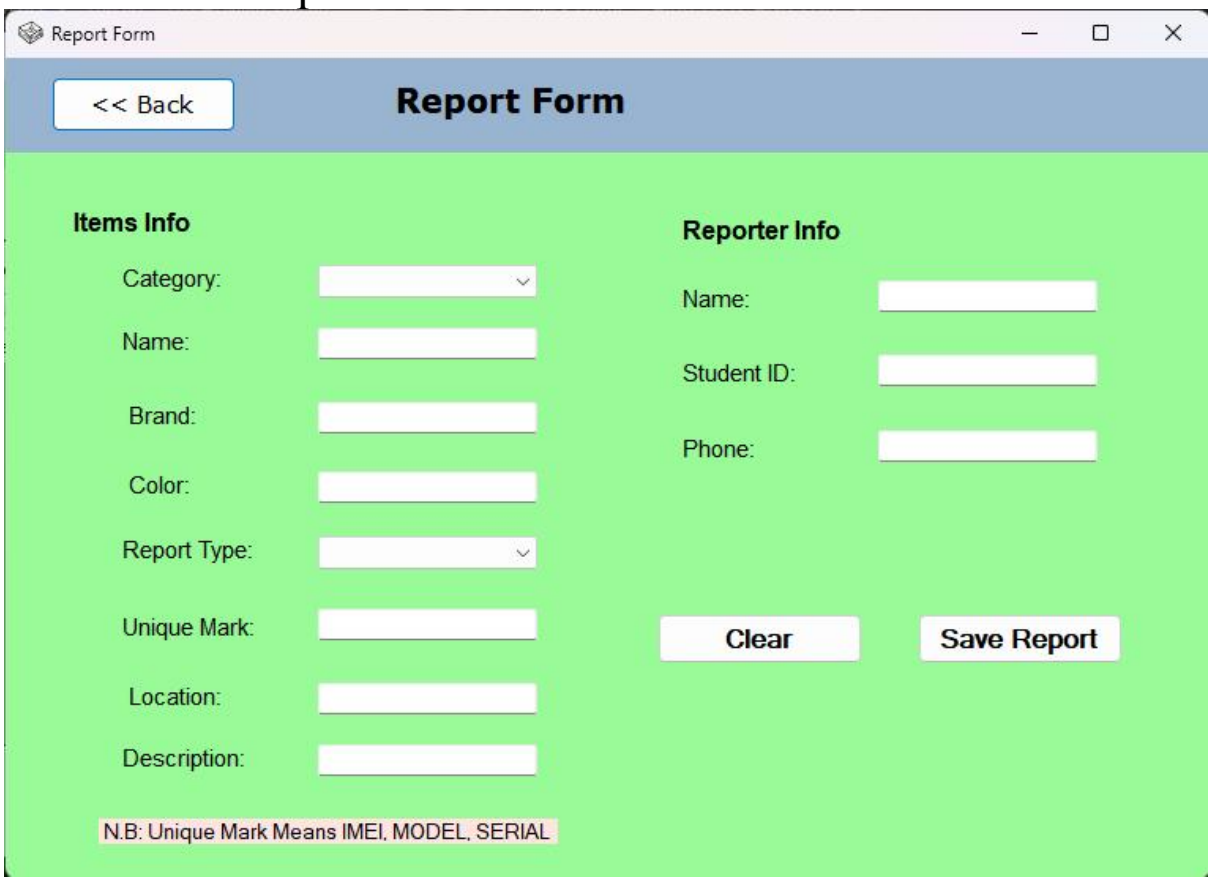
Staff/Employee Dashboard



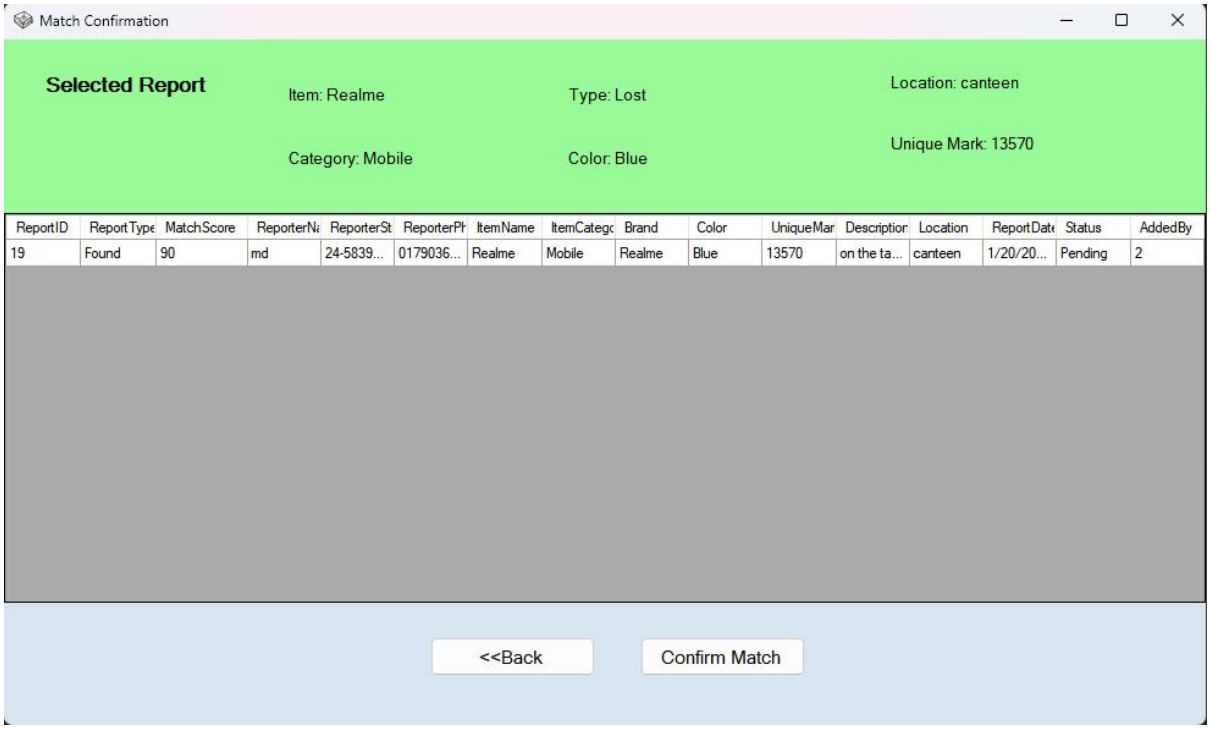
Report Management



Create New Report



Possible Match



Conclusion

The AIUB Lost & Found System is a efficient solution for managing lost and found items within the university. It simplifies the process of reporting, searching, and matching lost and found items through a secure and user-friendly platform. With features like role-based access, smart matching, advanced search, and status tracking, the system reduces complexity and improves accuracy. This project ensures better organization, faster recovery of lost items, and transparent record management. In the future, the system can be enhanced with mobile support, image-based search, and AI-powered matching for even better performance.