Room Management System

**Requirements Analysis Document**

Version 5

CSCI 4711 Software Engineering

Spring 2023

Augusta University

Team Members

James Wilt

Charles Smith

Ian Roberson

**Table of Contents**

[1 INTRODUCTION 3](#_Toc132278520)

[1.1 SCOPE OF SYSTEM 3](#_Toc132278521)

[2 REQUIREMENTS OF SYSTEM 4](#_Toc132278522)

[2.1 FUNCTIONAL REQUIREMENTS 4](#_Toc132278523)

[2.2 NON-FUNCTIONAL REQUIREMENTS 5](#_Toc132278524)

[2.3 USE CASES 6](#_Toc132278525)

[2.4 USE CASE DESCRIPTIONS 7](#_Toc132278526)

[2.5 REQUIREMENTS ANALYSIS 13](#_Toc132278527)

[3 USER INTERFACE MOCKUPS 20](#_Toc132278528)

[3.1 STARTUP/LOGIN 20](#_Toc132278529)

[3.2 EMPLOYEE DASHBOARD 21](#_Toc132278530)

[3.3 ADMIN DASHBOARD 21](#_Toc132278531)

[3.4 RESERVE ROOM 22](#_Toc132278532)

[3.5 CANCEL RESERVATION 22](#_Toc132278533)

[3.6 LOGOUT 23](#_Toc132278534)

[4 OBJECT DESIGN 24](#_Toc132278535)

[4.1 OBJECT RELATIONSHIP 24](#_Toc132278536)

[4.2 DETAILED CLASS DESIGN 26](#_Toc132278537)

[4.3 DATABASE DESIGN 27](#_Toc132278538)

[5 SYSTEM DESIGN 28](#_Toc132278539)

[5.1 SUBSYSTEM DECOMPOSITION 28](#_Toc132278540)

[6 APPENDIX 29](#_Toc132278541)

[6.1 APPENDIX A-SOURCE CODE 29](#_Toc132278542)

# INTRODUCTION

## SCOPE OF SYSTEM

The Room Management System (RMS) is a software tool for the reservation of rooms. It includes actors such as *Employee* who represent the users who reserve rooms, and *Admin,* the administrators who can executively cancel existing reservations.

The system supports both actors by logging and storing all the information pertaining to the reservations that are made via the system. This is stored via an internal database. The customers are able to interact with the system via the RMS Graphical User Interface.

The system includes functionality for verification and authorization of users, making room reservations, and canceling reservations. All changes to room statuses are logged in the internal database to ensure the most accurate status of the rooms are given to the actors utilizing the interface.

# REQUIREMENTS OF SYSTEM

## FUNCTIONAL REQUIREMENTS

* Startup: This functionality performs the initial configuration of the system including its database and user interface.
* Login – All users of the system must be verified as either an employee or an admin before they can use the Room Management System. This function verifies the user, grants them access to other functionality, and saves each successful long to the database.
* ReserveRoom – Employees can choose what room to reserve via this function. The employee will be able to choose which building, room, date, and time to reserve.
* CancelReservation – This function lets admins cancel reservations made by employees. Admins will be able to see the building, room, date, and time along with which employee reserved them.
* Logout – This function allows users of the system to logout of their accounts when they have finished using the RMS. Following this function will be the login screen so different users can begin using the RMS while saving each logout to the database.

## NON-FUNCTIONAL REQUIREMENTS

* + - * Platform:
        + Target operating system is Microsoft Windows.
        + System is not web-based and should not utilize a web browser.
        + System should be implemented using the Java or C# programming language.
      * Security
        + All user input is validated for appropriateness and prevention of SQL injection attacks.
        + User passwords should not be stored as plaintext in the database.
      * Usability
        + System should implement a Graphical User Interface.

## USE CASES

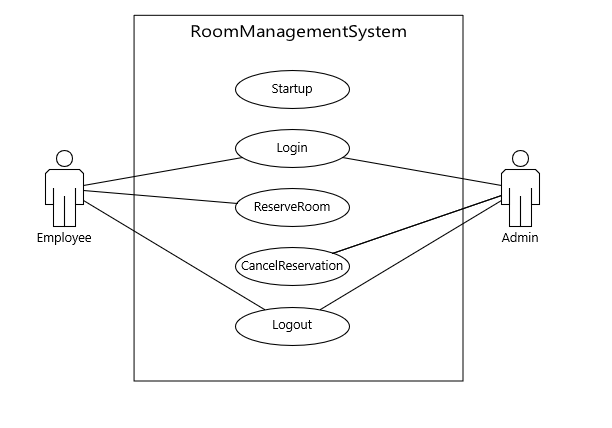


Figure 2.1: Use Case Diagarm

## USE CASE DESCRIPTIONS

|  |  |
| --- | --- |
| *Use case name* | Startup |
| *Participating*  *actors* |  |
| *Precondition* |  |
| *Flow of events* | 1. This use case is initiated when the application is launched. 2. **The RMS runs through its startup protocol, initializes the database and displays the GUI’s LoginMenu.** |
| *Post condition* | The LoginMenu is displayed. |
| *Security*  *requirements* | The Users Password is not stored as plaintext in the database. |

Figure 2.2: Startup

|  |  |
| --- | --- |
| *Use case name* | Login |
| *Participating*  *actors* | Initiated by the Employee and Admin |
| *Precondition* |  |
| *Flow of events* | 1. The user enters their username and password to login to the RMS. 2. **RMS receives the input and verifies the username and password are valid and authenticates the user against the database. Username and password are valid, and RMS displays the appropriate menu for the user. Login is saved to database.** |
| *Post condition* | The appropriate menu is displayed for the user |
| *Security*  *requirements* | Password should be obfuscated for the user.  User inputs should be verified against their specifications.  User inputs are validated to prevent SQL injection |

Figure 2.3: Login: Authorized User

|  |  |
| --- | --- |
| *Use case name* | Login |
| *Participating*  *actors* | Initiated by the Employee and Admin |
| *Precondition* |  |
| *Flow of events* | 1. The user enters their username and password to login to the RMS. 2. **RMS receives the input and verifies the username and password are valid and authenticates the user against the database. The username and password are invalid, and the GUI displays an error message.** |
| *Post condition* | The LoginMenu is displayed for the user |
| *Security*  *requirements* | Password should be obfuscated for the user.  User inputs should be verified against their specifications.  User inputs are validated to prevent SQL injection |

Figure 2.4: Login: Unauthorized User

|  |  |
| --- | --- |
| *Use case name* | ReserveRoom |
| *Participating*  *actors* | Initiated by Employee |
| *Precondition* | Employee is logged into the system. |
| *Flow of events* | 1. The Employee selects a building and room to reserve by pressing the “Reserve” button on the EmployeeDashboard. 2. **RMS presents the ReserveRoom window for the employee to choose the date and time for the reservation.** 3. The Employee selects the desired date and time to reserve, and then presses the “Make Reservation” button. 4. **RMS updates the database according to which building, room, date and time selected and presents the EmployeeDashboard.** |
| *Post condition* | The database is updated with room reservation information.  The EmployeeDashboard is displayed |
| *Security*  *requirements* |  |

Figure 2.5: ReserveRoom: Successful Reservation

|  |  |
| --- | --- |
| *Use case name* | CancelReservation |
| *Participating*  *actors* | Initiated by Admin |
| *Precondition* | Admin is logged into the system. |
| *Flow of events* | 1. Admin selects reservation to cancel using the “Cancel” button on the AdminDashboard 2. **RMS presents the CancelReservation menu with the details of what building, room, date and time were reserved by which employee.** 3. Admin confirms the room details and selects the “Cancel Reservation” button. 4. **RMS updates the database and displays the AdminDashboard**. |
| *Post condition* | The database is updated.  The AdminDashboard is displayed. |
| *Security*  *requirements* |  |

Figure 2.6: CancelReservation

|  |  |
| --- | --- |
| *Use case name* | Logout |
| *Participating*  *actors* | Initiated by Employee or Admin |
| *Precondition* | User is logged into the system. |
| *Flow of events* | 1. This function is activated when a user selects the “logout” button from the appropriate menu. 2. **RMS presents the LogoutMenu with a message and a “Close” button**. 3. The user selects the Close button. 4. **RMS returns the LoginMenu for another user to login. Logout is saved to the database.** |
| *Post condition* | The LoginMenu is displayed. |
| *Security*  *requirements* |  |

Figure 2.7: Logout

## REQUIREMENTS ANALYSIS

This page intentionally left blank

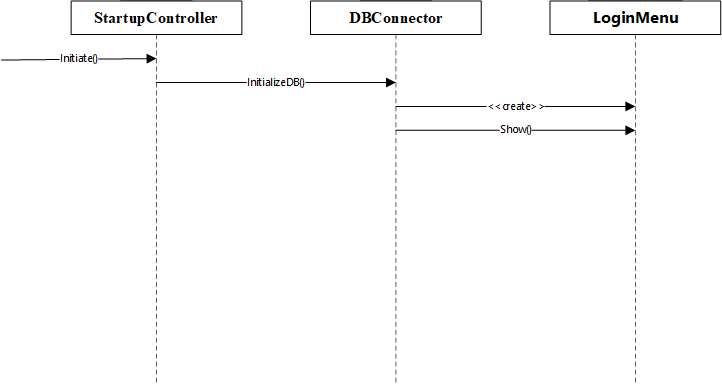


Figure 2.7: Startup

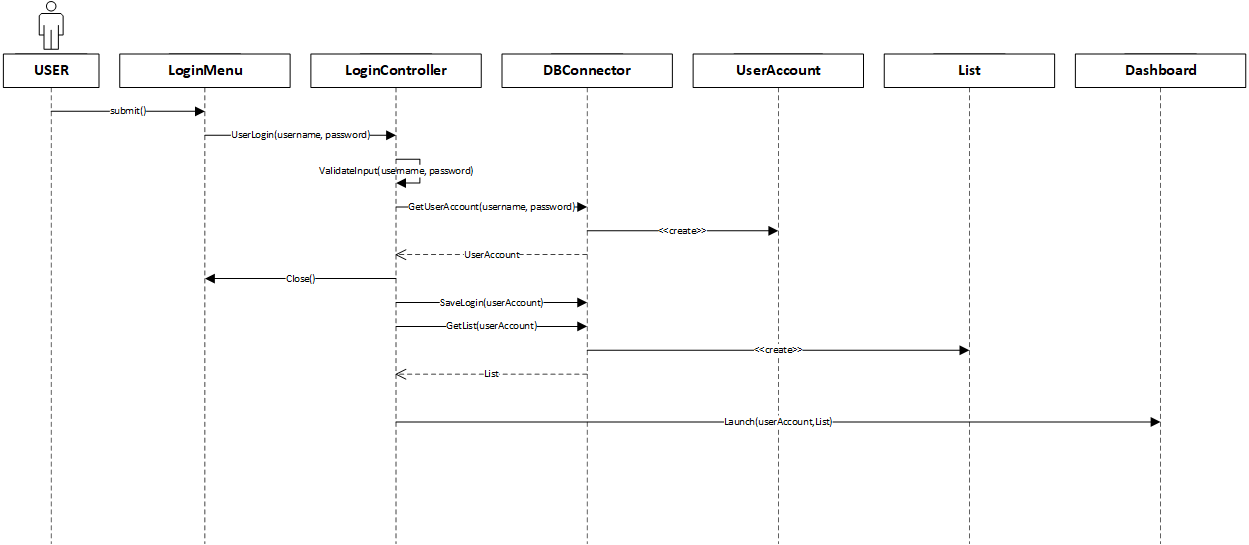


Figure 2.8: Login: Authorized User

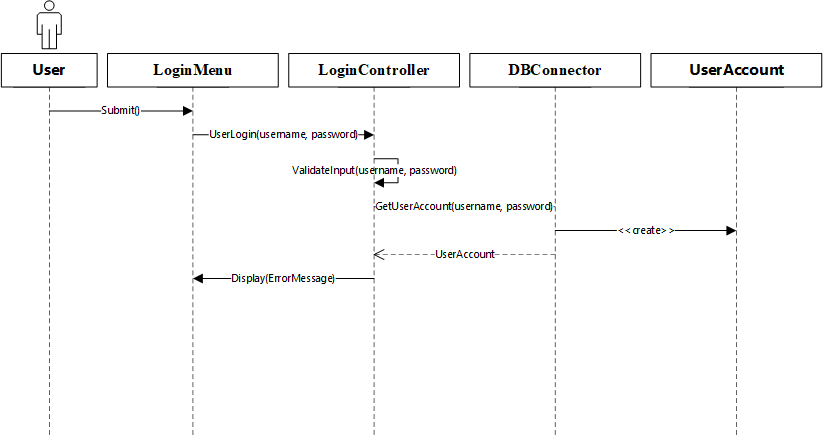


Figure 2.9: Login: Unauthorized User

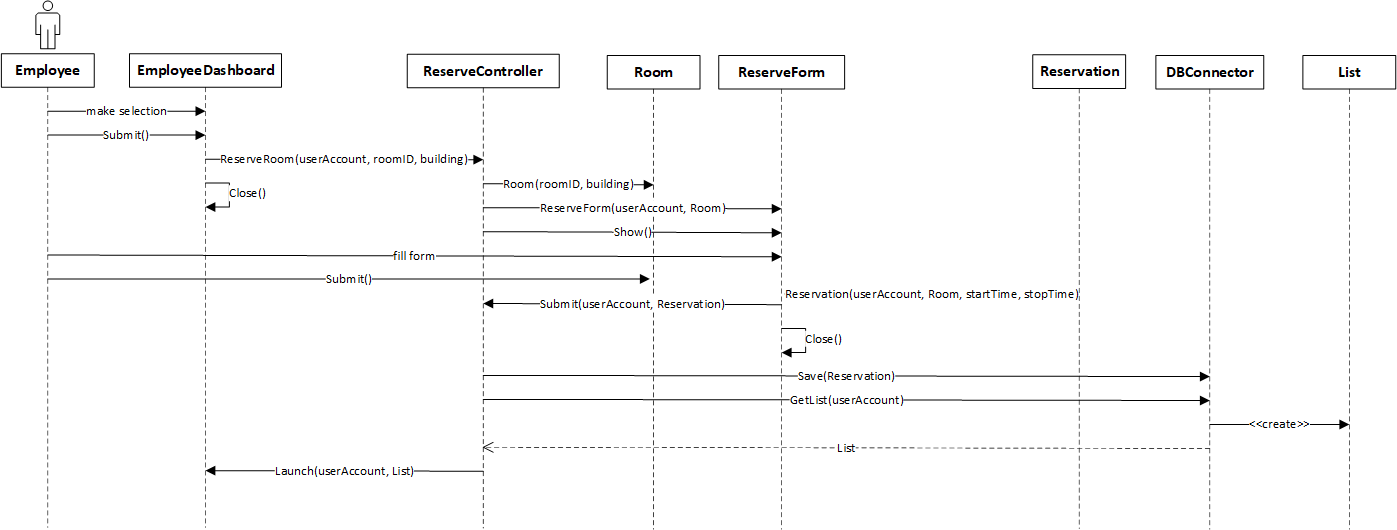


Figure 2.10 ReserveRoom

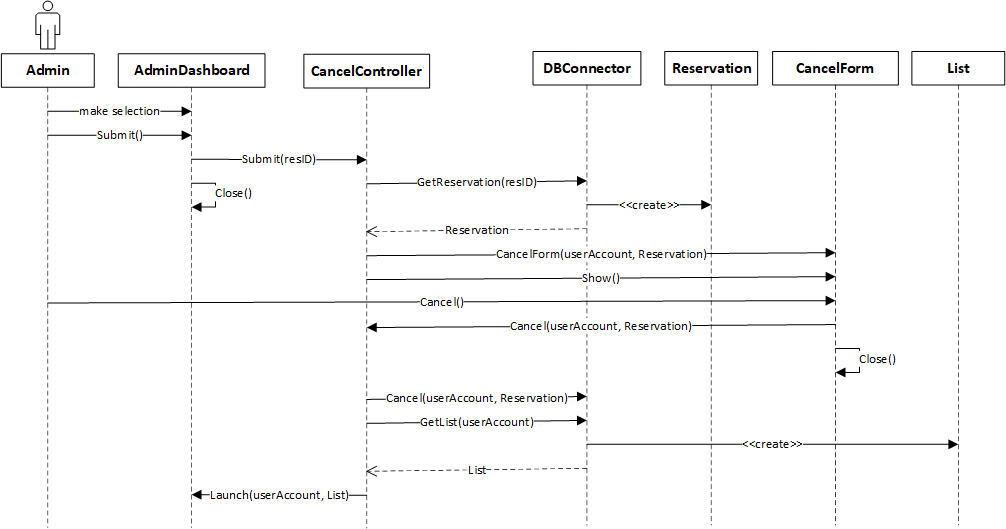


Figure 2.11 CancelReservation

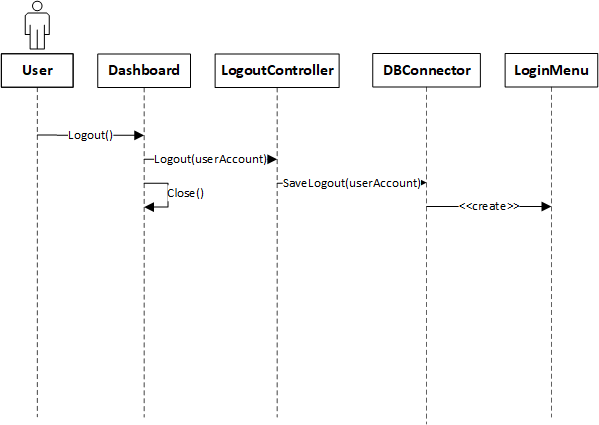


Figure 2.12 Logout

# USER INTERFACE MOCKUPS

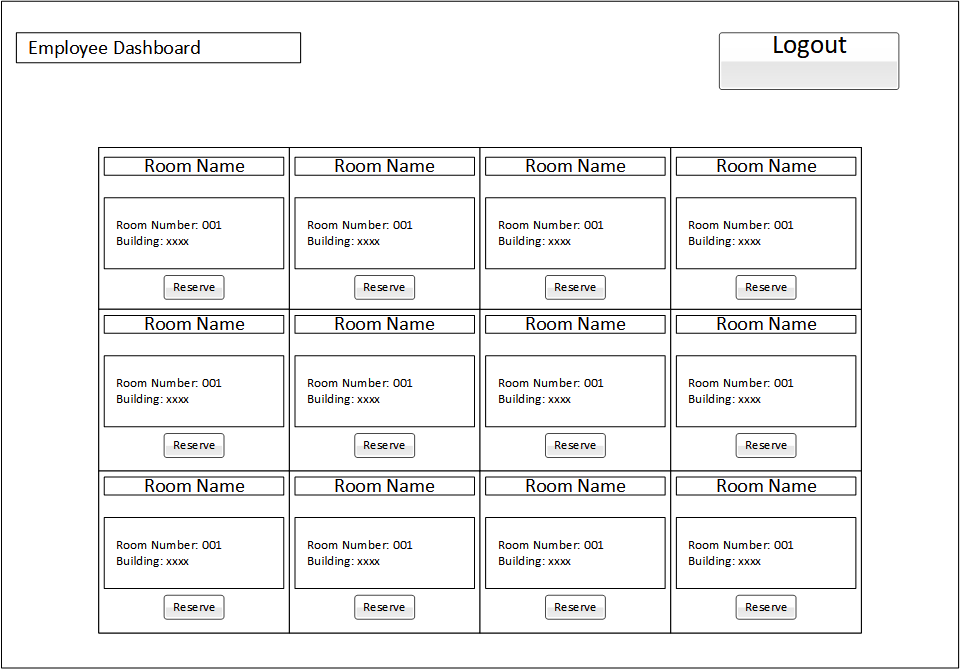
## STARTUP/LOGIN

Graphical user interface, text, application

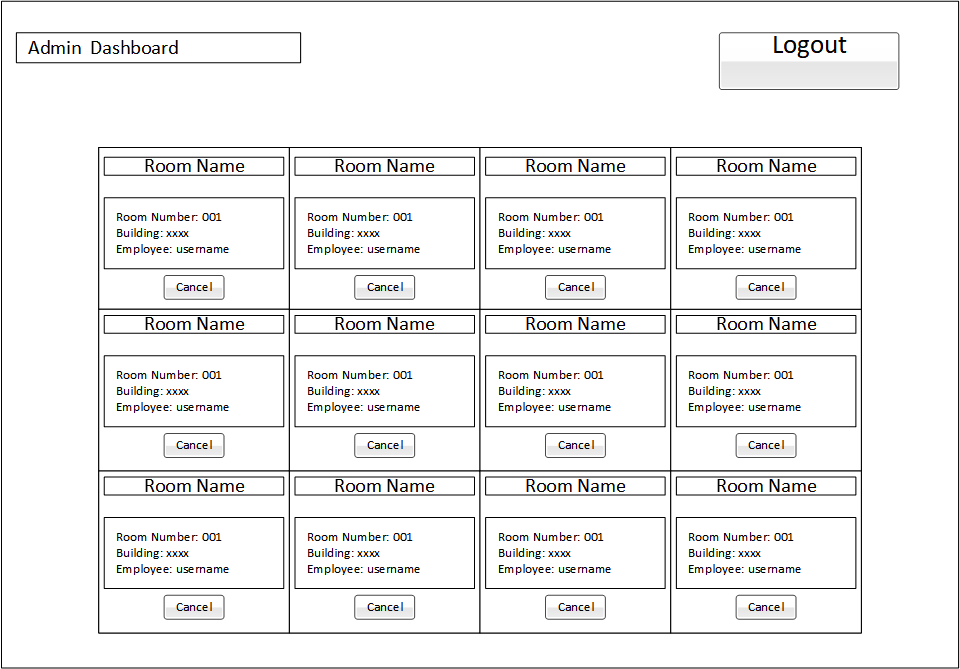
Description automatically generated

Figure 3.1: LoginMenu

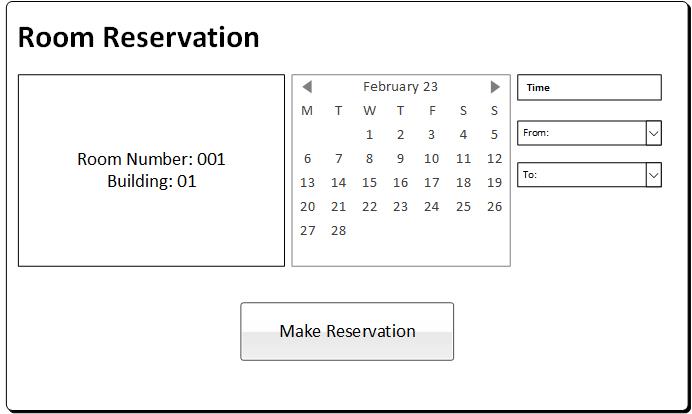
## EMPLOYEE DASHBOARD

  
Figure 3.2: EmployeeDashboard

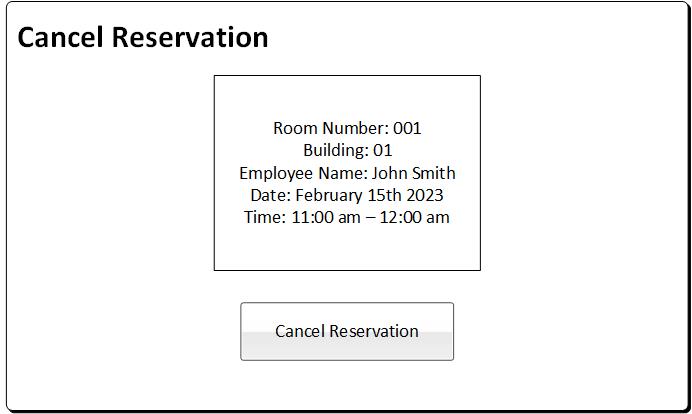
## ADMIN DASHBOARD

  
Figure 3.3: AdminDashboard

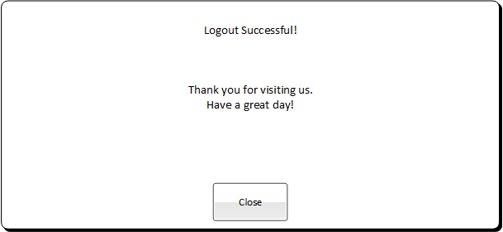
## RESERVE ROOM

  
Figure 3.4: ReserveRoom Menu

## CANCEL RESERVATION

  
Figure 3.5: CancelReservation Menu

## LOGOUT

  
Figure 3.6: Logout Menu

# OBJECT DESIGN

## OBJECT RELATIONSHIP

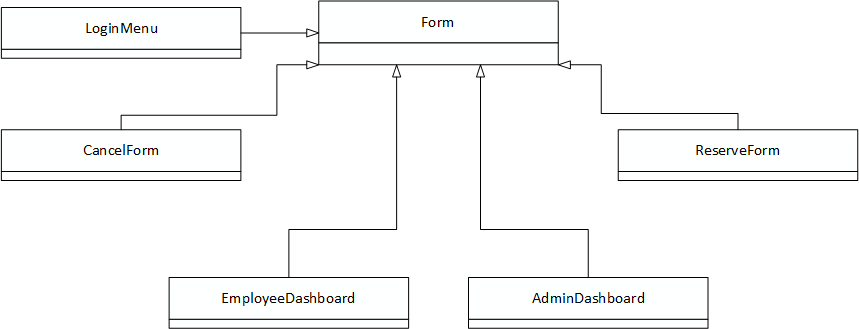
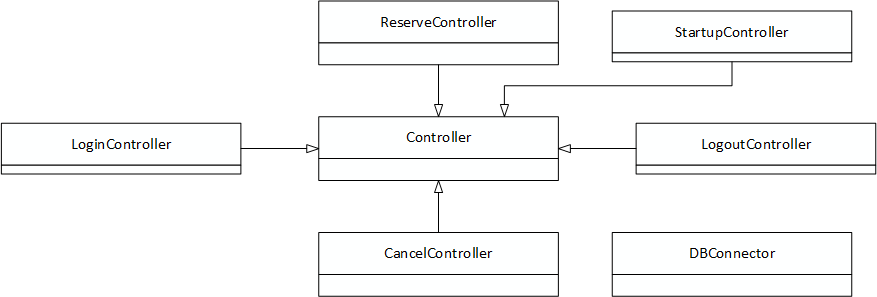


Figure 4.1: Class Diagram: Boundary

Figure 4.2: Class Diagram: Control

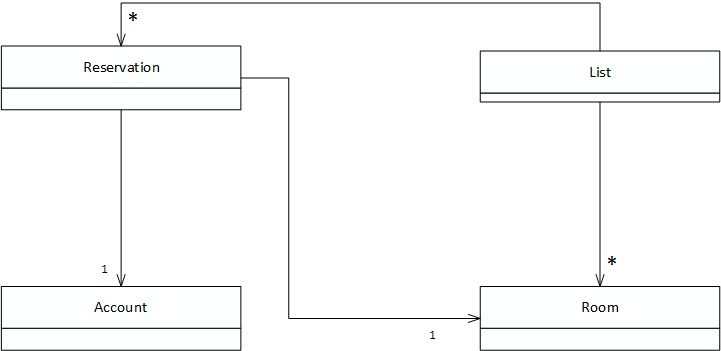


Figure 4.3: Class Diagram: Entity

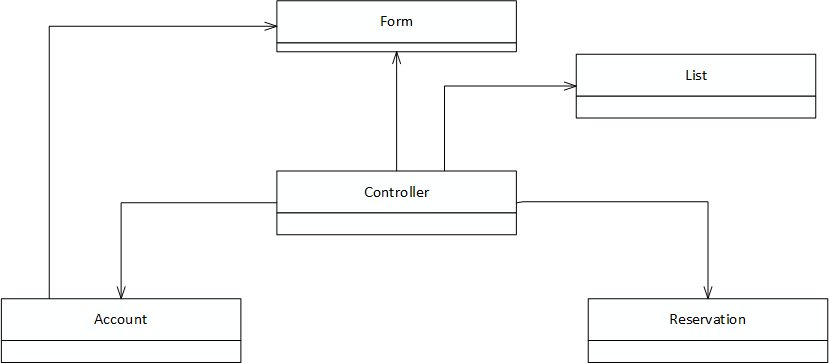


Figure 4.4: Class Diagram

## DETAILED CLASS DESIGN

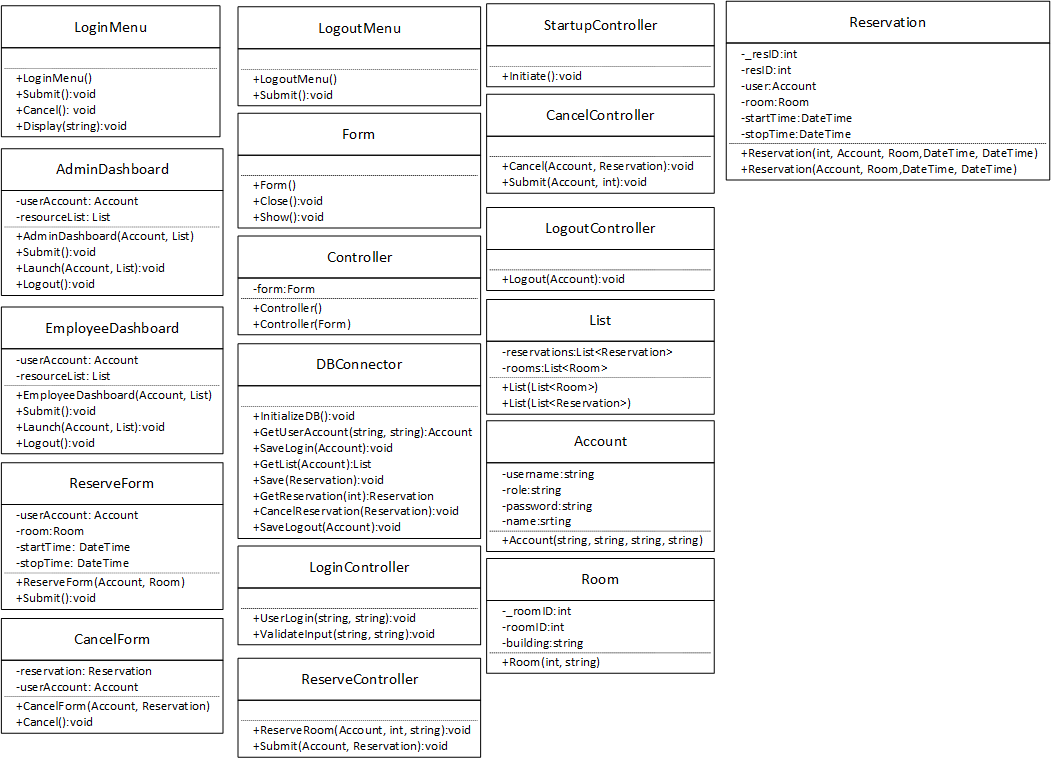


Figure 4.5: Detailed Diagram

## DATABASE DESIGN

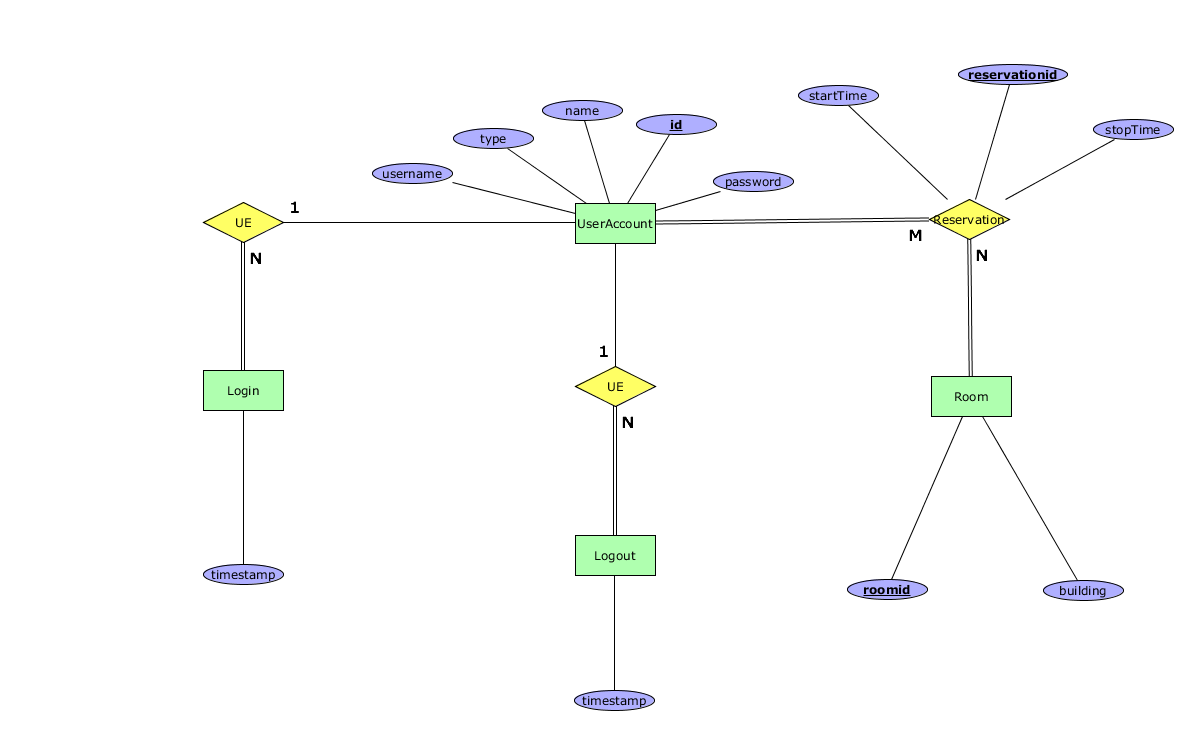


Figure 4.10 ER Diagram

**RELATIONAL MODEL**

RESERVATION (reservationid, startTime, stopTime, employeeidFK, roomidFK)

ROOM (roomid*,* building)

USERACCOUNT (id, username, password, type, name)

LOGIN (timestamp, accidFK)

LOGOUT (timestamp, accidFK)

# SYSTEM DESIGN

## SUBSYSTEM DECOMPOSITION

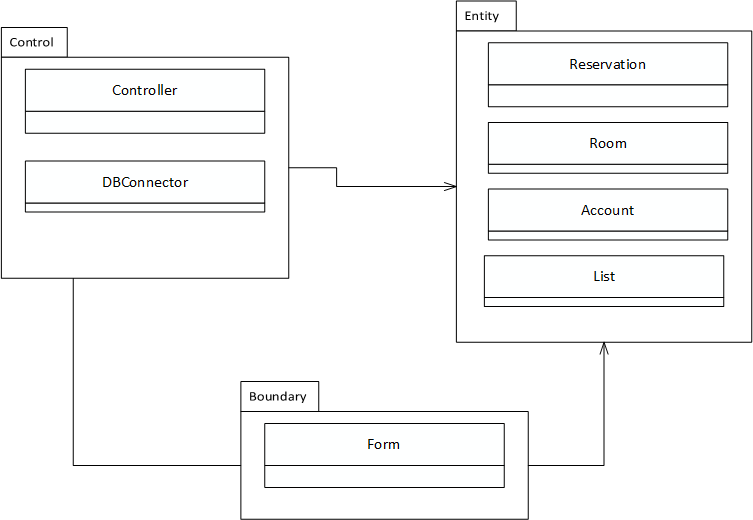


Figure 5.1 subsystems

# APPENDIX

## APPENDIX A-SOURCE CODE

namespace Entity

{

public class Account

{

public string username { get; set; }

public string password { get; set; }

public string role { get; set; }

public string name { get; set; }

public Account(string username, string role, string password, string name)

{

//Because this is all received from the db, validation is assumed to have happened.

this.username = username;

this.role = role;

this.password = password;

this.name = name;

}

}

}

namespace Entity

{

public class Reservation

{

private int \_resID;

public int resID

{

get

{

return this.\_resID;

}

set

{

if (value < int.MaxValue)

this.\_resID = value;

}

}

public Account user { get; set; }

public Room room { get; set; }

public DateTime startTime { get; set; }

public DateTime stopTime { get; set; }

//For grabbing old reservations from DB

public Reservation(int resID, Account userAccount, Room room, DateTime startTime, DateTime stopTime)

{

this.resID = resID;

this.user = userAccount;

this.room = room;

this.startTime = startTime;

this.stopTime = stopTime;

}

//For grabbing new reservations

public Reservation(Account userAccount, Room room, DateTime startTime, DateTime stopTime)

{

this.user = userAccount;

this.room = room;

this.startTime = startTime;

this.stopTime = stopTime;

}

}

}

namespace Controller

{

public static class DBController

{

public static void InitializeDB()

{

//DB Initialization. This should read data in from a flat file and create all the tables

if (!File.Exists(Path.Combine(Environment.CurrentDirectory, @"Data\", "RoomsDB.db")))

File.Create(Path.Combine(Environment.CurrentDirectory, @"Data\", "RoomsDB.db"));

//DB Initialization. This should read data in from a flat file and create all the tables

//File path is in the project folder bin

string root = "Data Source=" + Path.Combine(Environment.CurrentDirectory, @"Data\", "RoomsDB.db");

using (SQLiteConnection conn = new SQLiteConnection(root))

{

using (SQLiteCommand cmd = new SQLiteCommand())

{

conn.Open();

cmd.Connection = conn;

string strSql = @"BEGIN TRANSACTION;

DROP TABLE IF EXISTS ACCOUNT;

DROP TABLE IF EXISTS LOGIN;

DROP TABLE IF EXISTS LOGOUT;

DROP TABLE IF EXISTS ROOM;

DROP TABLE IF EXISTS RESERVATION;

COMMIT;";

cmd.CommandText = strSql;

cmd.ExecuteNonQuery();

string table = @"CREATE TABLE [ACCOUNT] (

[id] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL

, [username] TEXT NOT NULL

, [type] TEXT NOT NULL

, [password] TEXT NOT NULL

, [NAME] TEXT NOT NULL

);";

cmd.CommandText = table;

cmd.ExecuteNonQuery();

table = @"CREATE TABLE [LOGIN] (

[timestamp] DATETIME NOT NULL

, [accid] INTEGER NOT NULL

, FOREIGN KEY([accid]) REFERENCES [ACCOUNT]([id])

);";

cmd.CommandText = table;

cmd.ExecuteNonQuery();

table = @"CREATE TABLE [LOGOUT] (

[timestamp] DATETIME NOT NULL

, [accid] INTEGER NOT NULL

, FOREIGN KEY([accid]) REFERENCES [ACCOUNT]([id])

);";

cmd.CommandText = table;

cmd.ExecuteNonQuery();

table = @"CREATE TABLE [ROOM] (

[roomid] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL

, [building] TEXT NOT NULL

);";

cmd.CommandText = table;

cmd.ExecuteNonQuery();

table = @"CREATE TABLE [RESERVATION] (

[reservationid] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL

, [startTime] DATETIME NOT NULL

, [stopTime] DATETIME NOT NULL

, [employeeid] TEXT NOT NULL

, [roomid] TEXT NOT NULL

, FOREIGN KEY([employeeid]) REFERENCES [ACCOUNT]([id])

, FOREIGN KEY([roomid]) REFERENCES [ROOM]([roomid])

);";

cmd.CommandText = table;

cmd.ExecuteNonQuery();

strSql = @"BEGIN TRANSACTION;

INSERT INTO ACCOUNT (username, type, password, name) VALUES ('adminuser', 'admin', $hashpwd1, 'John');

INSERT INTO ACCOUNT (username, type, password, name) VALUES ('employeeuser', 'employee', $hashpwd2, 'Joe');

INSERT INTO ACCOUNT (username, type, password, name) VALUES ('jawilt-adm', 'admin', $hashpwd3, 'James');

INSERT INTO ACCOUNT (username, type, password, name) VALUES ('jawilt-emp', 'employee', $hashpwd4, 'James');

INSERT INTO ROOM (building) VALUES ('Mcknight');

INSERT INTO ROOM (building) VALUES ('Barret');

INSERT INTO ROOM (building) VALUES ('Gracie');

INSERT INTO ROOM (building) VALUES ('Lejuene');

INSERT INTO ROOM (building) VALUES ('Grant');

INSERT INTO ROOM (building) VALUES ('Hall');

INSERT INTO RESERVATION (startTime, stopTime, employeeid, roomid) VALUES ('2023-04-22 08:00:00','2023-04-22 09:00:00', 1, 1);

INSERT INTO RESERVATION (startTime, stopTime, employeeid, roomid) VALUES ('2023-04-22 09:00:00','2023-04-22 10:00:00', 2, 2);

INSERT INTO RESERVATION (startTime, stopTime, employeeid, roomid) VALUES ('2023-04-23 12:00:00','2023-04-23 13:00:00', 3, 4);

INSERT INTO RESERVATION (startTime, stopTime, employeeid, roomid) VALUES ('2023-04-23 13:00:00','2023-04-23 15:00:00', 3, 4);

COMMIT;";

cmd.CommandText = strSql;

cmd.Parameters.AddWithValue("$hashpwd1", "admin123".GetHashCode());

cmd.Parameters.AddWithValue("$hashpwd2", "employee123".GetHashCode());

cmd.Parameters.AddWithValue("$hashpwd3", "jawiltadm123".GetHashCode());

cmd.Parameters.AddWithValue("$hashpwd4", "jawilt123".GetHashCode());

cmd.ExecuteNonQuery();

conn.Close();

}

}

Debug.WriteLine("Initialized");

}

public static Account GetUserAccount(string username, string password)

{

Debug.WriteLine($"{password}");

string root = "Data Source=" + Path.Combine(Environment.CurrentDirectory, @"Data\", "RoomsDB.db");

using (SQLiteConnection conn = new SQLiteConnection(root))

{

conn.Open();

int x = username.GetHashCode();

int y = password.GetHashCode();

string stm = @"SELECT[id]

,[username]

,[password]

,[type]

,[name]

FROM[ACCOUNT]

WHERE[username] == ($name)

AND[password] == ($pd);";

using (SQLiteCommand cmd = new SQLiteCommand(stm, conn))

{

cmd.Parameters.AddWithValue("$name", username);

cmd.Parameters.AddWithValue("$pd", y);

using (SQLiteDataReader rdr = cmd.ExecuteReader())

{

while (rdr.Read())

{

//Account acct = new Account(rdr.GetInt32(0), user, rdr.GetString(3));

Debug.WriteLine($"{rdr.GetInt32(0).ToString()} {username} {rdr.GetString(1).ToString()} {rdr.GetString(2).ToString()} {rdr.GetString(3).ToString()}");

Account userAcc = new Account(username, rdr.GetString(3).ToString(), password, rdr.GetString(4).ToString());

return userAcc;

}

//Account act = new Account(0, null, null);

return null;

}

}

}

}

public static Entity.List GetList(Account userAccount)

{

if (userAccount.role == "admin")

{

List<Reservation> resourceList = new List<Reservation>();

string root = "Data Source=" + Path.Combine(Environment.CurrentDirectory, @"Data\", "RoomsDB.db");

using (SQLiteConnection conn = new SQLiteConnection(root))

{

using (SQLiteCommand cmd = new SQLiteCommand())

{

conn.Open();

cmd.Connection = conn;

cmd.CommandText = "SELECT \* FROM RESERVATION JOIN ACCOUNT ON RESERVATION.EMPLOYEEID = ACCOUNT.ID JOIN ROOM ON RESERVATION.ROOMID = ROOM.ROOMID;";

using (SQLiteDataReader rdr = cmd.ExecuteReader())

{

while (rdr.Read())

{

int id = rdr.GetInt32(0);

DateTime startTime = (DateTime)rdr["startTime"];

DateTime stopTime = (DateTime)rdr["stopTime"];

int roomid = int.Parse(rdr["roomid"].ToString());

string name = (string)rdr["name"];

string username = (string)rdr["username"];

string password = (string)rdr["password"];

string role = (string)rdr["type"];

string building = (string)rdr["building"];

Debug.WriteLine(DateTime.Parse($"{startTime}").ToString());

resourceList.Add(new Reservation(id, new Account(username, role, password, name), new Room(roomid, building), startTime, stopTime));

}

}

}

}

return new Entity.List(resourceList);

}

else

{

List<Room> resourceList = new List<Room>();

string root = "Data Source=" + Path.Combine(Environment.CurrentDirectory, @"Data\", "RoomsDB.db");

using (SQLiteConnection conn = new SQLiteConnection(root))

{

using (SQLiteCommand cmd = new SQLiteCommand())

{

conn.Open();

cmd.Connection = conn;

cmd.CommandText = "SELECT \* FROM ROOM;";

using (SQLiteDataReader rdr = cmd.ExecuteReader())

{

while (rdr.Read())

{

int roomid = int.Parse(rdr["roomid"].ToString());

string building = (string)rdr["building"];

resourceList.Add(new Room(roomid, building));

}

}

}

}

return new Entity.List(resourceList);

}

}

public static void SaveLogin(Account userAccount)

{

//This method saves the login user and time to the database

string root = "Data Source=" + Path.Combine(Environment.CurrentDirectory, @"Data\", "RoomsDB.db");

using (SQLiteConnection conn = new SQLiteConnection(root))

{

conn.Open();

int id = 0;

// int hash = userAccount.username.GetHashCode();

string stm = "SELECT [id] FROM ACCOUNT WHERE username = ($name);";

using (SQLiteCommand cmnd = new SQLiteCommand(stm, conn))

{

cmnd.Parameters.AddWithValue("$name", userAccount.username);

using (SQLiteDataReader rdr = cmnd.ExecuteReader())

{

while (rdr.Read())

{

id = rdr.GetInt32(0);

}

}

}

stm = @"INSERT INTO LOGIN VALUES($time,$id);";

using (SQLiteCommand cmnd = new SQLiteCommand())

{

cmnd.Connection = conn;

cmnd.CommandText = stm;

cmnd.Parameters.AddWithValue("$id", id);

cmnd.Parameters.AddWithValue("$time", DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"));

cmnd.ExecuteNonQuery();

Debug.WriteLine($"Saved Login for {userAccount.username} at {DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss")}");

}

}

}

public static void SaveLogout(Account userAccount)

{

//This method saves the login user and time to the database

string root = "Data Source=" + Path.Combine(Environment.CurrentDirectory, @"Data\", "RoomsDB.db");

using (SQLiteConnection conn = new SQLiteConnection(root))

{

conn.Open();

int id = 0;

// int hash = userAccount.username.GetHashCode();

string stm = "SELECT [id] FROM ACCOUNT WHERE username = ($name);";

using (SQLiteCommand cmnd = new SQLiteCommand(stm, conn))

{

cmnd.Parameters.AddWithValue("$name", userAccount.username);

using (SQLiteDataReader rdr = cmnd.ExecuteReader())

{

while (rdr.Read())

{

id = rdr.GetInt32(0);

}

}

}

stm = @"INSERT INTO LOGOUT VALUES($time, $id);";

using (SQLiteCommand cmnd = new SQLiteCommand())

{

cmnd.Connection = conn;

cmnd.CommandText = stm;

cmnd.Parameters.AddWithValue("$id", id);

cmnd.Parameters.AddWithValue("$time", DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"));

cmnd.ExecuteNonQuery();

Debug.WriteLine("Saved Logout");

}

}

}

public static Reservation GetReservation(int resID)

{

//This method Deletes a Reservation from the database via the resID primary key

Reservation res = null;

string root = "Data Source=" + Path.Combine(Environment.CurrentDirectory, @"Data\", "RoomsDB.db");

using (SQLiteConnection conn = new SQLiteConnection(root))

{

conn.Open();

string stm = "SELECT \* FROM RESERVATION JOIN ACCOUNT ON RESERVATION.EMPLOYEEID = ACCOUNT.ID JOIN ROOM ON RESERVATION.ROOMID=ROOM.ROOMID WHERE reservationid = ($resID);";

using (SQLiteCommand cmd = new SQLiteCommand(stm, conn))

{

cmd.Parameters.AddWithValue("$resID", resID);

using (SQLiteDataReader rdr = cmd.ExecuteReader())

{

while (rdr.Read())

{

int id = rdr.GetInt32(0);

DateTime startTime = (DateTime)rdr["startTime"];

DateTime stopTime = (DateTime)rdr["stopTime"];

int empid = int.Parse(rdr["employeeid"].ToString());

int roomid = int.Parse(rdr["roomid"].ToString());

string name = (string)rdr["name"];

string username = (string)rdr["username"];

string password = (string)rdr["password"];

string role = (string)rdr["type"];

string building = (string)rdr["building"];

Debug.WriteLine($"Reservation: {id}");

res = new Reservation(id, new Account(username, role, password, name), new Room(roomid, building), startTime, stopTime);

}

}

}

}

return res;

}

public static void Save(Reservation reservation)

{

int employeeid = 0;

//This method saves a new reservation to the Database

string root = "Data Source=" + Path.Combine(Environment.CurrentDirectory, @"Data\", "RoomsDB.db");

using (SQLiteConnection conn = new SQLiteConnection(root))

{

conn.Open();

string stm = "SELECT [id] FROM ACCOUNT WHERE username = ($name);";

using (SQLiteCommand cmnd = new SQLiteCommand(stm, conn))

{

cmnd.Parameters.AddWithValue("$name", reservation.user.username);

using (SQLiteDataReader rdr = cmnd.ExecuteReader())

{

while (rdr.Read())

{

employeeid = rdr.GetInt32(0);

}

}

}

stm = @"INSERT INTO RESERVATION (startTime, stopTime, employeeid, roomid) VALUES($startTime, $stopTime, $employeeid, $roomid);";

using (SQLiteCommand cmnd = new SQLiteCommand())

{

cmnd.Connection = conn;

cmnd.CommandText = stm;

cmnd.Parameters.AddWithValue("$startTime", $"{reservation.startTime.ToString("yyyy-MM-dd HH:mm:ss")}");

cmnd.Parameters.AddWithValue("$stopTime", $"{reservation.stopTime.ToString("yyyy-MM-dd HH:mm:ss")}");

cmnd.Parameters.AddWithValue("$employeeid", employeeid);

cmnd.Parameters.AddWithValue("$roomid", reservation.room.roomID);

cmnd.ExecuteNonQuery();

}

}

//Debug.WriteLine("Saved Logout");

Debug.WriteLine("Saved Reservation");

}

public static void CancelReservation(Reservation reservation)

{

//This method Deletes a Reservation from the database via the resID primary key

string root = "Data Source=" + Path.Combine(Environment.CurrentDirectory, @"Data\", "RoomsDB.db");

using (SQLiteConnection conn = new SQLiteConnection(root))

{

conn.Open();

string stm = "DELETE FROM RESERVATION WHERE reservationid = ($resID);";

using (SQLiteCommand cmnd = new SQLiteCommand())

{

cmnd.Connection = conn;

cmnd.CommandText = stm;

cmnd.Parameters.AddWithValue("$resID", reservation.resID);

cmnd.ExecuteNonQuery();

}

}

Debug.WriteLine($"Cancelled Reservation: {reservation.resID}");

}

}

}namespace Controller

{

public class LoginController : Controller

{

public LoginController(Form form) : base(form)

{

this.form = form;

//This is the Login Controller Constructor

}

public bool ValidateInput(string username, string password)

//Define what our username and password specification should be

{

if (username == null || password == null)

{

return false;

}

//If password or user is under 8 or over 12, not valid

if (password.Length < 8 || password.Length > 12 || username.Length < 8 || username.Length > 12)

return false;

//If pass or user contains no-no character, not valid

if (username.Contains(" ") || username.Contains("/") || username.Contains("\\") || username.Contains(".") || username.Contains(";") || username.Contains(":"))

return false;

if (password.Contains(" ") || password.Contains("/") || password.Contains("\\") || password.Contains(".") || password.Contains(";") || password.Contains(":"))

return false;

//Password validation

int digits = 0;

int capitals = 0;

foreach (char chr in password)

{

if (char.IsNumber(chr))

digits++;

if (char.IsUpper(chr))

capitals++;

}

if (digits == 0 && capitals == 0)

return false;

return true;

}

public void UserLogin(string username, string password)

{

//This method validates the input of the username and password then calls the GetUserAccount

if (this.ValidateInput(username, password) == true)

{

//After validation, we create a dbconnector and get the user account

Account userAccount = DBController.GetUserAccount(username, password);

if (userAccount == null)

{

LoginMenu.Display("Incorrect Username or Password; Please Try Again.");

}

else

{

DBController.SaveLogin(userAccount);

Entity.List resourceList = DBController.GetList(userAccount);

if (userAccount.role == "admin")

{

AdminDashboard.Launch(userAccount, resourceList);

this.form.Close();

}

else if (userAccount.role == "employee")

{

EmployeeDashboard.Launch(userAccount, resourceList);

this.form.Close();

}

}

}

else

{

LoginMenu.Display("Please enter a valid Username or Password.");

}

}

}

}

namespace Boundary

{

public partial class LoginMenu : Form //Implemented Code

{

public LoginMenu()

{

InitializeComponent();

}

private void Cancel(object sender, EventArgs e)

{

this.Close();

}

private void Submit(object sender, EventArgs e)

{

string username = this.username.Text;

string password = this.password.Text;

LoginController controller = new LoginController(this);

controller.UserLogin(username, password);

}

public static void Display(string ErrorMessage)

{

MessageBox.Show(ErrorMessage);

}

}  
  
 partial class LoginMenu //UI Generated Code

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.cancel = new System.Windows.Forms.Button();

this.enter = new System.Windows.Forms.Button();

this.username = new System.Windows.Forms.TextBox();

this.label1 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

this.password = new System.Windows.Forms.TextBox();

this.label3 = new System.Windows.Forms.Label();

this.errorLabel = new System.Windows.Forms.Label();

this.SuspendLayout();

//

// cancel

//

this.cancel.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);

this.cancel.Location = new System.Drawing.Point(178, 384);

this.cancel.Margin = new System.Windows.Forms.Padding(4, 3, 4, 3);

this.cancel.Name = "cancel";

this.cancel.Size = new System.Drawing.Size(150, 43);

this.cancel.TabIndex = 0;

this.cancel.Text = "Cancel";

this.cancel.UseVisualStyleBackColor = true;

this.cancel.Click += new System.EventHandler(this.Cancel);

//

// enter

//

this.enter.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);

this.enter.Location = new System.Drawing.Point(541, 384);

this.enter.Margin = new System.Windows.Forms.Padding(4, 3, 4, 3);

this.enter.Name = "enter";

this.enter.Size = new System.Drawing.Size(150, 43);

this.enter.TabIndex = 1;

this.enter.Text = "Enter";

this.enter.UseVisualStyleBackColor = true;

this.enter.Click += new System.EventHandler(this.Submit);

//

// username

//

this.username.Location = new System.Drawing.Point(345, 157);

this.username.Margin = new System.Windows.Forms.Padding(4, 3, 4, 3);

this.username.Name = "username";

this.username.Size = new System.Drawing.Size(346, 23);

this.username.TabIndex = 3;

//

// label1

//

this.label1.AutoSize = true;

this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);

this.label1.Location = new System.Drawing.Point(174, 157);

this.label1.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(91, 20);

this.label1.TabIndex = 4;

this.label1.Text = "Username";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);

this.label2.Location = new System.Drawing.Point(174, 267);

this.label2.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(86, 20);

this.label2.TabIndex = 5;

this.label2.Text = "Password";

//

// password

//

this.password.Location = new System.Drawing.Point(345, 269);

this.password.Margin = new System.Windows.Forms.Padding(4, 3, 4, 3);

this.password.Name = "password";

this.password.Size = new System.Drawing.Size(346, 23);

this.password.TabIndex = 6;

this.password.UseSystemPasswordChar = true;

//

// label3

//

this.label3.AutoSize = true;

this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);

this.label3.Location = new System.Drawing.Point(174, 63);

this.label3.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(354, 20);

this.label3.TabIndex = 7;

this.label3.Text = "Please enter your username and password.";

//

// errorLabel

//

this.errorLabel.AutoSize = true;

this.errorLabel.ForeColor = System.Drawing.Color.Red;

this.errorLabel.Location = new System.Drawing.Point(314, 342);

this.errorLabel.Name = "errorLabel";

this.errorLabel.Size = new System.Drawing.Size(0, 15);

this.errorLabel.TabIndex = 8;

this.errorLabel.Visible = false;

//

// LoginMenu

//

this.AutoScaleDimensions = new System.Drawing.SizeF(7F, 15F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(939, 583);

this.Controls.Add(this.errorLabel);

this.Controls.Add(this.label3);

this.Controls.Add(this.password);

this.Controls.Add(this.label2);

this.Controls.Add(this.label1);

this.Controls.Add(this.username);

this.Controls.Add(this.enter);

this.Controls.Add(this.cancel);

this.Margin = new System.Windows.Forms.Padding(4, 3, 4, 3);

this.Name = "LoginMenu";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Login";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button cancel;

private System.Windows.Forms.Button enter;

private System.Windows.Forms.TextBox username;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.TextBox password;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.Label errorLabel;

}

}

namespace Boundary

{

public partial class AdminDashboard : Form //User created code

{

private Account userAccount;

Entity.List resourceList;

public AdminDashboard(Account userAccount, Entity.List resourceList)

{

this.userAccount = userAccount;

this.resourceList = resourceList;

InitializeComponent();

//

// All the reservations

//

//Set the starting locations of each of the card assets

int panelPosX = 40;

int panelPosY = 120;

int counter = 0;

foreach (Reservation reserve in resourceList.reservations)

{

Panel card = new Panel();

Panel innercard = new Panel();

Label rlabel = new Label();

GroupBox rgroupBox = new GroupBox();

Button rbutton = new Button();

Label room = new Label();

Label room2 = new Label();

Label building = new Label();

Label building2 = new Label();

Label employee = new Label();

Label employee2 = new Label();

innercard.SuspendLayout();

card.SuspendLayout();

rgroupBox.SuspendLayout();

this.SuspendLayout();

//Configure the Card

card.Controls.Add(innercard);

card.Name = $"Card-{reserve.room.roomID}-Background";

card.Size = new Size(210, 238);

card.Location = new Point(panelPosX, panelPosY);

card.BackColor = Color.Black;

card.TabIndex = 0;

if (panelPosX < 930)

panelPosX = panelPosX + 228; //Tile Card along the row

else

{

panelPosX = 40;//Reset to next row

panelPosY = panelPosY + 260;

}

//Configure innercard

innercard.Controls.Add(rlabel);

innercard.Controls.Add(rgroupBox);

innercard.Controls.Add(rbutton);

innercard.Size = new Size(208, 236);

innercard.Location = new Point(1, 1);

innercard.TabIndex = 0;

innercard.BackColor = Color.White;

//Configure the Label

rlabel.AutoSize = true;

rlabel.Size = new Size(78, 32);

rlabel.Font = new Font("Segoe UI", 16F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point);

rlabel.Name = $"Label-{reserve.room.roomID}-{counter}";

rlabel.Text = $"Room #{reserve.room.roomID}";

rlabel.TabIndex = 0;

rlabel.Location = new Point(56, 9); //Location is relative to the card

//Configure the groupbox

rgroupBox.Location = new System.Drawing.Point(17, 44);

rgroupBox.Name = $"GroupBox-{reserve.room.roomID}-{counter}";

rgroupBox.Size = new System.Drawing.Size(176, 126);

rgroupBox.TabIndex = 1;

rgroupBox.TabStop = false;

rgroupBox.Controls.Add(room);

rgroupBox.Controls.Add(room2);

rgroupBox.Controls.Add(building);

rgroupBox.Controls.Add(building2);

rgroupBox.Controls.Add(employee);

rgroupBox.Controls.Add(employee2);

//Configure inner groupBox

room.AutoSize = true;

room.Font = new Font("Segoe UI", 12F, (FontStyle.Bold | FontStyle.Underline), GraphicsUnit.Point);

room.Location = new Point(10, 33);

room.Name = "Room Number";

room.Size = new Size(127, 21);

room.TabIndex = 0;

room.Text = $"Room Number:";

room2.AutoSize = true;

room2.Font = new Font("Segoe UI", 12F, GraphicsUnit.Point);

room2.Location = new Point(135, 33);

room2.Name = $"{reserve.room.roomID}-{counter}";

room2.Size = new Size(127, 21);

room2.TabIndex = 0;

room2.Text = $"{reserve.room.roomID}";

building.AutoSize = true;

building.Font = new Font("Segoe UI", 12F, (FontStyle.Bold | FontStyle.Underline), GraphicsUnit.Point);

building.Location = new Point(10, 53);

building.Name = "Building";

building.Size = new Size(127, 21);

building.TabIndex = 0;

building.Text = $"Building:";

building2.AutoSize = true;

building2.Font = new Font("Segoe UI", 12F, GraphicsUnit.Point);

building2.Location = new Point(85, 53);

building2.Name = $"{reserve.room.building}-{counter}";

building2.Size = new Size(127, 21);

building2.TabIndex = 0;

building2.Text = $"{reserve.room.building}";

employee.AutoSize = true;

employee.Font = new Font("Segoe UI", 12F, (FontStyle.Bold | FontStyle.Underline), GraphicsUnit.Point);

employee.Location = new Point(10, 73);

employee.Name = "Employee";

employee.Size = new Size(127, 21);

employee.TabIndex = 0;

employee.Text = $"Employee:";

employee2.AutoSize = true;

employee2.Font = new Font("Segoe UI", 12F, GraphicsUnit.Point);

employee2.Location = new Point(100, 73);

employee2.Name = $"{reserve.user.name}-{counter}";

employee2.Size = new Size(127, 21);

employee2.TabIndex = 0;

employee2.Text = $"{reserve.user.name}";

//Configure the Button

rbutton.Location = new Point(66, 176);

rbutton.Name = $"Button-{reserve.resID}";

rbutton.Size = new Size(78, 46);

rbutton.TabIndex = 2;

rbutton.Text = "Cancel";

rbutton.UseVisualStyleBackColor = true;

rbutton.Click += new EventHandler(this.Submit);

//Add the card to the window

this.Controls.Add(card);

rgroupBox.ResumeLayout(false);

card.ResumeLayout(false);

card.PerformLayout();

this.ResumeLayout(false);

counter++;

}

}

private void Logout(object sender, EventArgs e)

{

LogoutController.Logout(this.userAccount);

this.Close();

}

public void Submit(object sender, EventArgs e)

{

Button btn = sender as Button;

int resID = int.Parse(btn.Name.Split('-')[1]);

//We need to know Which component we've referenced for this. for now i'm using "this.card"

CancelController controller = new CancelController(this);

Debug.WriteLine($"Cancelling Reservation {resID}");

controller.Submit(userAccount, resID);

this.Close(); //Close immediately after sending deets to the Controller

}

public static void Launch(Account userAccount, Entity.List list)

{

new AdminDashboard(userAccount, list).Show();

}

}

partial class AdminDashboard //UI Generated Code

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.logout = new System.Windows.Forms.Button();

this.label1 = new System.Windows.Forms.Label();

this.SuspendLayout();

//

// logout

//

this.logout.AutoSize = true;

this.logout.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);

this.logout.Location = new System.Drawing.Point(1008, 19);

this.logout.Margin = new System.Windows.Forms.Padding(4, 3, 4, 3);

this.logout.Name = "logout";

this.logout.Size = new System.Drawing.Size(154, 67);

this.logout.TabIndex = 2;

this.logout.Text = "Logout";

this.logout.UseVisualStyleBackColor = true;

this.logout.Click += new System.EventHandler(this.Logout);

//

// label1

//

this.label1.AutoSize = true;

this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 16F, ((System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold | System.Drawing.FontStyle.Underline))), System.Drawing.GraphicsUnit.Point);

this.label1.Location = new System.Drawing.Point(35, 42);

this.label1.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(202, 26);

this.label1.TabIndex = 3;

this.label1.Text = "Admin Dashboard";

//

// AdminDashboard

//

this.AutoScaleDimensions = new System.Drawing.SizeF(7F, 15F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.AutoSize = true;

this.ClientSize = new System.Drawing.Size(1230, 715);

this.Controls.Add(this.label1);

this.Controls.Add(this.logout);

this.Margin = new System.Windows.Forms.Padding(4, 3, 4, 3);

this.Name = "AdminDashboard";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "AdminDashboard";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button logout;

private System.Windows.Forms.Label label1;

}

}