```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <Servo.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);

LiquidCrystal_I2C lcd(0x27, 16, 2);  // Change 0x27 if needed

Servo safeServo;

// Keypad Setup
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
```

```
byte rowPins[ROWS] = {2, 3, 4, 5};

byte colPins[COLS] = {6, 7, 8, A0};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);



bool isSafeOpen = false;

bool inMenu = false;

bool inTuneMenu = false;

bool inPasswordChange = false;

bool inRFIDManagement = false;

bool inRFIDRemoval = false;

int menuOption = 0;

const int menuLength = 3;

int menuStart = 0;



int selectedTune = 0; // 0 = Mario Tune, 1 = Fur Elise, 2 = Off

const int buzzerPin = A1;

const int buttonPin = A3; // Button connected to pin A3



String currentPassword = "1234";

String newPassword = "";



const int maxRFIDTags = 10;

byte validRFIDs[maxRFIDTags][4];

int numRFIDTags = 0;
```

```
unsigned long lastActivityTime = 0;
const unsigned long inactivityInterval = 10000; // 10 seconds
unsigned long unlockTime = 0; // Time when the safe was unlocked
int wrongPasswordCount = 0; // Count wrong password attempts

void setup() {
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
  Serial.println("RFID Reader Initialized");

  lcd.init();
  lcd.backlight();
  lcd.clear();
  Serial.println("LCD Initialized");

  safeServo.attach(A2); // Connect the servo motor to pin A2
  pinMode(buzzerPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP); // Set button pin as input with internal pull-up resistor
  lockSafe();
  Serial.println("System Setup Complete");
}
```

```
void loop() {
  char key = keypad.getKey();
  if (key) {
    lastActivityTime = millis();
    lcd.backlight(); // Turn on the backlight on any activity
    Serial.print("Key Pressed: ");
    Serial.println(key);
    if (!isSafeOpen) {
      handleSafeUnlock(key);
    } else {
      if (inTuneMenu) {
        handleTuneSelection(key);
      } else if (inPasswordChange) {
        handleChangePassword(key);
      } else if (inRFIDManagement) {
        handleRFIDManagement(key);
      } else if (inRFIDRemoval) {
        handleRFIDRemoval(key);
      } else if (inMenu) {
        handleMenuSelection(key);
      } else {
        handleMenuNavigation(key);
      }
    }
```

```
}


// Check for RFID tag
if (!isSafeOpen && mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
  lastActivityTime = millis();
  lcd.backlight(); // Turn on the backlight on any activity
  Serial.println("RFID Tag Detected");
  if (isValidRFID()) {
    unlockSafe();
  } else {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Invalid RFID Tag");
    delay(1500);
    showWelcomeScreen();
  }
  mfrc522.PICC_HaltA();
}


// Check if the button is pressed and the safe is open
if (isSafeOpen && digitalRead(buttonPin) == LOW) {
  if (millis() - unlockTime > inactivityInterval) {
    delay(500);
    lockSafe();
```

```
  }
 }

 // Check if the safe is locked and the button is unpushed
 if (!isSafeOpen && digitalRead(buttonPin) == HIGH) {
  unsigned long sirenStartTime = millis();
  while (!isSafeOpen && digitalRead(buttonPin) == HIGH && millis() - sirenStartTime < 5000) {
   playPoliceSiren();
  }
  noTone(buzzerPin); // Stop the buzzer after 5 seconds
 }

 // Turn off the backlight after inactivity
 if (millis() - lastActivityTime > inactivityInterval) {
  lcd.noBacklight();
 }
}

void showWelcomeScreen() {
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Enter Password");
 lcd.setCursor(0, 1);
 lcd.print("Or Scan a Tag");
```

```
}


void lockSafe() {
  Serial.println("Locking Safe");
  safeServo.write(0);
  isSafeOpen = false;
  inMenu = false;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Safe Locked");
  delay(2000);
  showWelcomeScreen();
  Serial.println("Safe Locked");
}


void unlockSafe() {
  Serial.println("Unlocking Safe");
  safeServo.write(90);
  isSafeOpen = true;
  unlockTime = millis(); // Record the time when the safe is unlocked
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Safe Opened");
  playTune();
```

```
  delay(1000);
  showPostUnlockScreen();
  Serial.println("Safe Opened");
}

void showPostUnlockScreen() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Press '1'");
  lcd.setCursor(0, 1);
  lcd.print("for Menu");
}

void handleSafeUnlock(char key) {
  static String enteredCode = "";
  static bool isEnteringPassword = false;

  if (key == '#') {
    enteredCode = "";
    isEnteringPassword = false;
    showWelcomeScreen();
  } else if (isDigit(key)) { // Add this condition to check if the key is a digit
    if (!isEnteringPassword) {
      lcd.clear();
```

```
  lcd.setCursor(0, 0);
  lcd.print("Entering Password...");
  isEnteringPassword = true;
 }
 enteredCode += key;
 lcd.setCursor(0, 1);
 for (int i = 0; i < enteredCode.length(); i++) {
  lcd.print('*');
 }


 if (enteredCode.length() == currentPassword.length()) {
  if (enteredCode == currentPassword) {
   unlockSafe();
   wrongPasswordCount = 0; // Reset the wrong password count
  } else {
   wrongPasswordCount++;
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Wrong Code!");
   Serial.println("Wrong Code Entered");
   delay(1500);
   if (wrongPasswordCount >= 3) {
    unsigned long sirenStartTime = millis();
    while (millis() - sirenStartTime < 5000) {
```

```
          playPoliceSiren(); // Play the siren if wrong password entered 3 times
        }
        noTone(buzzerPin); // Stop the buzzer after 5 seconds
        wrongPasswordCount = 0; // Reset the wrong password count after playing the siren
      }
      showWelcomeScreen();
    }
    enteredCode = "";
    isEnteringPassword = false;
  }
 }
}


void handleMenuNavigation(char key) {
 if (key == '1') {
  inMenu = true;
  menuOption = 0;
  menuStart = 0;
  showMenu();
 }
}


void showMenu() {
 lcd.clear();
```

```
  updateMenuDisplay();
  Serial.println("Menu Displayed");
}

void updateMenuDisplay() {
  lcd.clear();
  for (int i = 0; i < 2; i++) {
    int menuIndex = menuStart + i;
    if (menuIndex < menuLength) {
      lcd.setCursor(0, i); // Start menu options one more space to the right
      lcd.print(menuIndex == menuOption ? "> " : "  ");
      switch (menuIndex) {
        case 0: lcd.print("Change Password"); break;
        case 1: lcd.print("Manage Tags"); break;
        case 2: lcd.print("Opening Tune"); break;
      }
    }
  }
  Serial.print("Menu Option: ");
  Serial.println(menuOption);
}

void handleMenuSelection(char key) {
  if (key == 'A' && menuOption > 0) {
```

```
    menuOption--;
    if (menuOption < menuStart) menuStart--;
  } else if (key == 'B' && menuOption < menuLength - 1) {
    menuOption++;
    if (menuOption >= menuStart + 2) menuStart++;
  } else if (key == '6') {
    if (menuOption == 0) {
      showPasswordChangeScreen(); // Call this function when the "Change Password" option is
selected
    } else if (menuOption == 1) {
      showRFIDManagementMenu();
    } else if (menuOption == 2) {
      showTuneMenu();
    }
  } else if (key == '1') {
    inMenu = false;
    showPostUnlockScreen();
    return;
  }
  updateMenuDisplay();
}


void showTuneMenu() {
  inTuneMenu = true;
```

```
  menuOption = 0;
  lcd.clear();
  updateTuneMenu();
  Serial.println("Tune Menu Displayed");
}


void updateTuneMenu() {
  lcd.clear();
  for (int i = 0; i < 2; i++) {
    int menuIndex = menuStart + i;
    if (menuIndex < 3) {
      lcd.setCursor(0, i); // Start menu options one more space to the right
      lcd.print(menuIndex == menuOption ? "> " : "  ");
      switch (menuIndex) {
        case 0: lcd.print("Mario Tune"); break;
        case 1: lcd.print("Fur Elise"); break;
        case 2: lcd.print("Off"); break;
      }
    }
  }
  Serial.print("Tune Menu Option: ");
  Serial.println(menuOption);
}
```

```
void handleTuneSelection(char key) {
 if (key == 'A' && menuOption > 0) {
   menuOption--;
   if (menuOption < menuStart) menuStart--;
 } else if (key == 'B' && menuOption < 2) {
   menuOption++;
   if (menuOption >= menuStart + 2) menuStart++;
 } else if (key == '6') {
   selectedTune = menuOption;
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Tune Selected!");
   Serial.print("Tune Selected: ");
   Serial.println(selectedTune == 0 ? "Mario Tune" : selectedTune == 1 ? "Fur Elise" : "Off");
   delay(1000);
   inTuneMenu = false;
   showMenu();
 } else if (key == '1') {
   inTuneMenu = false;
   showMenu();
 }
 updateTuneMenu();
}
```

```
void showPasswordChangeScreen() {
  inPasswordChange = true;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Enter Old Password:");
}

void handleChangePassword(char key) {
  static String enteredOldPassword = "";
  static String enteredNewPassword = "";
  static bool enteringOldPassword = true;
  static bool isEnteringPassword = false;

  if (key == '1') {
    enteredOldPassword = "";
    enteredNewPassword = "";
    enteringOldPassword = true;
    isEnteringPassword = false;
    inPasswordChange = false;
    showMenu();
  } else if (key == '#') {
    enteredOldPassword = "";
    enteredNewPassword = "";
    enteringOldPassword = true;
```

```
  isEnteringPassword = false;
  showPasswordChangeScreen();
 } else if (isDigit(key)) { // Add this condition to check if the key is a digit
  if (!isEnteringPassword) {
   lcd.clear();
   lcd.setCursor(0, 0);
   if (enteringOldPassword) {
    lcd.print("Old Password:");
   } else {
    lcd.print("New Password:");
   }
   isEnteringPassword = true;
  }
  if (enteringOldPassword) {
   enteredOldPassword += key;
   lcd.setCursor(0, 1);
   for (int i = 0; i < enteredOldPassword.length(); i++) {
    lcd.print('*');
   }
   if (enteredOldPassword.length() == currentPassword.length()) {
    if (enteredOldPassword == currentPassword) {
     lcd.clear();
     lcd.setCursor(0, 0);
     lcd.print("Enter New Pass:");
```

```
      enteredOldPassword = "";
      enteringOldPassword = false;
      isEnteringPassword = false;
    } else {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Wrong Pass!");
      delay(1500);
      showPasswordChangeScreen();
      enteredOldPassword = "";
      isEnteringPassword = false;
    }
  }
} else {
  enteredNewPassword += key;
  lcd.setCursor(0, 1);
  for (int i = 0; i < enteredNewPassword.length(); i++) {
    lcd.print('*');
  }
  if (enteredNewPassword.length() == currentPassword.length()) {
    newPassword = enteredNewPassword;
    currentPassword = newPassword;
    lcd.clear();
    lcd.setCursor(0, 0);
```

```
        lcd.print("Pass Changed!");
        Serial.println("Password Changed Successfully");
        delay(1500);
        inPasswordChange = false;
        showMenu();
      }
    }
  }
}

void showRFIDManagementMenu() {
  inRFIDManagement = true;
  menuOption = 0;
  lcd.clear();
  updateRFIDManagementMenu();
  Serial.println("RFID Management Menu Displayed");
}

void updateRFIDManagementMenu() {
  lcd.clear();
  lcd.setCursor(0, 0); // Start menu options one more space to the right
  lcd.print(menuOption == 0 ? "> Add Tags" : "  Add Tags");
  lcd.setCursor(0, 1);
  lcd.print(menuOption == 1 ? "> Remove Tags" : "  Remove Tags");
```

```
  Serial.print("Tags Management Option: ");
  Serial.println(menuOption);
}


void handleRFIDManagement(char key) {
  if (key == 'A' && menuOption > 0) {
    menuOption--;
  } else if (key == 'B' && menuOption < 1) {
    menuOption++;
  } else if (key == '6') {
    if (menuOption == 0) {
      addRFIDTag();
    } else if (menuOption == 1) {
      showRFIDRemovalScreen();
    }
  } else if (key == '1') {
    inRFIDManagement = false;
    showMenu();
  }
  updateRFIDManagementMenu();
}


void addRFIDTag() {
  lcd.clear();
```

```
lcd.setCursor(0, 0);
lcd.print("Scan New Tag:");
Serial.println("Waiting for new RFID tag...");
while (!mfrc522.PICC_IsNewCardPresent() || !mfrc522.PICC_ReadCardSerial()) {
  delay(100);
}


if (numRFIDTags < maxRFIDTags) {
  for (int i = 0; i < 4; i++) {
    validRFIDs[numRFIDTags][i] = mfrc522.uid.uidByte[i];
  }
  numRFIDTags++;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Tag Added!");
  Serial.println("New tag added");
} else {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Max Tags Reached");
  Serial.println("Max RFID tags reached");
}
delay(1500);
showRFIDManagementMenu();
```

```
}


void showRFIDRemovalScreen() {
  inRFIDRemoval = true;
  menuOption = 0;
  lcd.clear();
  updateRFIDRemovalScreen();
  Serial.println("RFID Removal Screen Displayed");
}


void updateRFIDRemovalScreen() {
  lcd.clear();
  if (numRFIDTags == 0) {
    lcd.setCursor(0, 0);
    lcd.print("No Tags Stored");
    Serial.println("No Tags Stored");
    delay(1500);
    showRFIDManagementMenu();
    return;
  }


  lcd.setCursor(0, 0); // Start menu options one more space to the right
  lcd.print(menuOption == 0 ? "> " : "  ");
  for (int j = 0; j < 4; j++) {
```

```
    lcd.print(validRFIDs[menuOption][j], HEX);
    if (j < 3) lcd.print(":");
  }
  Serial.print("RFID Removal Option: ");
  Serial.println(menuOption);
}

void handleRFIDRemoval(char key) {
  if (key == 'A' && menuOption > 0) {
    menuOption--;
  } else if (key == 'B' && menuOption < numRFIDTags - 1) {
    menuOption++;
  } else if (key == '6') {
    removeRFIDTag(menuOption);
  } else if (key == '1') {
    inRFIDRemoval = false;
    showRFIDManagementMenu();
  }
  updateRFIDRemovalScreen();
}

void removeRFIDTag(int index) {
  for (int i = index; i < numRFIDTags - 1; i++) {
    for (int j = 0; j < 4; j++) {
```

```
      validRFIDs[i][j] = validRFIDs[i + 1][j];
    }
  }
  numRFIDTags--;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Tag Removed!");
  Serial.println("RFID tag removed");
  delay(1500);
  if (numRFIDTags == 0) {
    inRFIDRemoval = false;
    showRFIDManagementMenu();
  } else {
    updateRFIDRemovalScreen();
  }
}


void playTune() {
  if (selectedTune == 0) {
    playMarioTune();
  } else if (selectedTune == 1) {
    playFurElise();
  }
}
```

```
void playMarioTune() {
  int melody[] = {262, 262, 392, 392, 440, 440, 392};
  int duration[] = {300, 300, 300, 300, 300, 300, 600};
  for (int i = 0; i < 7; i++) {
    tone(buzzerPin, melody[i], duration[i]);
    delay(duration[i] + 50);
  }
  Serial.println("Played Mario Tune");
}

void playFurElise() {
  int melody[] = {330, 311, 330, 311, 330, 247, 311, 262};
  int duration[] = {300, 300, 300, 300, 300, 300, 300, 600};
  for (int i = 0; i < 8; i++) {
    tone(buzzerPin, melody[i], duration[i]);
    delay(duration[i] + 50);
  }
  Serial.println("Played Fur Elise");
}

void playPoliceSiren() {
  for (int i = 440; i < 1000; i += 10) { // Increase pitch (woo)
    tone(buzzerPin, i);
```

```
    delay(20);
  }
  for (int i = 1000; i > 440; i -= 10) { // Decrease pitch (woo)
    tone(buzzerPin, i);
    delay(20);
  }
}

bool isValidRFID() {
  for (int i = 0; i < numRFIDTags; i++) {
    bool match = true;
    for (int j = 0; j < 4; j++) {
      if (validRFIDs[i][j] != mfrc522.uid.uidByte[j]) {
        match = false;
        break;
      }
    }
    if (match) {
      return true;
    }
  }
  return false;
}
```