```
#include <SPI.h>              // SPI communication library for RFID and SD card
#include <MFRC522.h>          // RFID library
#include <Wire.h>             // Library for I2C communication
#include <LiquidCrystal_I2C.h> // Library for I2C LCD
#include <RTClib.h>           // Real-time clock library
#include <Arduino.h>          // Standard Arduino library
#include <SD.h>               // SD card library for file operations

#define SS_PIN 3              // Slave Select pin for RFID
#define RST_PIN 2             // Reset pin for RFID
#define LED_PIN 8             // LED pin for feedback
#define SD_CS_PIN 10          // Chip select pin for SD card

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create an instance of the RFID reader
RTC_DS3231 rtc;                    // Create an instance of the DS3231 RTC
LiquidCrystal_I2C lcd(0x27, 16, 2); // Set up the LCD with I2C address 0x27, 16 columns, and 2 rows

// Array of valid RFID UIDs (Unique Identifier) for students
byte validUIDs[][4] = {
  {0x2D, 0xB3, 0x9A, 0x3F},  // UID of student 1
  {0x26, 0x27, 0xCA, 0xC6},  // UID of student 2
  {0x7D, 0xAF, 0xBB, 0x3F},  // UID of student 3
  {0x96, 0x26, 0xC9, 0xC6},  // UID of student 4
  {0x7E, 0x4C, 0x31, 0x60},  // UID of student 5
};

// Names corresponding to each student's UID
const char* studentNames[] = {
  "NISHA TOLENTINO",
  "JANSSEN HUGO",
  "NEZEL BAYNOSA",
  "XYMON LOPEZ",
  "PAUL BIARES",
};

struct Schedule {
  int startHour;
  int startMinute;
  int endHour;
  int endMinute;
};

Schedule schedules[] = {
  {7, 20, 7, 35}, {7, 35, 8, 35}, {9, 5, 10, 5}, {10, 5, 11, 5}, {11, 5, 12, 5},
  {13, 10, 14, 10}, {14, 30, 15, 30}, {15, 30, 16, 30}, {16, 30, 17, 30}
};
```

```
void setup() {
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
  lcd.begin(16, 2);
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("Scan ID card");
  pinMode(LED_PIN, OUTPUT);

  if (!rtc.begin()) {
    Serial.println("RTC initialization failed!");
    lcd.setCursor(0, 1);
    lcd.print("RTC Init Failed");
    while (1);
  }
  if (rtc.lostPower()) {
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }

  if (!SD.begin(SD_CS_PIN)) {
    Serial.println("SD card initialization failed!");
    lcd.setCursor(0, 1);
    lcd.print("SD Init Failed!");
    while (1);
  }
  Serial.println("SD card initialized successfully.");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Ready to scan!");
}

void loop() {
  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
    String content = "";
    for (byte i = 0; i < mfrc522.uid.size; i++) {
      content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : ""));
      content.concat(String(mfrc522.uid.uidByte[i], HEX));
    }
    content.toUpperCase();

    bool uidMatched = false;
    int studentIndex = -1;
    for (int i = 0; i < sizeof(validUIDs) / sizeof(validUIDs[0]); i++) {
      if (memcmp(mfrc522.uid.uidByte, validUIDs[i], mfrc522.uid.size) == 0) {
        uidMatched = true;
```

```
        studentIndex = i;
        break;
      }
    }

    if (uidMatched) {
      DateTime now = rtc.now();
      const char* status = getAttendanceStatus(now.hour(), now.minute());
      digitalWrite(LED_PIN, HIGH);
      delay(1000);
      digitalWrite(LED_PIN, LOW);
      updateAttendance(studentIndex + 1, status, now);
      delay(3000);
    } else {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("ID doesn't match");
      delay(2000);
    }

    lcd.clear();
    lcd.setCursor(2, 0);
    lcd.print("Scan ID card");
    mfrc522.PICC_HaltA();
    mfrc522.PCD_StopCrypto1();
  }
}

const char* getAttendanceStatus(int hour, int minute) {
  for (Schedule sched : schedules) {
    if (hour == sched.startHour && minute >= sched.startMinute && minute <
sched.startMinute + 10) {
      return "Present";
    } else if (hour == sched.startHour && minute >= sched.startMinute + 10 && minute <
sched.startMinute + 20) {
      return "Late";
    } else if ((hour == sched.startHour && minute >= sched.startMinute + 20) || (hour >
sched.startHour && hour < sched.endHour) || (hour == sched.endHour && minute <=
sched.endMinute)) {
      return "Absent";
    }
  }
  return "No Class";
}

void updateAttendance(int studentID, const char* status, DateTime timestamp) {
  lcd.clear();
  lcd.setCursor(0, 0);
```

```
  lcd.print(studentNames[studentID - 1]);
  lcd.setCursor(0, 1);
  lcd.print(status);
  lcd.setCursor(8, 1);
  lcd.print(timestamp.hour(), DEC);
  lcd.print(":");
  lcd.print(timestamp.minute(), DEC);

  char filename[12];
  sprintf(filename, "%02d-%02d-%02d.txt", timestamp.year() % 100, timestamp.month(),
timestamp.day());

  File dataFile = SD.open(filename, FILE_WRITE);
  if (dataFile) {
    dataFile.print(studentNames[studentID - 1]);
    dataFile.print(",");
    dataFile.print(status);
    dataFile.print(",");
    dataFile.print(timestamp.hour());
    dataFile.print(":");
    dataFile.print(timestamp.minute());
    dataFile.println();
    dataFile.close();
    Serial.println("Attendance logged successfully!");
  } else {
    Serial.println("Error opening file!");
    lcd.setCursor(0, 1);
    lcd.print("SD Write Failed");
    delay(2000);
  }
}
```