SYNOPSIS

The NFC-Based Attendance Management Tracker offers a streamlined solution to traditional attendance tracking methods by leveraging Near Field Communication (NFC) technology. Using an ESP32 microcontroller, PN532 NFC module, LCD display, and buttons, the system allows for effortless user registration and attendance recording. Administrators can register new users by tapping their NFC ID cards, while registered users can mark their attendance by tapping their cards on the designated reader. The system provides real-time feedback through the LCD display, ensuring transparency and accuracy in attendance recording. Attendance data is securely stored in a cloud-based database for easy access and analysis. With its intuitive interface and efficient functionality, the system aims to enhance productivity and reliability in attendance management for educational institutions and organizations.

Table Of Contents

Problem Statement

The traditional manual attendance system utilized by college teachers poses significant inefficiencies and challenges. The current process demands teachers to manually call out student roll numbers and mark attendance, leading to time wastage, potential errors, and lack of comprehensive attendance data. These shortcomings undermine the effectiveness of the attendance monitoring process and fail to provide meaningful insights or feedback to both students and teachers.

Our project aims to address these issues by leveraging existing NFC-enabled IDs already in circulation among students and teachers. We propose the development of a smart attendance system that revolutionizes the way attendance is managed in the college environment. By implementing NFC technology, students will be able to conveniently tap their IDs on a NFC reader device, enabling automatic attendance recording directly into a centralized spreadsheet system.

Furthermore, our solution will incorporate a user-friendly web interface tailored for teachers, offering functionalities to effortlessly view, edit, and export attendance data for their respective classes. Additionally, the system will generate comprehensive reports and insights, highlighting attendance trends, correlations, and anomalies. These analytics will serve as valuable tools for both students and teachers to monitor attendance patterns, enhance engagement, and improve overall learning outcomes.

Our project's overarching goal is to enhance the efficiency, accuracy, and convenience of the attendance system, ultimately fostering a conducive learning environment that promotes student success and satisfaction.

Objective and Scope

The objective of this project is to develop an NFC-based attendance management system that automates the process of recording attendance in educational institutions and organizations.

The scope of this project encompasses the integration of NFC technology, microcontroller programming, and Google Sheets connectivity to create a comprehensive solution for automating and authenticating attendance. The project will involve designing and developing a hardware system comprising a NFC reader module, a microcontroller board, and a LCD display. This system will read NFC tags of students and teachers, verify their identities, and display their names and attendance status on the LCD screen. Additionally, the microcontroller board will be programmed to communicate with the NFC reader module and LCD display, as well as to send attendance data to a Google Sheets spreadsheet via Wi-Fi or GSM module. Security features such as encryption and authentication will be implemented to safeguard attendance data. Furthermore, a user-friendly interface will be created within the Google Sheets spreadsheet, allowing teachers to view, edit, and export attendance data for their classes. The project will also include testing and evaluation of hardware and software components, as well as consideration of ethical, social, and environmental implications.

In addition to the outlined scope, several enhancements will be incorporated into the project to further improve functionality and usability. These enhancements include the integration of a feedback mechanism for students and teachers to provide input on the attendance process, development of a mobile application companion to the web interface for increased accessibility, implementation of automated notifications to alert users of upcoming classes and absentees, and incorporation of advanced data analytics and visualization tools within the Google Sheets spreadsheet. Security measures will be strengthened through the implementation of multi-factor authentication, and the system architecture will be designed to be scalable and customizable for future expansion. Accessibility features will also be integrated to ensure inclusivity for users with disabilities, while considerations for environmental sustainability will be made to minimize energy consumption and waste generation. These enhancements aim to deliver a robust, user-friendly, and sustainable solution that meets the evolving needs of both students and teachers.

System Analysis and Design

The NFC-Based Attendance Management System is designed to automate and streamline the attendance tracking process within our college. This report presents the results of the system analysis and design phase, which focused on understanding requirements, defining system architecture, and designing components to meet stakeholders' needs effectively.

Requirement Analysis:

1. Stakeholder Identification: Stakeholders including administrators, teachers, and students were identified, and their requirements were gathered through interviews and surveys.
2. Requirements Elicitation: Functional and non-functional requirements were documented, prioritized, and validated to ensure alignment with stakeholders' needs and expectations.
3. Existing Process Analysis: The current manual attendance tracking process was analyzed to identify inefficiencies and areas for improvement.

System Architecture:

1. Component Definition: Hardware components such as the ESP32 microcontroller, PN532 NFC module, LCD display, and software components like the database and user interface were defined.
2. Architectural Design: The system architecture was designed to facilitate communication and interaction between hardware and software components, ensuring scalability and flexibility.

Component Design:

1. Database Design: The database schema was defined to store user information, attendance records, and course details efficiently.
2. Module Design: Modules for user registration, attendance recording, and feedback generation were designed, focusing on modularity and reusability.

3. Interface Design: User interfaces for interaction with the system, including navigation controls and real-time feedback mechanisms, were designed for ease of use and intuitive navigation.

Integration and Testing:

1. Continuous Integration: Components were integrated continuously to detect integration issues early and ensure smooth interactions between hardware and software components.
2. Unit Testing: Individual modules and components were tested to verify their functionality, and system-level testing was conducted to validate the overall behavior and performance of the system.

Documentation:

1. Requirements Specification: Detailed requirements specifications, including use cases and acceptance criteria, were documented to provide a clear understanding of the system's functionality.
2. Design Documentation: Architectural diagrams, component diagrams, and sequence diagrams were created to illustrate the system's design and interactions between components.
3. User Documentation: User manuals, guides, and tutorials were prepared to assist administrators, teachers, and students in deploying, configuring, and using the system effectively.

Software Engineering Paradigm

The chosen software engineering paradigm for this project is Iterative and Incremental Development (IID). IID emphasizes the iterative refinement of the system through repeated cycles of planning, implementation, testing, and feedback incorporation. It allows for flexibility, adaptability, and continuous improvement based on stakeholder feedback and evolving requirements.

Development Process:

Weekly Planning: The project team conducted weekly planning sessions to set objectives and prioritize tasks for the upcoming week. Goals were established based on the project timeline and requirements.

Iterative Development: Each week, the team focused on specific modules or functionalities of the system. For example, the initial focus was on making the scanning module work reliably with the PN532 NFC module.

Incremental Enhancements: As each module was completed, additional functionalities were incrementally added to the system. For instance, after achieving successful NFC scanning, the team integrated Google Sheets functionality for attendance logging.

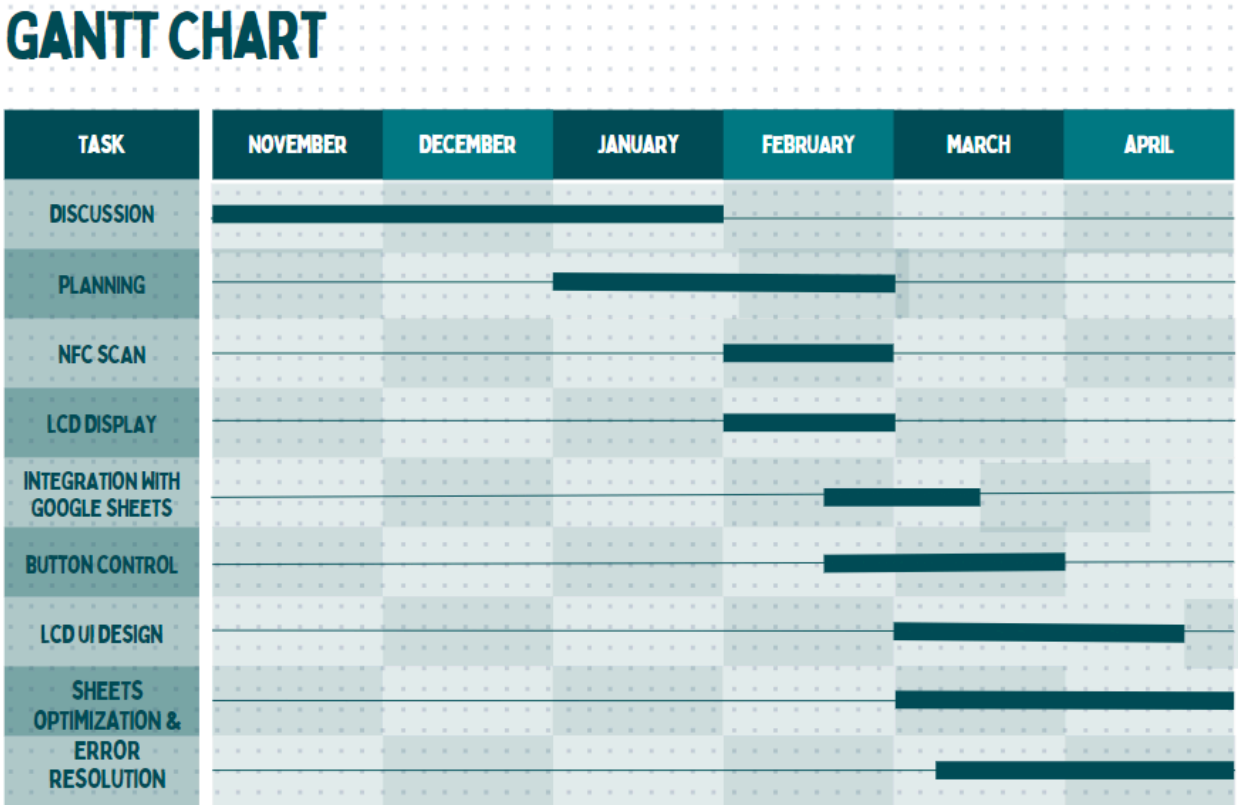
Testing and Feedback: Throughout the development process, regular testing sessions were conducted to identify bugs, verify functionality, and ensure system reliability. Feedback from these testing sessions, along with guidance from the project guide, informed subsequent iterations and refinements.

Integration of Components: Hardware and software components were integrated iteratively, ensuring seamless communication and interaction between them. Integration testing sessions were held to validate system integration and identify any integration issues.

Continuous Improvement: The project team continuously improved the system based on feedback received from stakeholders and their own observations. Iterations were planned based on lessons learned and evolving project requirements.

The application of the Iterative and Incremental Development paradigm allowed for the systematic development of the NFC-Based Attendance Management System. By breaking down the project into manageable iterations and incrementally adding features, the team achieved steady progress while maintaining flexibility to accommodate changes and feedback. The iterative approach facilitated regular communication and collaboration, ensuring alignment with stakeholder expectations and project objectives.

Gantt Chart

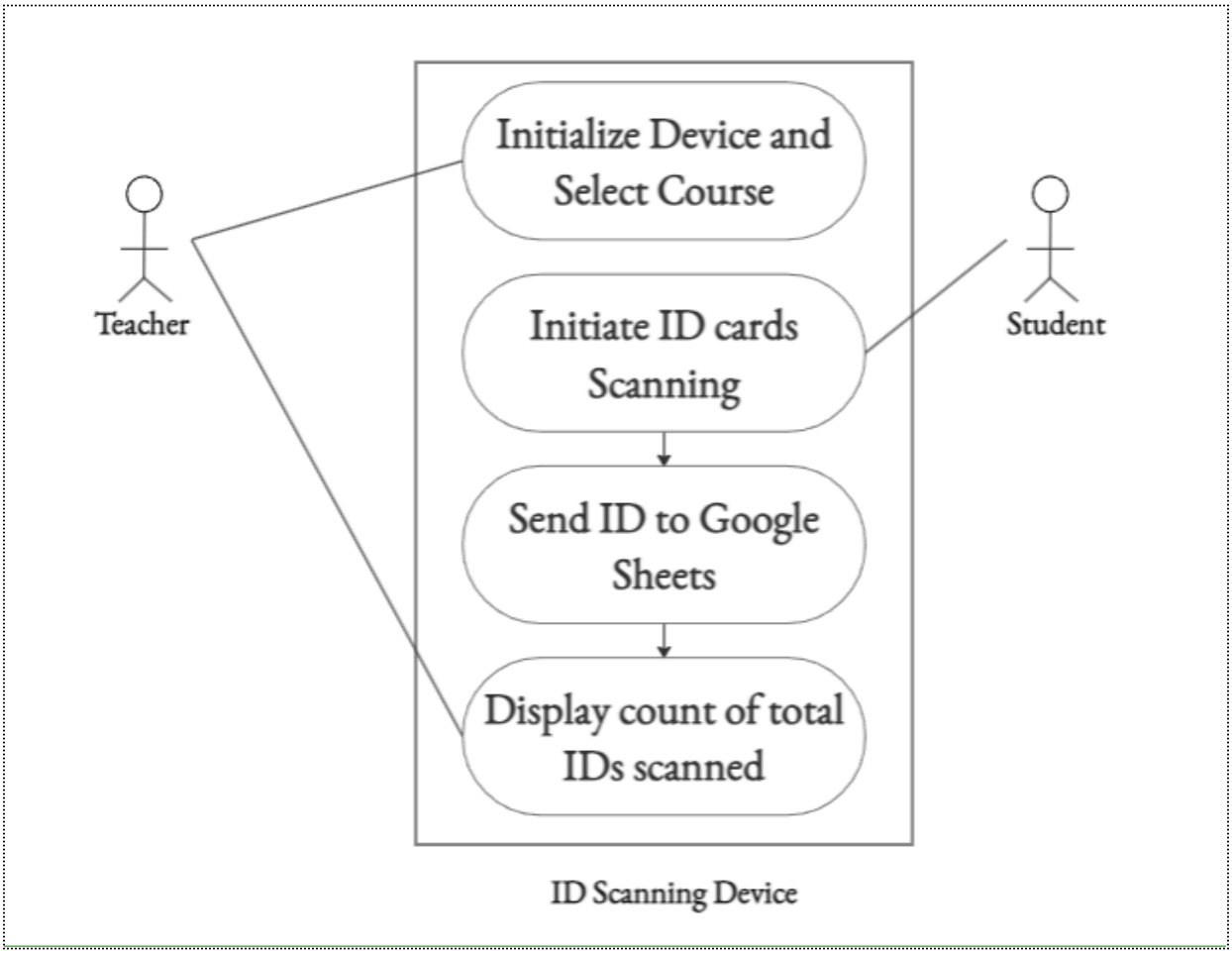


Discussion and Planning: This initial phase consumed the most time. Teams engaged in thorough discussions, laying the groundwork for the project. Decisions on scope, resources, and milestones were made during this period.

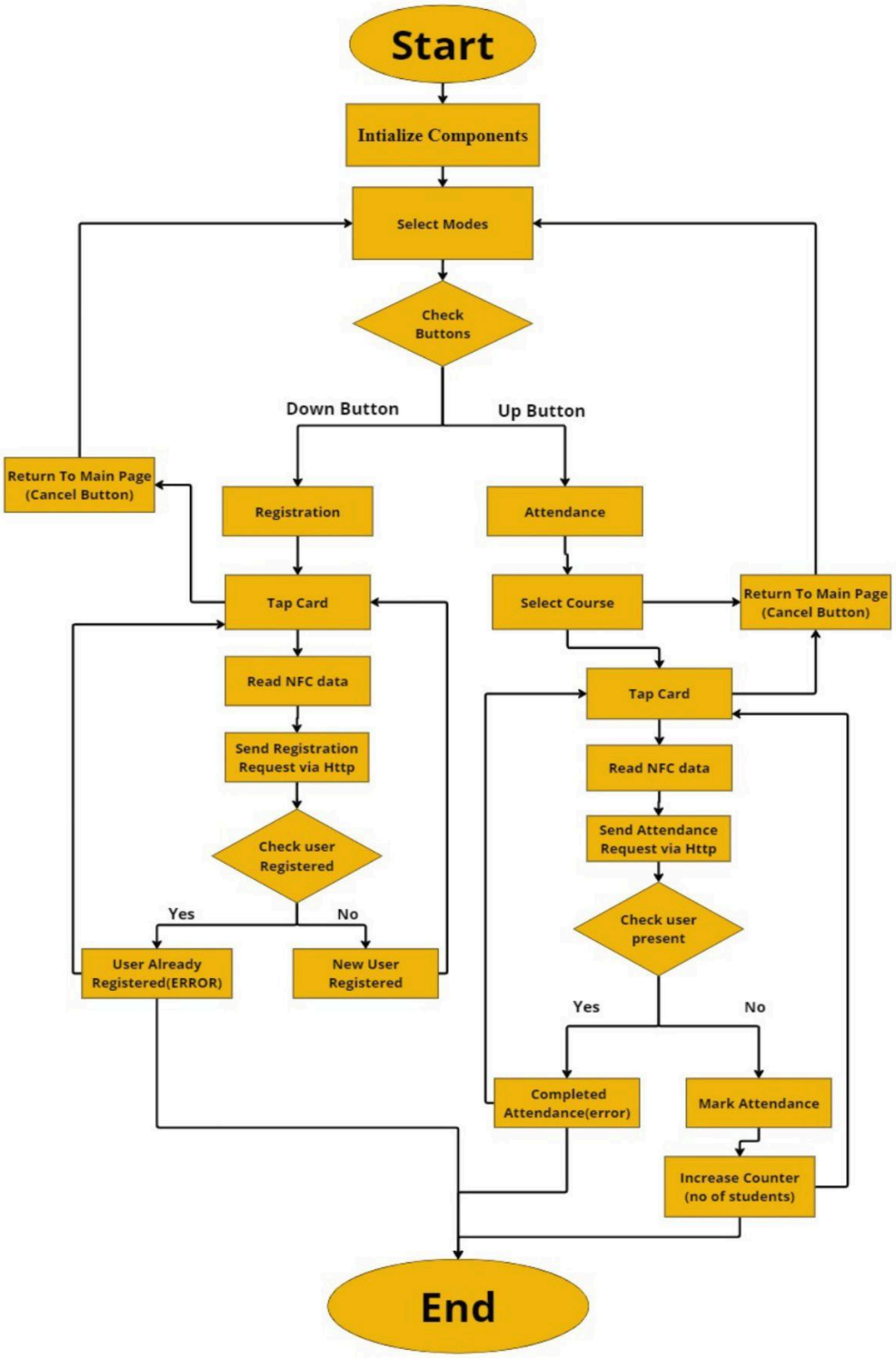
Testing: Following planning, rigorous testing took place. It was a critical phase to ensure quality and functionality. However, testing also incurred significant costs due to specialized tools and personnel.

Implementation: Once testing was complete, the project moved into implementation. Actual development work occurred here, translating plans into reality. This phase was relatively swift compared to the preceding stages.

Use Case Diagram



Flow Chart


Start
Intialize Components
Select Modes
Check Buttons
Down Button
Up Button
Return To Main Page (Cancel Button)
Registration
Attendance
Tap Card
Select Course
Return To Main Page (Cancel Button)
Read NFC data
Send Registration Request via Http
Check user Registered
Yes
No
User Already Registered(ERROR)
New User Registered
Tap Card
Read NFC data
Send Attendance Request via Http
Check user present
Yes
No
Completed Attendance(error)
Mark Attendance
Increase Counter (no of students)
End

Feasibility Analysis

Technical Feasibility:

- The proposed system utilizes readily available hardware components, including the ESP32 microcontroller and PN532 NFC module, both of which are widely used and supported in the development community.
- The system architecture is straightforward, consisting of the ESP32 microcontroller interfaced with the PN532 NFC module and an LCD display for user interaction.
- The software development for the system involves programming the microcontroller to handle NFC card scanning, data processing, and communication with the Google Sheets API for attendance logging.
- Technical documentation and online resources are abundant for developing similar projects, ensuring accessibility to necessary technical knowledge and support.

Economical Feasibility:

- The primary cost consideration for the project is the hardware components required for each module. Based on market research and supplier quotations, the total cost for constructing a single module falls well within the allocated budget of Rs. 1200.
- Bulk purchasing of components further reduces the unit cost, enabling cost-effective procurement for the entire college.
- Additionally, the system's low power consumption and minimal maintenance requirements contribute to long-term cost savings for the college.

Operational Feasibility:

- The proposed system is designed with simplicity and ease of use in mind, making it accessible to administrators, teachers, and students alike.
- Operationally, the system offers a seamless user experience, with intuitive navigation controls and real-time feedback through the LCD display.
- Implementation of the system across various departments and classrooms within the college is feasible, thanks to its modular design and scalability.
- Training sessions and user manuals will be provided to ensure smooth deployment and adoption of the system by college staff and students.

Coding

```
//_____________________________________________________________________________
___http_Req()
// Subroutine for sending HTTP requests to Google Sheets.
void http_Req(String str_modes, String str_uid, String courseName) {
  if (WiFi.status() == WL_CONNECTED) {
    String http_req_url = "";

    //---------------------------------------Create links to make HTTP requests to Google Sheets.
    if (str_modes == "atc") {
      http_req_url  = Web_App_URL + "?sts=atc";
      http_req_url += "&uid=" + str_uid;
      http_req_url += "&course=" + courseName; // Add course name to the URL
    }
    if (str_modes == "reg") {
      http_req_url = Web_App_URL + "?sts=reg";
      http_req_url += "&uid=" + str_uid;
    }
    //-----------------------------------------

    //---------------------------------------Sending HTTP requests to Google Sheets.
    Serial.println();
    Serial.println("-------------");
    Serial.println("Sending request to Google Sheets...");
    Serial.print("URL : ");
    Serial.println(http_req_url);

    // Create an HTTPClient object as "http".
    HTTPClient http;

    // HTTP GET Request.
    http.begin(http_req_url.c_str());
    http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);

    // Gets the HTTP status code.
    int httpCode = http.GET();
    Serial.print("HTTP Status Code : ");
    Serial.println(httpCode);

    // Getting response from google sheet.
    String payload;
    if (httpCode > 0) {
      payload = http.getString();
      Serial.println("Payload : " + payload);
    }

    Serial.println("-------------");
    http.end();
    //-----------------------------------------
```

```
  String sts_Res = getValue(payload, ',', 0);

  //--------------------------------------Conditions that are executed are based on the payload response from Google Sheets (the payload response is set in Google Apps Script).
  if (sts_Res == "OK") {
    //.....................
    if (str_modes == "atc") {
      atc_Info = getValue(payload, ',', 1);

      if (atc_Info == "TI_Successful") {
        atc_Name = getValue(payload, ',', 2);
        atc_Date = getValue(payload, ',', 3);
        atc_Time_In = getValue(payload, ',', 4);

        //::::::::::::::::::Create a position value for displaying "Name" on the LCD so that it is centered.
        int name_Lenght = atc_Name.length();
        int pos = 0;
        if (name_Lenght > 0 && name_Lenght <= lcdColumns) {
          pos = map(name_Lenght, 1, lcdColumns, 0, (lcdColumns / 2) - 1);
          pos = ((lcdColumns / 2) - 1) - pos;
        } else if (name_Lenght > lcdColumns) {
          atc_Name = atc_Name.substring(0, lcdColumns);
        }
        //::::::::::::::::::

        lcd.clear();
        delay(500);
        lcd.setCursor(3,0);
        lcd.print(atc_Name);
        lcd.setCursor(5,1);
        lcd.print("DONE");
        delay(3000);
        lcd.clear();
        delay(500);
      }

      if (atc_Info == "P_Successful") {
        atc_Name = getValue(payload, ',', 2);
        atc_Date = getValue(payload, ',', 3);
        atc_Time_In = getValue(payload, ',', 4);
        atc_Time_Out = getValue(payload, ',', 5);

        //::::::::::::::::::Create a position value for displaying "Name" on the LCD so that it is centered.
        int name_Lenght = atc_Name.length();
        int pos = 0;
        if (name_Lenght > 0 && name_Lenght <= lcdColumns) {
          pos = map(name_Lenght, 1, lcdColumns, 0, (lcdColumns / 2) - 1);
          pos = ((lcdColumns / 2) - 1) - pos;
        } else if (name_Lenght > lcdColumns) {
```

```
      atc_Name = atc_Name.substring(0, lcdColumns);
    }
    //::::::::::::::::::

    lcd.clear();
    delay(500);
    lcd.setCursor(3,0);
    lcd.print(atc_Name);
    lcd.setCursor(5,1);
    lcd.print("DONE ");
    delay(3000);
    lcd.clear();
    delay(500);
  }

  if (atc_Info == "atcErr02") {
    counter--;
    lcd.clear();
    delay(500);
    lcd.setCursor(0,0);
    lcd.print("You have completed");
    lcd.setCursor(0,1);
    lcd.print("your  attendance");
    delay(5000);
    lcd.clear();
    delay(500);
  }

  if (atc_Info == "atcErr01") {
    counter--;
    lcd.clear();
    delay(500);
    lcd.setCursor(6,0);
    lcd.print("Error !");
    lcd.setCursor(1,1);
    lcd.print("Not Registered.");
    delay(5000);
    lcd.clear();
    delay(500);
  }

  atc_Info = "";
  atc_Name = "";
  atc_Date = "";
  atc_Time_In = "";
  atc_Time_Out = "";
}
//.................

//.................
if (str_modes == "reg") {
```

```
        reg_Info = getValue(payload, ',', 1);

        if (reg_Info == "R_Successful") {
          lcd.clear();
          delay(500);
          lcd.setCursor(2,0);
          lcd.print("Registration");
          lcd.setCursor(3,1);
          lcd.print("Successful!");
          delay(5000);
          lcd.clear();
          delay(500);
        }

        if (reg_Info == "regErr01") {
          lcd.clear();
          delay(500);
          lcd.setCursor(6,0);
          lcd.print("Error !");
          lcd.setCursor(0,1);
          lcd.print("Already Registered");
          delay(5000);
          lcd.clear();
          delay(500);
        }

        reg_Info = "";
      }
      //.....................
    }
    //--------------------------------------
  } else {
    lcd.clear();
    delay(500);
    lcd.setCursor(0,0);
    lcd.print("Error !");
    lcd.setCursor(1,1);
    lcd.print("WiFi disconnected");
    delay(3000);
    lcd.clear();
    delay(500);
  }
}

//_____________________________________________________________________________________
int getUID() {
  uint8_t success;
  uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };// Buffer to store UID
  uint8_t uidLength; // Length of UID (4 or 7 bytes depending on tag type)
  Serial.print(" nfc scanning: ");

  delay(3500);//PRESS CANCEL BUTTON WITHIN THIS TIME OR SCAN A CARD.
```

```
  // Continuous check for cancel button before and during card reading
  while (true) {
    if (digitalRead(CANCEL_PIN) == LOW) {
      return 0; // Exit the loop
    }

  success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength);

   if (success) {
    Serial.print(" loopcheck: ");

     UID_Result = "";

    for (uint8_t i = 0; i < uidLength; i++) {
      if (uid[i] < 0x10) {
        // If the byte is less than 0x10, pad it with a leading zero
        UID_Result += "0";
      }
      UID_Result += String(uid[i], HEX); // Convert byte to hexadecimal string
      if (i < uidLength - 1) {
        UID_Result += "-"; // Add a dash as separator between bytes
      } }

    Serial.print("  UID Length: ");Serial.print(uidLength, DEC);Serial.println(" bytes");
    Serial.print("  UID Value: ");
    nfc.PrintHex(uid, uidLength);
Serial.println(UID_Result);
    if (uidLength == 4) {
      // We probably have a Mifare Classic card ...
      uint32_t cardid = uid[0];
      cardid <<= 8;
      cardid |= uid[1];
      cardid <<= 8;
      cardid |= uid[2];
      cardid <<= 8;
      cardid |= uid[3];

      Serial.print("Seems to be a Mifare Classic card #");
      Serial.println(cardid);
    }
    // Add a short delay after updating the LCD display
    delay(1000);
    return 1; //SUCCESS
  }
  else{
    return 0;  //FAILURE
  }
  }
}

void loop() {
```

```
lcd.clear();
lcd.setCursor(3, 0);
lcd.print("Select mode");
lcd.setCursor(0, 1);
lcd.write((byte)0); //up - reg
lcd.setCursor(1, 1);
lcd.print("-REG");
lcd.setCursor(11, 1);
lcd.print("ATC-");
lcd.setCursor(15, 1);
lcd.write((byte)1); //down - atc
delay(500);

 if (digitalRead(DOWN_BTN_PIN) == LOW) {
Serial.println("loop1");
modes = "atc";
selectCourseAndReadCard();
}

// Check the state of the REG button
if (digitalRead(UP_BTN_PIN) == LOW) {
  Serial.println("loop reg");
    modes = "reg";

 while(modes=="reg"){
  lcd.clear();
  lcd.setCursor(2,0);
  lcd.print("REGISTRATION");
  lcd.setCursor(3,1);
  lcd.print("Tap card");
  delay(1000);

  readsuccess = getUID();
  if (readsuccess){
    lcd.clear();
    delay(500);
    lcd.setCursor(0,0);
    lcd.print("Getting  UID");
    lcd.setCursor(0,1);
    lcd.print("Please wait...");
    delay(1000);

    http_Req(modes, UID_Result, "");

    pinMode(CANCEL_PIN, INPUT_PULLUP);
  }
  else{
    modes="";
    lcd.clear();
    Serial.print("BREAK");
    break;
```

```
    }

    if (digitalRead(CANCEL_PIN) == LOW) {
          break; // Exit the function
        }
    delay(1000); // Delay for button debouncing
   } //while
  }  //reg

  delay(10);
}
void displayCourse(int index) {
  lcd.setCursor(0, 1);
  lcd.print(" ");

  if (index == 0) {
    lcd.print(" ");
  } else {
    lcd.write((byte)0); // Left arrow
  }
  lcd.print(courses[index]);

  if (index == numCourses - 1) {
    lcd.print(" ");
  } else {
    lcd.write((byte)1); // Right arrow
  }}
```

Google AppScript Code

```
  //---------------------------------------Conditions for registering new users.
    if (sts_val == 'reg') {
      var check_new_UID = checkUID(sheet_id, sheet_UD, 2, uid_val);

      // Conditions when the UID has been registered. Then registration was cancelled.
      if (check_new_UID == true) {
        result += ",regErr01"; // Err_01 = UID is already registered.

        // Sends response payload to ESP32.
        return ContentService.createTextOutput(result);
      }

      // Writes the new user's UID to the "user data" sheet.
      var getLastRowUIDCol = findLastRow(sheet_id, sheet_UD, "B");  // Look for a row to write the
new user's UID.
      var newUID = sheet_open.getRange("B" + (getLastRowUIDCol + 1));
      newUID.setValue(uid_val);
      result += ",R_Successful";

      // Sends response payload to ESP32.
      return ContentService.createTextOutput(result);
```

```
    }
    //--------------------------------------

    if (sts_val == 'atc') {
  // Checks whether the UID is already registered in the "user data" sheet.
  // findUID(Spreadsheet ID, sheet name, index column, UID value)
  // index column : 1 = column A, 2 = column B and so on.
  var FUID = findUID(sheet_id, sheet_UD, 2, uid_val);

  // "(FUID == -1)" means that the UID has not been registered in the "user data" sheet, so
attendance filling is rejected.
  if (FUID == -1) {
    result += ",atcErr01"; // atcErr01 = UID not registered.
    return ContentService.createTextOutput(result);
  } else {
    // After the UID has been checked and the result is that the UID has been registered,
    // then take the "name" of the UID owner from the "user data" sheet.
    // The name of the UID owner is in column "A" on the "user data" sheet.
    var get_Range = sheet_user_data.getRange("A" + (FUID+2));
    var user_name_by_UID = get_Range.getValue();

    // Check if the user is already marked present in the course sheet
var courseSheet = sheet_open.getSheetByName(courseName);
var headers = courseSheet.getRange(1, 1, 1,
courseSheet.getLastColumn()).getDisplayValues()[0]; // Change to getDisplayValues()
var currentDate = Utilities.formatDate(new Date(), "Asia/Kolkata", "dd/MM/yyyy");
    var dateColumnIndex = headers.indexOf(currentDate) + 1; // Change to currentDate
    var dateExists = headers.includes(currentDate); // Change to currentDate


if (!dateExists) {
  // Date column doesn't exist, add new column
  var lastColumn = courseSheet.getLastColumn();
  courseSheet.getRange(1, lastColumn + 1).setValue(currentDate);
  dateColumnIndex = lastColumn + 1;
}

// Now, check if the user is already marked present for today
if (dateExists) {
  var userDataRange = courseSheet.getRange("A2:A").getValues();
  var userNames = userDataRange.flat();

  // Check if the user's name is in the userNames array
  if (userNames.includes(user_name_by_UID)) {
    // Check if the user is already marked as present ("P") for today
    var userRowIndex = userNames.indexOf(user_name_by_UID);
    var presentToday = courseSheet.getRange(userRowIndex + 2, dateColumnIndex).getValue();

    if (presentToday == "P") {
      result += ",atcErr02"; // User already marked present today
      return ContentService.createTextOutput(result);
    }
```

```
  }
}

// Marks attendance in the course sheet with "P"
markAttendance(courseSheet, user_name_by_UID, dateColumnIndex);

// Sends response payload to ESP32.
result += ",P_Successful" + "," + user_name_by_UID + "," + currentDate + "," + dateExists +",," +
"," + headers;
return ContentService.createTextOutput(result);
    //----------------------------------------
  }
}

  }
}
```

Mark Attendance Function:

```
function markAttendance(sheet, userName, dateColumnIndex) {

  columnIndex=dateColumnIndex;

  var lastRow = sheet.getLastRow();
  var range = sheet.getRange(2, 1, lastRow - 1, 1);
  var values = range.getValues();

  for (var i = 0; i < values.length; i++) {
    if (values[i][0] == userName) {
        sheet.getRange(i + 2, columnIndex).setValue("P");
        break;
    }
  }
)
```

Code Efficiency

This project, designed to streamline the attendance management process, showcases an exemplary model of code efficiency, leveraging the robust capabilities of both hardware and software components to deliver a seamless, real-time attendance tracking solution.

One of the standout features of the code's efficiency is its ability to handle date and time discrepancies, a common challenge in attendance systems. Through clever programming, the system accurately recognizes and records attendance dates, even when discrepancies arise from different time zones or formats. This is achieved by employing standardized date formatting and comparison techniques, ensuring that each attendance log is precise and consistent, regardless of the user's locale or the device's clock settings.

Furthermore, the code's structure emphasizes modularity and readability, making it accessible for future enhancements or troubleshooting. This modularity allows for easy integration of additional features, such as automated alerts for absenteeism or integration with educational tools, without necessitating a complete overhaul of the existing system. The clear segmentation of functionality within the codebase also facilitates easier maintenance and debugging, reducing downtime and ensuring that the system remains operational with minimal interruptions.

Another pillar of the project's code efficiency is its proactive approach to error handling and data validation. By incorporating robust error-checking mechanisms, the system can gracefully handle unexpected inputs or system errors, thereby preventing data corruption and ensuring the integrity of the attendance records. This level of diligence not only protects the system from potential malfunctions but also guarantees that the data remains reliable and actionable for administrative purposes.

In conclusion, the automated attendance system utilizing ESP32 and Google Sheets is a testament to the power of efficient coding practices in the development of innovative solutions. It exemplifies how thoughtful programming and system design can transform routine administrative tasks into streamlined, digital processes, thereby enhancing operational efficiency and providing valuable insights into attendance patterns.

Testing

For the attendance management system integrating ESP32 with Google Sheets, a combination of black-box and white-box testing methodologies was employed to ensure thorough evaluation of the system's functionality and reliability.

Black-box testing, focusing on the system's external behavior without delving into its internal logic, was instrumental in validating user interactions and system responses. Test scenarios were crafted to simulate various user actions, such as registering new users and marking attendance.

On the other hand, white-box testing, which examines the internal structure and implementation of the system, was utilized to scrutinize code logic, error handling mechanisms, and data processing algorithms. Through code reviews, static analysis, and dynamic testing techniques, white-box testing helped identify potential vulnerabilities, edge cases, and performance bottlenecks. Unit tests were developed to validate individual functions and modules, ensuring that each component of the system behaves as expected under various conditions.

Unit Testing:

1. PN532 Scan Check:

- Test Objective: Verify that the PN532 NFC module can successfully scan NFC cards.
- Test Steps:
  1. Place a valid NFC card within the range of the PN532 NFC module.
  2. Trigger the NFC scanning process.
  3. Verify that the PN532 module detects and reads the NFC card.
- Expected Result: PN532 module should successfully detect and read the NFC card.

2. LCD Check:

- Test Objective: Verify the functionality of the LCD display.
- Test Steps:
  1. Display predefined text or message on the LCD.
  2. Verify that the text is displayed correctly on the LCD screen.
- Expected Result: The text/message should be displayed accurately on the LCD screen.

3. Button Check:

- Test Objective: Verify the functionality of the buttons.
- Test Steps:
  1. Press each button (up, down, ok, cancel) individually.
  2. Verify that each button press is detected.
- Expected Result: Each button press should be detected and trigger the corresponding action.

Integration Testing:

4. PN532 with LCD Integration:

- Test Objective: Verify the integration between the PN532 NFC module and the LCD display.
- Test Steps:
  1. Scan a valid NFC card using the PN532 module.
  2. Verify that the user's information retrieved from the NFC card is displayed on the LCD screen.
- Expected Result: User information should be displayed accurately on the LCD screen after scanning the NFC card.

System Testing:

5. Card Registration and Google Sheet Update:

- Test Objective: Verify that the system registers the scanned NFC card and updates the Google Sheet with user information.
- Test Steps:
  1. Scan a valid NFC card.
  2. Verify that the system captures the user information from the NFC card.
  3. Check the Google Sheet to confirm that the user's information is updated.
- Expected Result: User information should be successfully registered and updated on the Google Sheet.

6. Navigation Controls Functionality:

- Test Objective: Verify that the navigation controls (up, down, ok, cancel) work as expected.
- Test Steps:

1. Navigate through the system using each navigation control.
2. Verify that the system responds correctly to each navigation action.

- Expected Result: The system should navigate smoothly between different options and screens using the navigation controls.

Acceptance Testing:

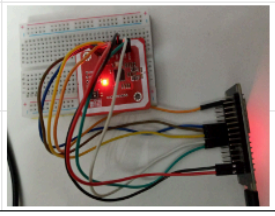
7. Teacher (Customer) Acceptance:

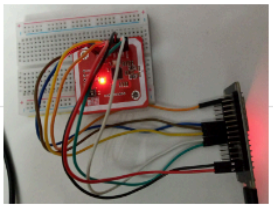
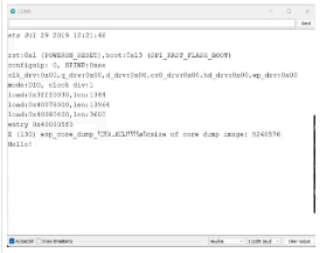
- Test Objective: Verify that the teacher (customer) accepts the system based on its functionality and usability.
- Test Steps:
  1. Demonstrate the system's features and functionalities to the teacher.
  2. Collect feedback from the teacher regarding the system's performance, ease of use, and alignment with their requirements.
- Expected Result: The teacher provides positive feedback and accepts the system for deployment in the educational institution.

These test cases cover various aspects of the NFC-Based Attendance Management System, ensuring that each module, integration, and system functionality meets the specified requirements and expectations.

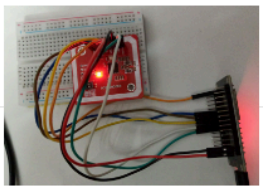
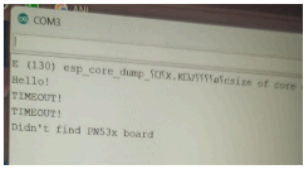
Writing Test Cases

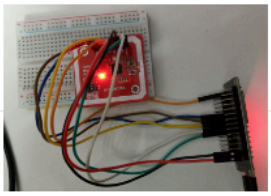
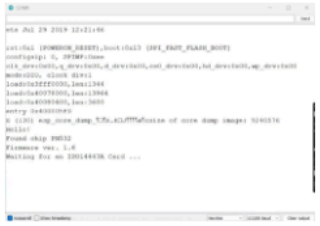
Initializing ESP32 and PN532:

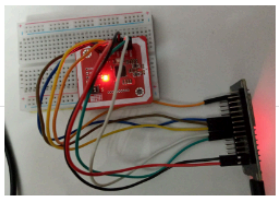
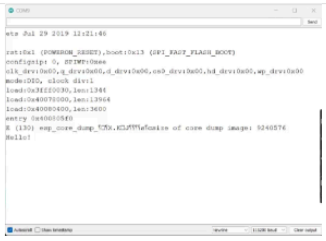
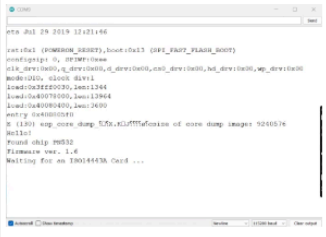
Test Scenario ID	Initialize Device	**Test Case ID**	Initialize_Device -1		
Test Case Description		**Test Priority**	High		
Pre-Requisite	None	**Post-Requisite**	None		
Test Execution Steps: Supplying power to the ESP Board using a Cable					
Sr No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Supply Power	Connect cable to board	Lights ON on ESP32		PASS
2	Supply Power	Connect cable to board	Lights ON on PN532		PASS

Test Scenario ID	Initialize Device	**Test Case ID**	Initialize_Device -2		
Test Case Description		**Test Priority**	High		
Pre-Requisite	None	**Post-Requisite**	None		
Test Execution Steps: Supplying power to the ESP Board using a Cable					
Sr No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Supply Power	Connect cable to board	Lights ON on ESP32		PASS
2	Supply Power	Connect cable to board	Lights ON on PN532		PASS
3	Supply Power	Connect cable to board	ESP32 START		PASS

PN532 Scan Check:

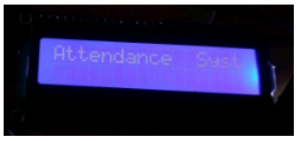
Test Scenario ID	Card Scanning	Test Case ID	Card Scanning -1		
Test Case Description		Test Priority	High		
Pre-Requisite	Power Supply	Post-Requisite	None		
Test Execution Steps: Supplying power to the ESP Board using a Cable					
Sr No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Supply Power	Connect cable to board	Lights ON on ESP32		PASS
2	Supply Power	Connect cable to board	Lights ON on PN532		PASS
3	Supply Power	Connect cable to board	ESP32 START		PASS
4	Supply Power	Connect cable to board	Scanning Initialize		FAIL

Test Scenario ID	Card Scanning	Test Case ID	Card Scanning -2		
Test Case Description		Test Priority	High		
Pre-Requisite	Power Supply	Post-Requisite	None		
Test Execution Steps: Supplying power to the ESP Board using a Cable					
Sr No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Supply Power	Connect cable to board	Lights ON on ESP32		PASS
2	Supply Power	Connect cable to board	Lights ON on PN532		PASS
3	Supply Power	Connect cable to board	ESP32 START		PASS
4	Supply Power	Connect cable to board	Scanning Initialize		PASS

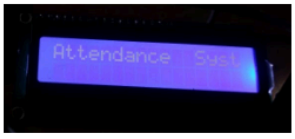
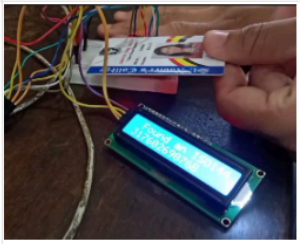
Test Scenario ID	Card Scanning	Test Case ID	Card Scanning -3		
Test Case Description		Test Priority	High		
Pre-Requisite	Power Supply, NFC Card	Post-Requisite	None		
Test Execution Steps: Supplying power to the ESP Board using a Cable and place a NFC card when Firmware detected.					
Sr No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Supply Power	Connect cable to board	Lights ON on ESP32		PASS
2	Supply Power	Connect cable to board	Lights ON on PN532		PASS
3	Supply Power	Connect cable to board	ESP32 START		PASS
4	Supply Power	Connect cable to board	Scanning Initialize		PASS
5	Place NFC card near PN532	Place Card near Scanner	Display Card Data		PASS

LCD Check:

Test Scenario ID	LCD	Test Case ID	LCD -1		
Test Case Description		Test Priority	High		
Pre-Requisite	Power Supply	Post-Requisite	None		
Test Execution Steps: Supplying power to the ESP Board using a Cable.					
Sr No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Supply Power	Connect cable to board	Lights ON of LCD		PASS

Test Scenario ID	LCD	Test Case ID	LCD -2		
Test Case Description		**Test Priority**	Medium		
Pre-Requisite	Power Supply	**Post-Requisite**	None		
Test Execution Steps: Supplying power to the ESP Board using a Cable.					
Sr No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Supply Power	Connect cable to board	"Attendance System"		PASS

PN532 with LCD Integration:

Test Scenario ID	LCD	Test Case ID	LCD -3		
Test Case Description		**Test Priority**	High		
Pre-Requisite	Power Supply and NFC Card	**Post-Requisite**	None		
Test Execution Steps: Supplying power to the ESP Board using a Cable and then place the NFC card on succesful firmware detection.					
Sr No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1	Supply Power	Connect cable to board	"Attendance System"		PASS
2	Place NFC card near PN532	Place Card near Scanner	Display "Found card ...$cardid"		PASS

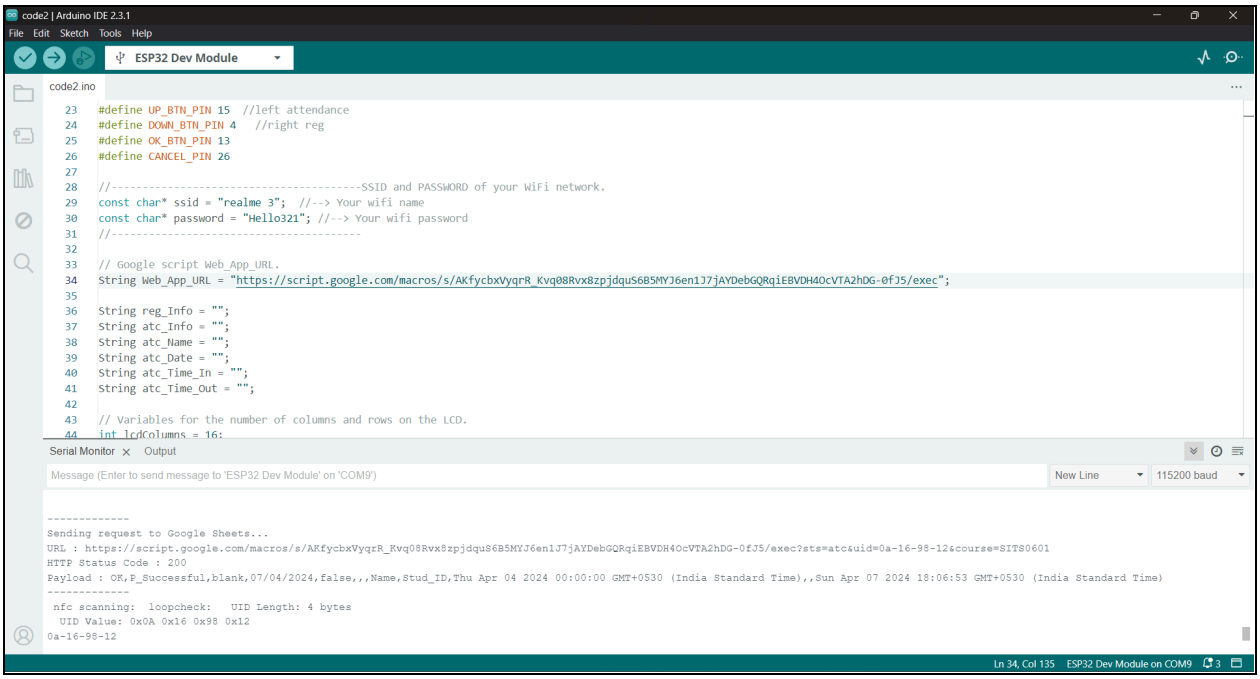
Debugging

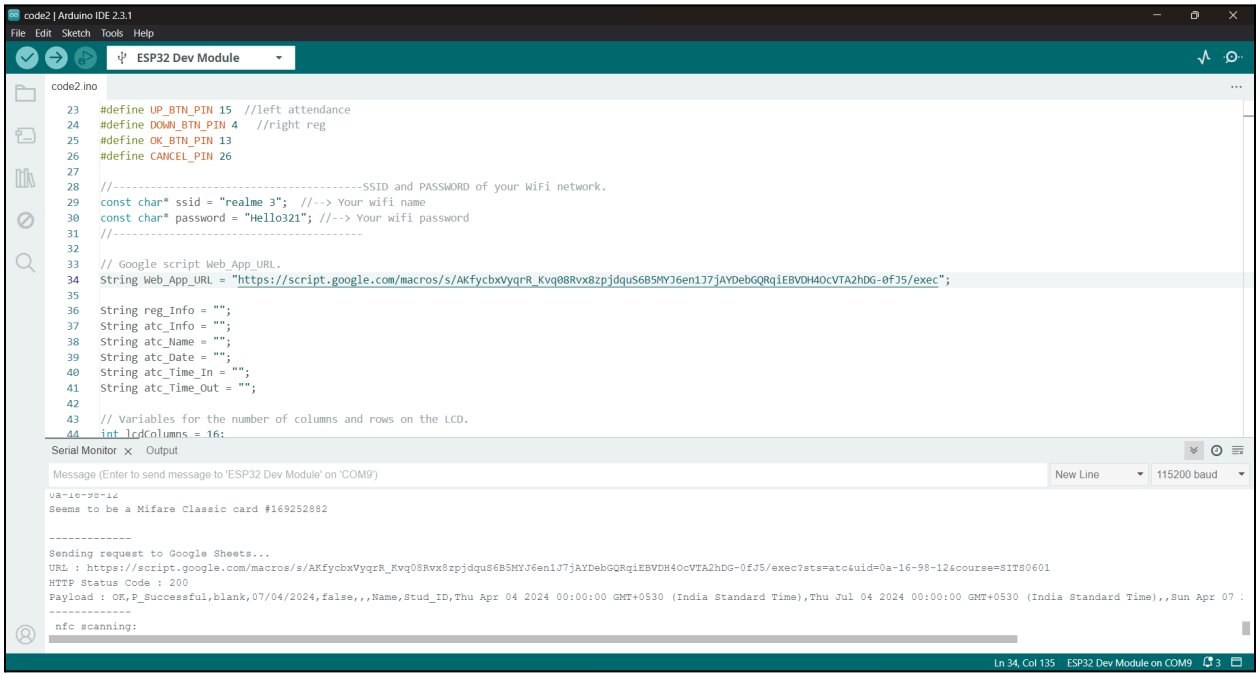
The debugging process for the date issue within our attendance management project was intricate, demonstrating a meticulous approach to solving a complex problem. Initially, we encountered a perplexing issue where the system continuously failed to recognize existing dates within the Google Sheets header, leading to redundant column creation for dates that should have been recognized as present. This was first observed when inputs indicating the current date and its existence within the header row returned unexpected outputs, flagging the current date as absent and erroneously appending new date columns.

Delving into the problem, we identified the root cause as a mismatch in the date formats between the system's output and Google Sheets' date representation. While our system formatted dates in a "DD/MM/YYYY" structure, Google Sheets internally converted and displayed these dates in a full date string format, including the day of the week, month, year, and time zone information. This discrepancy in date formats led to the system's inability to correctly identify existing dates within the spreadsheet, as the comparison between the simplified format and Google Sheets' detailed format always resulted in a mismatch.

To address this issue, we focused on normalizing the date formats to ensure consistent comparisons. This involved modifying the system to recognize and work with Google Sheets' internal date representation. We implemented a solution where the current date, initially formatted in "DD/MM/YYYY," was programmatically converted to match Google Sheets' comprehensive date string format. This conversion process took into account the correct parsing of the current date into a format that included the necessary details for an accurate comparison with the header row values.

Through this methodical debugging process, we managed to resolve the date recognition issue, eliminating the creation of unnecessary columns and ensuring that the system accurately identified and worked with existing dates. This experience not only underscored the importance of understanding the underlying data representations and formats used by external systems (in this case, Google Sheets) but also highlighted the critical nature of flexible and adaptive problem-solving strategies in software development. The resolution of this issue was a testament to the diligent and analytical approach taken by our team, showcasing our commitment to delivering a robust and efficient attendance management system.


code2 | Arduino IDE 2.3.1
File Edit Sketch Tools Help
ESP32 Dev Module
code2.ino
#define UP_BTN_PIN 15 //left attendance
#define DOWN_BTN_PIN 4 //right reg
#define OK_BTN_PIN 13
#define CANCEL_PIN 26
//----------SSID and PASSWORD of your WiFi network.
const char* ssid = "realme 3"; //--> Your wifi name
const char* password = "Hello321"; //--> Your wifi password
//----------
// Google script Web_App_URL.
String Web_App_URL = "https://script.google.com/macros/s/AKfycbxVyqrR_Kvq08Rvx8zpjdquS6B5MYJ6en1J7jAYDebGQRqiEBVDH4OcVTA2hDG-0fJ5/exec";
String reg_Info = "";
String atc_Info = "";
String atc_Name = "";
String atc_Date = "";
String atc_Time_In = "";
String atc_Time_Out = "";
// Variables for the number of columns and rows on the LCD.
int lcdColumns = 16;
Serial Monitor
Output
Message (Enter to send message to 'ESP32 Dev Module' on 'COM9')
New Line
115200 baud
Sending request to Google Sheets...
URL : https://script.google.com/macros/s/AKfycbxVyqrR_Kvq08Rvx8zpjdquS6B5MYJ6en1J7jAYDebGQRqiEBVDH4OcVTA2hDG-0fJ5/exec?sts=atc&uid=0a-16-98-12&course=SITS0601
HTTP Status Code : 200
Payload : OK,P_Successful,blank,07/04/2024,false,,,Name,Stud_ID,Thu Apr 04 2024 00:00:00 GMT+0530 (India Standard Time),,Sun Apr 07 2024 18:06:53 GMT+0530 (India Standard Time)
nfc scanning: loopcheck: UID Length: 4 bytes
UID Value: 0x0A 0x16 0x98 0x12
0a-16-98-12
Ln 34, Col 135 ESP32 Dev Module on COM9


code2 | Arduino IDE 2.3.1
File Edit Sketch Tools Help
ESP32 Dev Module
code2.ino
#define UP_BTN_PIN 15 //left attendance
#define DOWN_BTN_PIN 4 //right reg
#define OK_BTN_PIN 13
#define CANCEL_PIN 26
//----------SSID and PASSWORD of your WiFi network.
const char* ssid = "realme 3"; //--> Your wifi name
const char* password = "Hello321"; //--> Your wifi password
//----------
// Google script Web_App_URL.
String Web_App_URL = "https://script.google.com/macros/s/AKfycbxVyqrR_Kvq08Rvx8zpjdquS6B5MYJ6en1J7jAYDebGQRqiEBVDH4OcVTA2hDG-0fJ5/exec";
String reg_Info = "";
String atc_Info = "";
String atc_Name = "";
String atc_Date = "";
String atc_Time_In = "";
String atc_Time_Out = "";
// Variables for the number of columns and rows on the LCD.
int lcdColumns = 16;
Serial Monitor
Output
Message (Enter to send message to 'ESP32 Dev Module' on 'COM9')
New Line
115200 baud
0a-16-98-12
Seems to be a Mifare Classic card #169252882
Sending request to Google Sheets...
URL : https://script.google.com/macros/s/AKfycbxVyqrR_Kvq08Rvx8zpjdquS6B5MYJ6en1J7jAYDebGQRqiEBVDH4OcVTA2hDG-0fJ5/exec?sts=atc&uid=0a-16-98-12&course=SITS0601
HTTP Status Code : 200
Payload : OK,P_Successful,blank,07/04/2024,false,,,Name,Stud_ID,Thu Apr 04 2024 00:00:00 GMT+0530 (India Standard Time),Thu Jul 04 2024 00:00:00 GMT+0530 (India Standard Time),,Sun Apr 07
nfc scanning:
Ln 34, Col 135 ESP32 Dev Module on COM9

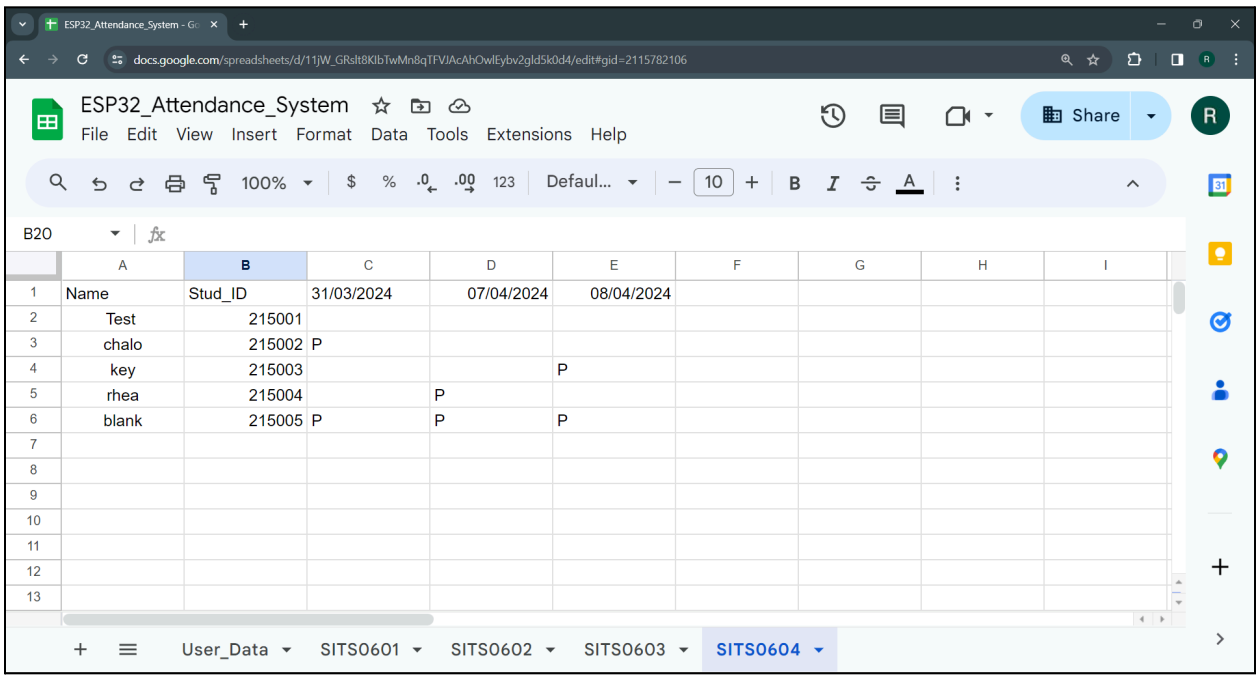
INITIALLY :

	A	B	C	D	E	F	G	H	I	J	K
1	Name	Stud_ID	02/04/2024	02/04/2024	02/04/2024	02/04/2024	02/04/2024	02/04/2024	02/04/2024		
2	Test	215001									
3	chalo	215002	P								
4	key	215003			P		P		P		
5	rhea	215004									
6	blank	215005		P		P		P			
7											
8											
9											
10											
11											
12											
13											

A1 Name

	A	B	C	D	E	F	G	H
1	Name	Stud_ID	31/03/2024	02/04/2024	02/04/2024	02/04/2024		
2	Test	215001						
3	chalo	215002			P	P		
4	key	215003		P				
5	rhea	215004	P					
6	blank	215005	P					
7								
8								
9								
10								

AFTER DEBUGGING :



Error Handling

Error handling in the provided code is implemented through various mechanisms such as conditional statements, return values, and exception handling. Let's break down how error handling is addressed in the code:

Conditional Statements: Throughout the code, conditional statements are used to check for specific conditions or states and execute corresponding actions. For example:

a. if (WiFi.status() == WL_CONNECTED): This condition checks if the ESP32 is successfully connected to the Wi-Fi network before proceeding with HTTP requests.

b. if (success): This condition checks if the NFC reading operation is successful before further processing the obtained UID.

Return Values: Functions in the code often return values indicating the success or failure of certain operations. For instance:

a. The getUID() function returns 1 if the NFC reading operation is successful (SUCCESS) and 0 if it fails (FAILURE).

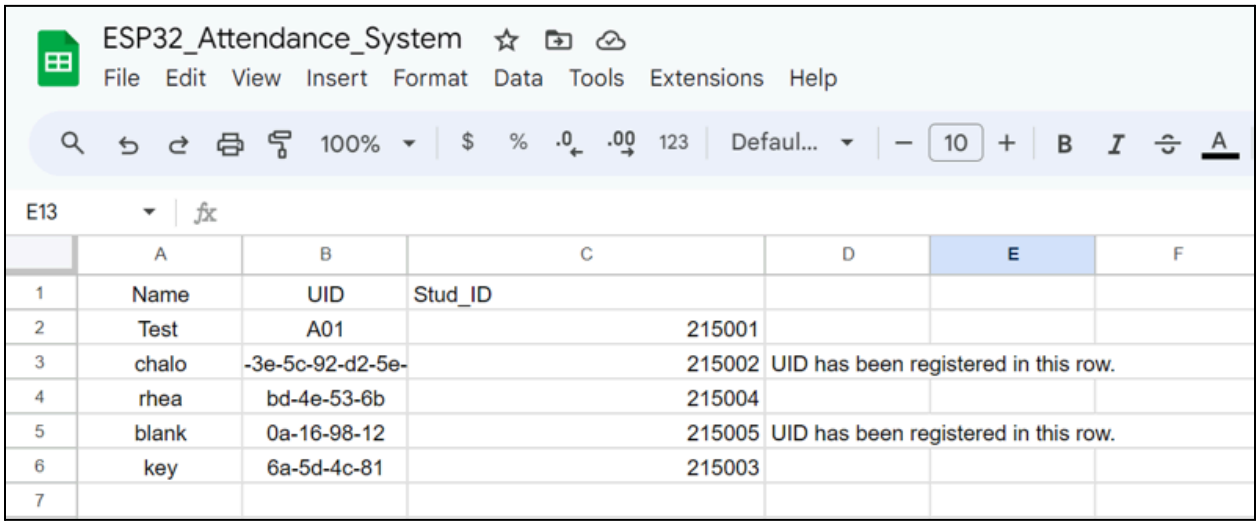
b. The http_Req() function returns different response statuses such as "OK", "R_Successful", or error messages like "atcErr01" based on the outcome of HTTP requests to Google Sheets.

Exception Handling: While not explicitly implemented in the provided code, error handling can be enhanced with exception handling mechanisms such as try-catch blocks in languages like C++ or Python. Exception handling allows for more robust error detection and recovery by catching and handling exceptions that occur during runtime.

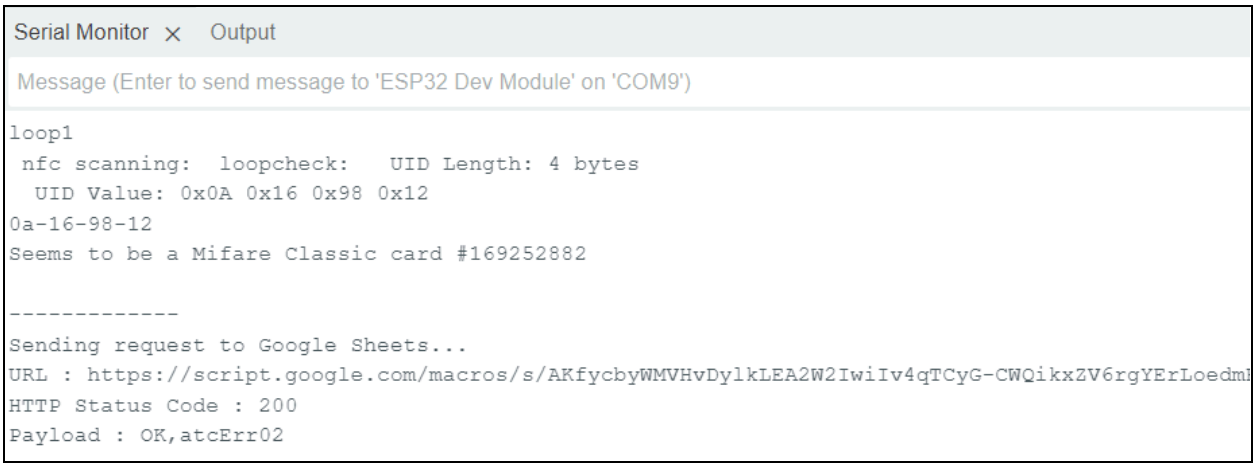
Feedback and Indicators: The LCD display is utilized to provide feedback to users regarding the system's status and any encountered errors. Messages are displayed on the LCD screen to inform users of successful operations, errors, or waiting states, enhancing user experience and aiding in troubleshooting.

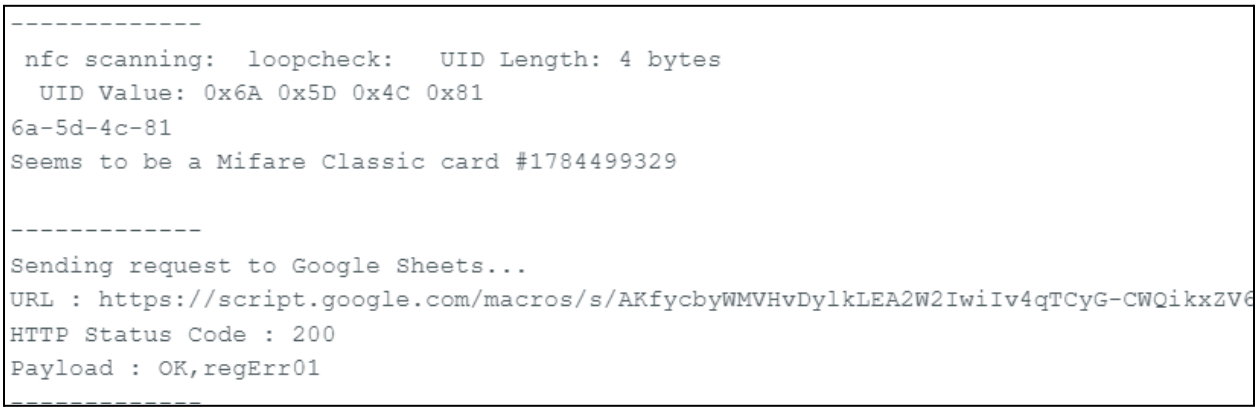
Logging and Serial Output: The Serial.println() statements are used for debugging purposes, providing real-time feedback via the serial monitor. By printing relevant information such as HTTP status codes, payload data, and debug messages, developers can diagnose issues and monitor the system's behavior during runtime.

Overall, error handling in the code involves a combination of conditional checks, return values, and feedback mechanisms to detect, report, and manage errors effectively, ensuring the robustness and reliability of the NFC-based attendance tracker.

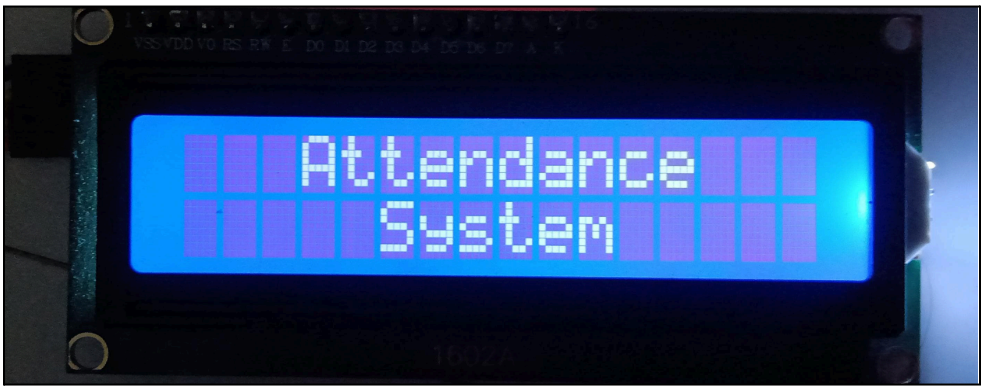


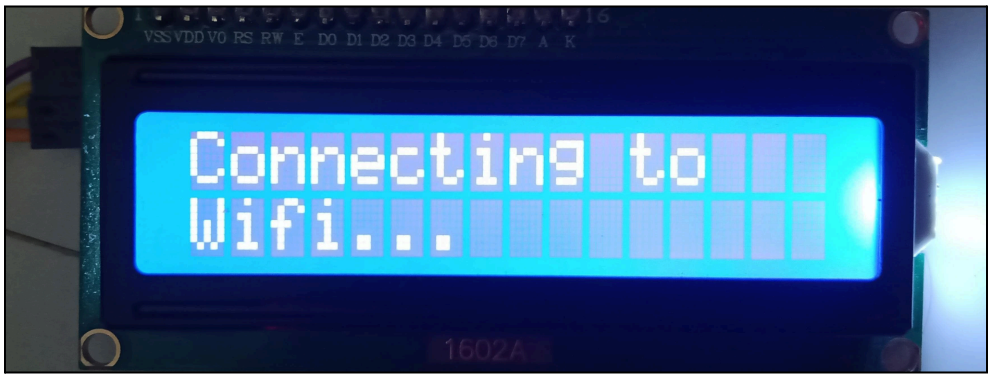
	A	B	C	D	E	F
1	Name	UID	Stud_ID			
2	Test	A01	215001			
3	chalo	-3e-5c-92-d2-5e-	215002	UID has been registered in this row.		
4	rhea	bd-4e-53-6b	215004			
5	blank	0a-16-98-12	215005	UID has been registered in this row.		
6	key	6a-5d-4c-81	215003			
7						


Serial Monitor
Output
Message (Enter to send message to 'ESP32 Dev Module' on 'COM9')
loop1
nfc scanning: loopcheck: UID Length: 4 bytes
UID Value: 0x0A 0x16 0x98 0x12
0a-16-98-12
Seems to be a Mifare Classic card #169252882
Sending request to Google Sheets...
URL : https://script.google.com/macros/s/AKfycbyWMVHvDylkLEA2W2IwiIv4qTCyG-CWQikxZV6rgYErLoedm
HTTP Status Code : 200
Payload : OK,atcErr02


nfc scanning: loopcheck: UID Length: 4 bytes
UID Value: 0x6A 0x5D 0x4C 0x81
6a-5d-4c-81
Seems to be a Mifare Classic card #1784499329
Sending request to Google Sheets...
URL : https://script.google.com/macros/s/AKfycbyWMVHvDylkLEA2W2IwiIv4qTCyG-CWQikxZV6
HTTP Status Code : 200
Payload : OK,regErr01

Snapshots


Attendance
System

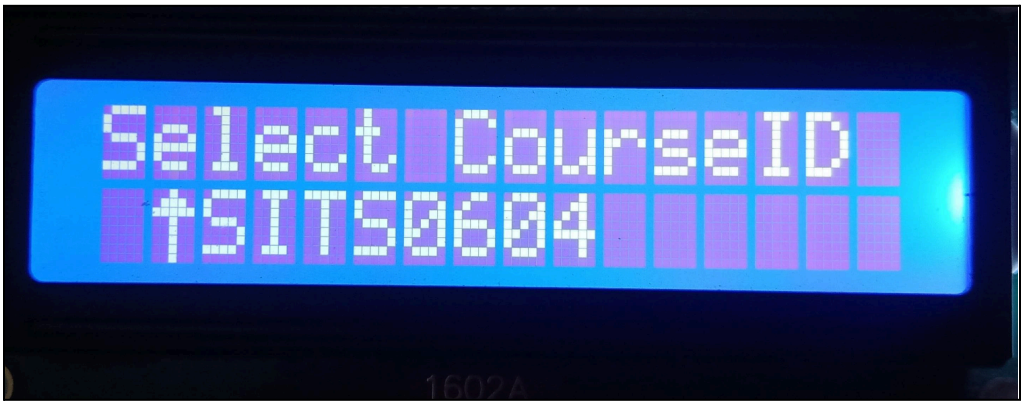

Connecting to
Wifi...


WiFi connected


Select mode
↑-REG
ATC-↓

ATC Mode:


Select CourseID
↑SIT50603↓


Select CourseID
↑SIT50604
1602A


Selected Course:
SIT50604 CARD


Getting UID
Please wait...
1602A


blank
DONE


SITS0604 DONE
TAP CARD 1


Getting UID
Please wait...
1602A


rhea
DONE


SITS0604 DONE
TAP CARD 2

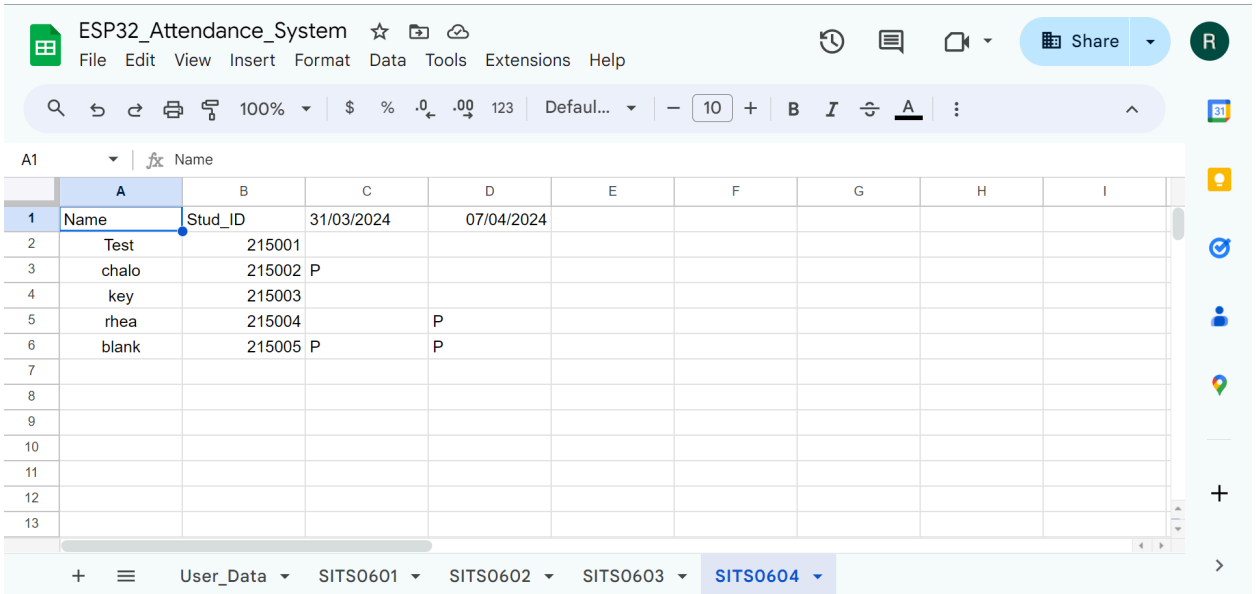
NFC init and Wifi Connection Serial Monitor:

```
Found chip PN532
Firmware ver. 1.6
838927879
-------------
WIFI mode : STA
-------------


------------
Connecting to realme 3
..
WiFi connected
------------
```

ATC Success:

```
 nfc scanning:  loopcheck:   UID Length: 4 bytes
  UID Value: 0x6A 0x5D 0x4C 0x81
6a-5d-4c-81
Seems to be a Mifare Classic card #1784499329

------------
Sending request to Google Sheets...
URL : https://script.google.com/macros/s/AKfycbyWMVHvDylkLEA2W2IwiIv4qTCyG-CWQikxZV6rgYErLoedmRMOIAo9
HTTP Status Code : 200
Payload : OK,P_Successful,key,07/04/2024,true,,,Name,Stud_ID,31/03/2024,07/04/2024
------------
```

Google Sheets:



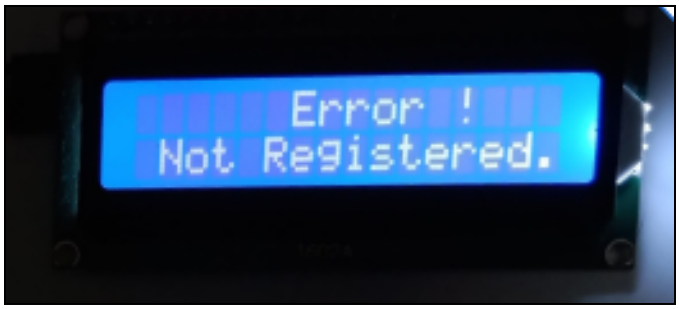
Rescanning Marked Card:

Serial Monitor × Output

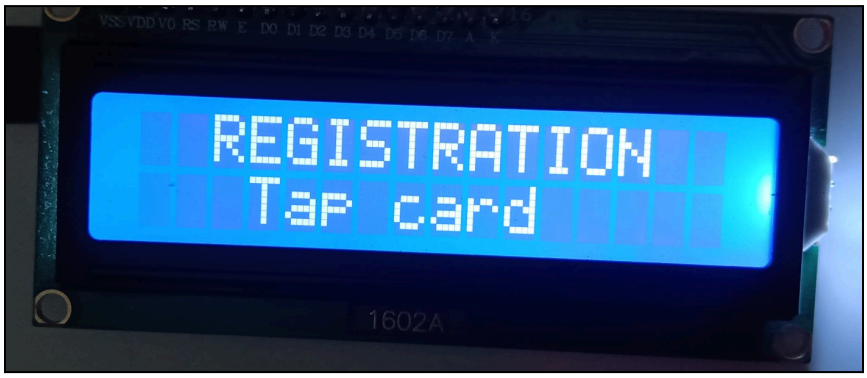
Message (Enter to send message to 'ESP32 Dev Module' on 'COM9')

```
loop1
 nfc scanning:  loopcheck:   UID Length: 4 bytes
  UID Value: 0x0A 0x16 0x98 0x12
0a-16-98-12
Seems to be a Mifare Classic card #169252882

-------------
Sending request to Google Sheets...
URL : https://script.google.com/macros/s/AKfycbyWMVHvDylkLEA2W2IwiIv4qTCyG-CWQikxZV6rgYErLoedml
HTTP Status Code : 200
Payload : OK,atcErr02
```

Scanning unregistered student:

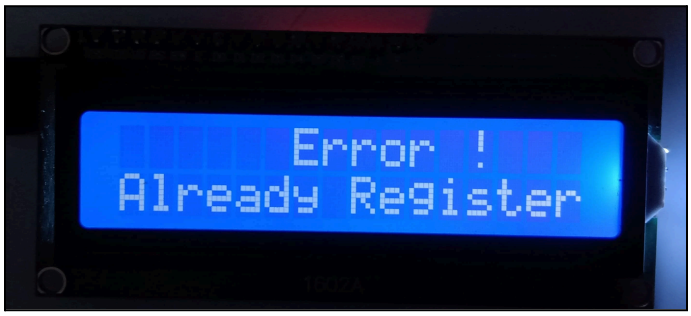


REG Mode:



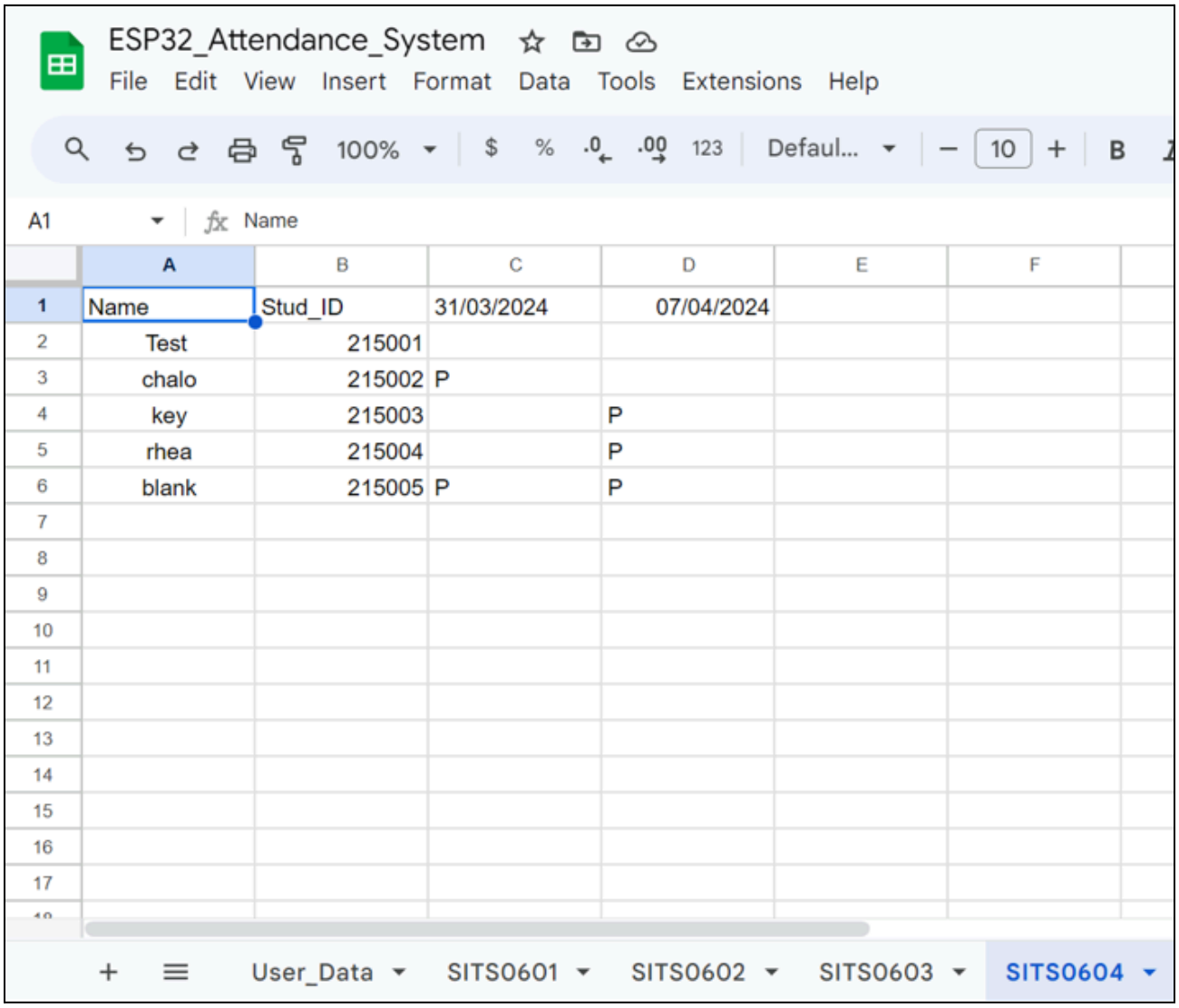

```
-------------
loop reg
 nfc scanning:  loopcheck:   UID Length: 4 bytes
  UID Value: 0x6A 0x5D 0x4C 0x81
6a-5d-4c-81
Seems to be a Mifare Classic card #1784499329

-------------
Sending request to Google Sheets...
URL : https://script.google.com/macros/s/AKfycbyWMVHvDylkLEA2W2IwiIv4qTCyG-CWQikxZV6rgYErLoedmRM
HTTP Status Code : 200
Payload : OK,R_Successful
```

Already Registered Error:

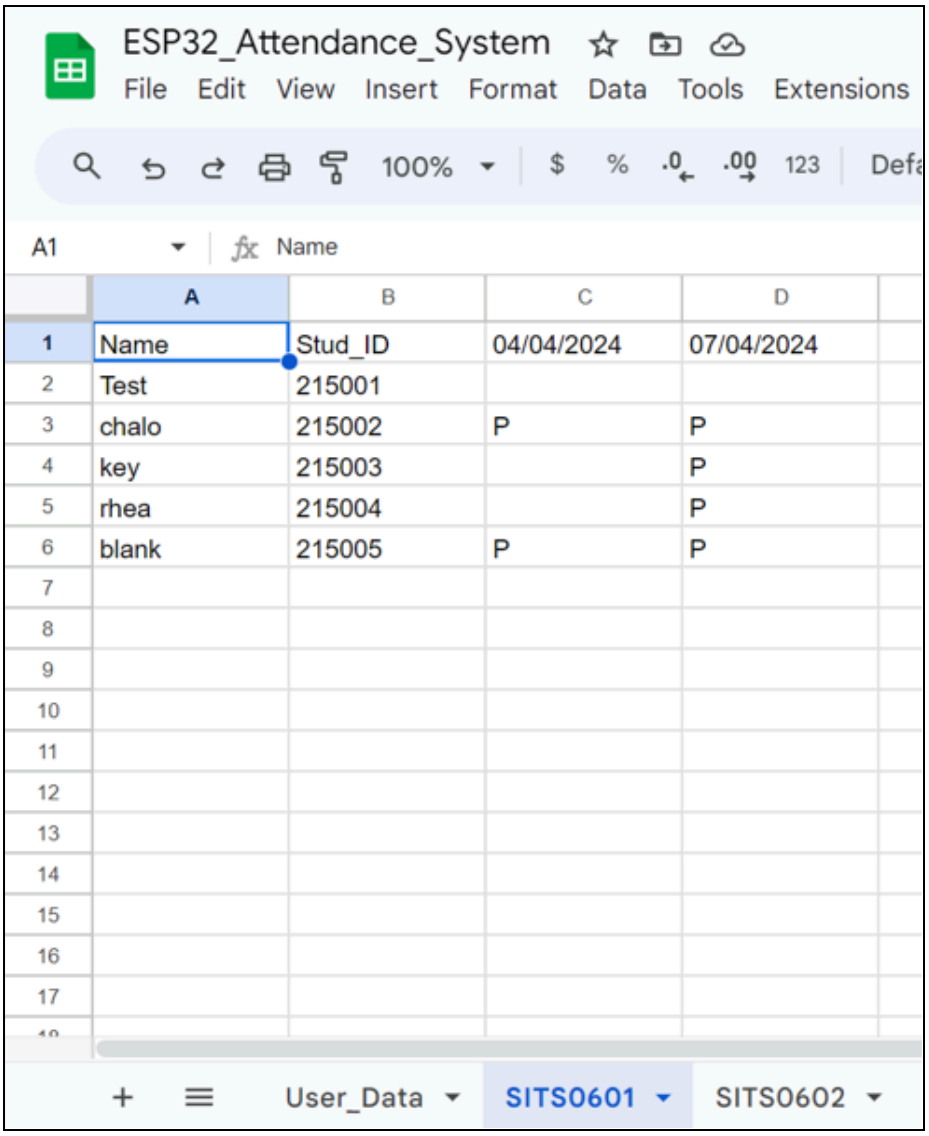



```
-------------
 nfc scanning:  loopcheck:   UID Length: 4 bytes
  UID Value: 0x6A 0x5D 0x4C 0x81
6a-5d-4c-81
Seems to be a Mifare Classic card #1784499329

-------------
Sending request to Google Sheets...
URL : https://script.google.com/macros/s/AKfycbyWMVHvDylkLEA2W2IwiIv4qTCyG-CWQikxZV6
HTTP Status Code : 200
Payload : OK,regErr01
-------------
```

Output Sheets



	A	B	C	D	E	F
1	Name	Stud_ID	31/03/2024	07/04/2024		
2	Test	215001				
3	chalo	215002	P			
4	key	215003		P		
5	rhea	215004		P		
6	blank	215005	P	P		
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						



Cancel Button (BREAK) :

```
-------------
Sending request to Google Sheets...
URL : https://script.google.com/macros/s/AKfycbyWMVHvDylkLEA2W2IwiIv4qTCyG-C
HTTP Status Code : 200
Payload : OK,regErr01
-------------
 nfc scanning: BREAK
```

Limitations

Lack of prevention against stacking of IDs:

1. The system does not have mechanisms to detect and prevent the simultaneous tapping of multiple IDs.
2. This allows individuals to exploit the system by tapping multiple IDs at once, resulting in inaccurate attendance records.
3. As a result, the integrity and accuracy of the attendance data may be compromised.

Absence of proper authentication:

1. The system lacks robust methods for verifying the identity of the person tapping the ID.
2. Without proper authentication measures such as PIN codes, biometric scans, or two-factor authentication, there is a risk of unauthorized access and misuse of ID cards.
3. This loophole can be exploited by students or teachers to impersonate others and falsely mark their attendance.

Vulnerability to cheating and impersonation:

1. Due to the aforementioned limitations, the system is vulnerable to various forms of cheating and impersonation.
2. Students or teachers can manipulate the system to record attendance for absent individuals or to inflate their own attendance records.
3. This compromises the reliability and credibility of the attendance tracking process, affecting the overall integrity of academic records.

Enhancements

Integration of face or fingerprint authentication:

1. Enhance identity verification by integrating biometric authentication methods such as facial recognition or fingerprint scanning.
2. This ensures that the person tapping the ID is indeed the authorized individual associated with that ID, thereby reducing the risk of identity misuse and fraud.

Implementation of advanced security measures:

1. Strengthen the system's security by implementing encryption protocols, secure communication channels, and access controls.
2. Employ techniques such as tokenization or digital signatures to ensure the integrity and confidentiality of attendance data.
3. This enhances data protection and mitigates the risk of unauthorized access or tampering with attendance records.

Bibliography

- **https://warlord0blog.wordpress.com/2021/10/09/esp32-and-nfc-over-i2c/**
- **https://youtu.be/2qf6gIqhWNA?si=VK-cWvI46Pvwgnun**
- **https://www.youtube.com/watch?v=h-jqF8Y5iV4**
- **https://randomnerdtutorials.com/esp32-pinout-reference-gpios/**
- **https://forum.arduino.cc/t/a-fatal-error-occurred-failed-to-connect-to-esp32-no-serial-data-received/1150133/2**
- **https://randomnerdtutorials.com/learn-esp32-with-arduino-ide/#sign-up**
- **https://lastminuteengineers.com/esp32-i2c-lcd-tutorial/**
- **https://forum.arduino.cc/t/esp32-compilation-error/948350/4**
- **https://techexplorations.com/guides/esp32/begin/power/**