Visvesvaraya National Institute of Technology (VNIT), Nagpur

Embedded Systems (ECP403)

End-Semester Examination

Submitted by :
Prajyot Jadhav (BT20ECE046)
Semester 5

Submitted to :
Dr. Ankit A. Bhurane
(Course Instructor)
Department of Electronics and Communication Engineering,
VNIT Nagpur

End-Semester Examination

Problem Statement:

- An ATM machine system to be implemented using ESP32.

Arduino code with comments:

```
// Including the required libraries
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>
#include <HTTPClient.h>
#include "time.h"

// Network credentials for the WiFi to which the ESP shoul be
 ↪  connected
const char* ssid = "DESKTOP-RD2660R 3053"; //ssid of the WiFI
 ↪  network
const char* password = "1565+qD7"; //password of the Wifi network

// Initialize Telegram BOT
#define BOTtoken "5966245620:AAFDir2fvXxtVgKTjbBuHTAZfuQn7pfRg6w"
 ↪  // Bot Token for my bot BT20ECE046_ATMBot obtained from
 ↪  Botfather
// The chat id of the Telegram account being used to access the
 ↪  bot
#define CHAT_ID "1418xxxxxx";


WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client); //The Universal
 ↪  Telegram Bot library is used to access the bot

// The messages sent are checked for every one second
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;

//Global Variables text and prev_text containing the latest and
 ↪  second last message of the user
```

```
String text = ""; // contains the latest message of the user
String prev_text = ""; // contains the second last message of the
 ↪  user

// Flags
bool FlagU = false; //This flag is setted while taking username
 ↪  input from the user
bool FlagP = false; //This flag is setted when the user enters the
 ↪  password
bool FlagNU = false; // Flag for going into third loop for
 ↪  New_User
bool FlagFT = false; //Flag for accessing third loop of Fund
 ↪  Transfer
bool Logged_in = false; // Flag to check whether the user is Logged
 ↪  in

// Variables used in Arduino
String user; // variable used to store the username, during login
 ↪  once it is verified
String PassPIN; // this variable stores the password during
 ↪  verification and once the user is logged in

String funds; // variable used to store the amount during fund
 ↪  transfer
String reciever; // variable which is used to store the name of
 ↪  user to whom the funds are transferred

// Variables used in the function New_User
String newuser; // variable to store the username entered for
 ↪  creating a new user
String newuserpass; // this variable stores the password entered
 ↪  during creation of a new account

// Variables used for sending to GoogleScript
String namelogin;
String pinlogin;

String GOOGLE_SCRIPT_ID =
 ↪  "AKfycbxx9X5f0GwE9VFjX_QFs_JFC7qlBHYxbQmTsgHYICgesHXbSSmjoX_dFNB
N7dD3Ucx9";    // The ID of Deployment for the sheet BT20ECE046
```

```
// Commands used in the ATM machine

// verification of username and password
int verification(String Name,String Pass){ //This function is
 ↪  called to verify the entered username and password

  Serial.println("Verification started: " + Name +" " + Pass); //
   ↪  Print that verification has started on the Serial Monitor

  String url_write = "https://script.google.com/macros/s/" +
   ↪  GOOGLE_SCRIPT_ID +
   ↪  "/exec?namelogin="+String(user)+"&pinlogin="+String(PassPIN);
   ↪  // the unique variables namelogin and pinlogin used to send
   ↪  values to google script

  HTTPClient http;
  http.begin(url_write.c_str());
  http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
  int conf = http.GET();
  String result = String(http.getString()); // The output from the
   ↪  google script is stored in this variable

  Serial.println(result); // the result from scripts is printed on
   ↪  serial monitor

  if(result == "0"){
    return 0; // Zero returned when password matches
    Serial.println("The password MATCHED!");
  }
  if(result == "1"){
    return 1; // One returned when the password doesn't match
    Serial.println("The password DOESN'T match!");
  }
  if(result == "2"){
    return 2; // Two is returned when the entered username is not
     ↪  present in the database
    Serial.println("User doesn't exist");
  }
}
```

```
// This function is used to handle when we recieve messages from
 ↪  the user on the Telegram bot
void handleNewMessages(int numNewMessages) {
  Serial.println("handleNewMessages");
  Serial.println(String(numNewMessages));

  for (int i=0; i<numNewMessages; i++) {
    // Chat id of the telegram account which interacts with the
     ↪  user
    String chat_id = String(bot.messages[i].chat_id);

    text = bot.messages[i].text; //The latest message of the user
     ↪  is stored in this variable
    Serial.println(text); // it is printed on Serial Monitor

    String from_name = bot.messages[i].from_name; // The Username
     ↪  of the

  // Start Function: Prompts the user to various available
   ↪  functions
  // This if block executes when the last message of user was
   ↪  start
    if (text == "/start") {
    // Welcome message for the user and prompt to login or create
     ↪  a new user
      String msg = "Welcome, " + from_name + ".\n";
      msg += "You can control me using the following commands.\n\n";
      msg += "/Login - to login using existing username and password
       ↪  \n\n";
      msg += "/New_User - to create a new account with a balance of
       ↪  Rs. 15,000";

      String keyboardJson = "[[\"/Login\",
       ↪  \"/New_User\"],[\"/Exit\"]]"; // keyboard used to make
       ↪  user experience more interactive

      bot.sendMessageWithReplyKeyboard(chat_id, msg, "",
       ↪  keyboardJson, true); // Bot sends this message along with
       ↪  a reply keyboard
```

```
}

else if (text == "/Login"){ // This block executed when the
↪  current message is Login
  String keyboardJson = "[[\"/Exit\"]]"; // keyboard used to
  ↪  make user experience more interactive
  bot.sendMessageWithReplyKeyboard(chat_id, "Enter username: ",
  ↪  "", keyboardJson, true); // Bot sends this message along
  ↪  with a reply keyboard
  FlagU = true; // Flag corresponding to username set true
  return;
}

else if (text == "/Enter_PIN"){ // executed when last user
↪  message is enter pin
  String keyboardJson = "[[\"/Exit\"]]"; // keyboard used to
  ↪  make user experience more interactive
  bot.sendMessageWithReplyKeyboard(chat_id, "Enter the password:
  ↪  ", "", keyboardJson, true); // Bot sends this message
  ↪  along with a reply keyboard
  FlagP = true; // Flag corresponding to password set true
  return;
}

// Code for the command new user
else if (text == "/New_User"){ // block executed when last user
↪  messsage is new user
  String keyboardJson = "[[\"/Exit\"]]"; // keyboard used to
  ↪  make user experience more interactive
  bot.sendMessageWithReplyKeyboard(chat_id, "Enter the new
  ↪  username: ", "", keyboardJson, true); // Bot sends this
  ↪  message along with a reply keyboard
}
else if (prev_text == "/New_User"){ // executed when second
↪  last message is new user
  newuser = text; // the latest message is the username
  FlagNU = true; // flag corresponding to new user set true
  String keyboardJson = "[[\"/Exit\"]]"; // keyboard used to
  ↪  make user experience more interactive
```

```
    bot.sendMessageWithReplyKeyboard(chat_id, "Enter the password
     ↪  for this username: ", "", keyboardJson, true); // Bot
     ↪  sends this message along with a reply keyboard
  }
  else if (FlagNU == true){ //executed after the previous loop
   ↪  when the flag is true
    FlagNU = false; // flag set to false
    newuserpass = text; // the latest text is new password
    Serial.println("Creating new user : " + newuser+ " "
     ↪  +newuserpass);
    String url_write = "https://script.google.com/macros/s/" +
     ↪  GOOGLE_SCRIPT_ID + "/exec?usernew="+ String(newuser)
     ↪  +"&usernewpass="+String(newuserpass); // data sent to
     ↪  script using unique variables

    HTTPClient http;
    http.begin(url_write.c_str());
    http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
    int conf = http.GET();
    String result = String(http.getString()); // return string
     ↪  from script stored in result

    if (result == "0"){ // Username already exists
      newuser = ""; // global variable is set to empty
      newuserpass = ""; // global variable is set to empty
      String keyboardJson = "[[\"/Login\",
       ↪  \"/New_User\"],[\"/Exit\"]]"; // keyboard used to make
       ↪  user experience more interactive
      bot.sendMessageWithReplyKeyboard(chat_id, "The entered
       ↪  username already exists. Please select one of the
       ↪  following actions ", "", keyboardJson, true); // Bot
       ↪  sends this message along with a reply keyboard
    }
    else if (result == "1"){ // Successfully created new user
      newuser = ""; // global variable is set to empty
      newuserpass = ""; // global variable is set to empty
      String keyboardJson = "[[\"/Login\",
       ↪  \"/New_User\"],[\"/Exit\"]]"; // keyboard used to make
       ↪  user experience more interactive
```

```
      bot.sendMessageWithReplyKeyboard(chat_id, "New user created.
       ↪  Please login to continue. ", "", keyboardJson, true); //
       ↪  Bot sends this message along with a reply keyboard
    }
    // all the global variables are set to empty and flags are
     ↪  set to false
    user = "";
    PassPIN = "";
    text = "";
    prev_text = "";
    FlagU = false;
    FlagP = false;
    Logged_in = false;
  }
  // this is the code for password change i.e. password reset
   ↪  after login
  else if (text == "/PIN_Change"){ // executed when the latest
   ↪  message is pin change
    if (Logged_in == true){ // checks whether the user is logged
     ↪  in
      String keyboardJson = "[[\"/Exit\"]]"; // keyboard used to
       ↪  make user experience more interactive
      bot.sendMessageWithReplyKeyboard(chat_id, "Enter the new
       ↪  password:", "", keyboardJson, true); // Bot sends this
       ↪  message along with a reply keyboard
    }
    else if (Logged_in == false){ //asks the user to log in
      bot.sendMessage(chat_id, "The user is not Logged in", "");
    }
  }
  else if (prev_text == "/PIN_Change"){ //executed for second
   ↪  last text as pin change
    String newpass = text; // the latest text is new password
    Serial.println("Password change started : " + newpass+ " "
     ↪  +user); // Process has started, printed on serial
     ↪  monitor
```

```
    String url_write = "https://script.google.com/macros/s/" +
    ↪  GOOGLE_SCRIPT_ID + "/exec?passreset="+ String(newpass)
    ↪  +"&userid1="+String(user); // sending data to google
    ↪  script using unique variables

    HTTPClient http;
    http.begin(url_write.c_str());
    http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
    int conf = http.GET();
    String result = String(http.getString()); // value returned
    ↪  from script stored in this variable

    if (result == "0"){ // Successful reset
      String keyboardJson = "[[\"/Login\"],[\"/Exit\"]]"; //
      ↪  keyboard used to make user experience more interactive
      bot.sendMessageWithReplyKeyboard(chat_id, "The password
      ↪  reset successfully. Please Login again. ", "",
      ↪  keyboardJson, true); // Bot sends this message along
      ↪  with a reply keyboard
    }
  }

  //This is code for Withdrawal of money from user's account
  else if (text == "/Debit"){ //executed for latest message
  ↪  debit

    if(Logged_in == true){ // Checks if the user is logged in
      String keyboardJson = "[[\"/Exit\"]]"; // keyboard used to
      ↪  make user experience more interactive
      bot.sendMessageWithReplyKeyboard(chat_id, "Enter the amount
      ↪  to be debited: ", "", keyboardJson, true); // Bot sends
      ↪  this message along with a reply keyboard
    }
    else if (Logged_in == false) {
      bot.sendMessage(chat_id, "The user is not Logged in", "");
    }

  }
  else if (prev_text == "/Debit"){ // executed for second last
  ↪  text debit
```

```
    String amount = text; // the amount to be debited
    Serial.println("Withdrawal started : " + amount+ " " +user);
    ↪ // Serial monitor print that function has started

    String url_write = "https://script.google.com/macros/s/" +
    ↪ GOOGLE_SCRIPT_ID + "/exec?withdraw="+ String(amount)
    ↪ +"&userid="+String(user); // send data to google script
    ↪ using unique variables

    HTTPClient http;
    http.begin(url_write.c_str());
    http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
    int conf = http.GET();
    String result = String(http.getString()); // String returned
    ↪ from the script is stored here

    if (result == "1"){ // Successful debit
      String keyboardJson = "[[\"/Debit\",
      ↪ \"/Credit\"],[\"/Balance_Inquiry\",\"/PIN_Change\"],
      [\"/Fund_Transfer\",\"/Exit\"]]"; // keyboard used to make
      ↪ user experience more interactive
      bot.sendMessageWithReplyKeyboard(chat_id, "Rs. "+amount+"
      ↪ debited successfully.", "", keyboardJson, true); // Bot
      ↪ sends this message along with a reply keyboard
    }
    else if (result == "0"){ // balance is less
      String keyboardJson = "[[\"/Debit\",
      ↪ \"/Credit\"],[\"/Balance_Inquiry\",\"/PIN_Change\"],
      [\"/Fund_Transfer\",\"/Exit\"]]"; // keyboard used to make
      ↪ user experience more interactive
      bot.sendMessageWithReplyKeyboard(chat_id, "Insufficient
      ↪ Balance.", "", keyboardJson, true); // Bot sends this
      ↪ message along with a reply keyboard
    }
  }
  //Code For Balance Inquiry
  else if (text == "/Balance_Inquiry"){ // checking the user
  ↪ balance in his account
    Serial.println("Balance Inquiry started : " +user);
```

```
    String url_write = "https://script.google.com/macros/s/" +
    ↪  GOOGLE_SCRIPT_ID + "/exec?balancecheckuser="+
    ↪  String(user); // send data to sheet using unique id

    HTTPClient http;
    http.begin(url_write.c_str());
    http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
    int conf = http.GET();
    String result = String(http.getString()); // String retured
    ↪  from script is stored here

    bot.sendMessage(chat_id, "The total balance of the user " +
    ↪  user + " is Rs. " + result, "");
  }
  // This is the code for Credit
  else if (text == "/Credit"){ //executed when the latest message
  ↪  is text
    if(Logged_in == true){
      bot.sendMessage(chat_id, "Enter amount to be debited", "");
    }
    else if (Logged_in == false) {
      bot.sendMessage(chat_id, "The user is not Logged in", "");
    }
  }
  else if (prev_text == "/Credit"){  //executed when the
  ↪  second-last message is text
    String amount = text;
    Serial.println("Credit started : " + amount +" "+user);
    ↪  //Serial monitor shows start of function

    String url_write = "https://script.google.com/macros/s/" +
    ↪  GOOGLE_SCRIPT_ID + "/exec?credit="+ String(amount)
    ↪  +"&userid="+String(user); // send data to sheet using
    ↪  unique variables

    HTTPClient http;
    http.begin(url_write.c_str());
    http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
    int conf = http.GET();
    String result = String(http.getString()); // redundant
```

```
}

else if (prev_text == "/Login"){ // Comes here for flag or
↪  second last text
  user = text; // latest message is username
  FlagU = false; // Flag set to false
  Serial.println("user : " + user);
  String keyboardJson = "[[\"/Enter_PIN\"],[\"/Exit\"]]"; //
  ↪  keyboard used to make user experience more interactive
  String msg3 = "You can choose one of the following
  ↪  options,\n";
  msg3 += "/Enter_PIN - to enter the password for logging
  ↪  in\n\n";
  msg3 += "/Exit - to exit the entire process \n\n";
  bot.sendMessageWithReplyKeyboard(chat_id, msg3, "",
  ↪  keyboardJson, true);


}

else if (prev_text == "/Enter_PIN"){ // Comes here for flag or
↪  second last text
  PassPIN = text; //Latest message is password
  FlagP = false; // Flag set to false
  Serial.println("PIN : " + PassPIN);
  int v = verification(user,PassPIN);
  if(v==0){ // all available options displayed
    String msg = "Successfully logged in,\n";
    msg += "You can use the following commands.\n\n";
    msg += "/Debit - to withdraw money from your bank account
    ↪  \n\n";
    msg += "/Credit - to credit money into your bank
    ↪  account\n\n";
    msg += "/Balance_Inquiry - to get your current account
    ↪  balance\n\n";
    msg += "/PIN_Change - to reset your password\n\n";
    msg += "/Fund_Transfer - to transfer money from your bank
    ↪  account to some other user";
    Logged_in = true;
```

```
    String keyboardJson = "[[\"/Debit\",
    ↪  \"/Credit\"],[\"/Balance_Inquiry\",\"/PIN_Change\"],
    [\"/Fund_Transfer\",\"/Exit\"]]"; // keyboard used to make
    ↪  user experience more interactive
    bot.sendMessageWithReplyKeyboard(chat_id, msg, "",
    ↪  keyboardJson, true); // Bot sends this message along
    ↪  with a reply keyboard
  }
  if(v==1){ // unsucessful
    String keyboardJson = "[[\"/Enter_PIN\"],[\"/Exit\"]]"; //
    ↪  keyboard used to make user experience more interactive
    String msg2 = "Incorrect Password. Select one of the
    ↪  following options:\n";
    msg2+= "\Enter_PIN - to re-enter the password";
    msg2+="\Exit - to exit the process";
    bot.sendMessageWithReplyKeyboard(chat_id, msg2, "",
    ↪  keyboardJson, true);
    PassPIN = "";
    text = "";
    prev_text = "";
    FlagU = false;
    FlagP = false;
    FlagNU = false;
    Logged_in = false;
  }
  if(v==2){ // Username doesn't exist
    String keyboardJson = "[[\"/Login\",
    ↪  \"/New_User\"],[\"/Exit\"]]"; // keyboard used to make
    ↪  user experience more interactive
    bot.sendMessageWithReplyKeyboard(chat_id, "Unauthorized
    ↪  username\nPlease retry login using a valid username",
    ↪  "", keyboardJson, true); // Bot sends this message
    ↪  along with a reply keyboard
    user = "";
    PassPIN = "";
    text = "";
    prev_text = "";
    FlagU = false;
    FlagP = false;
    FlagNU = false;
```

```
      Logged_in = false;
    }
  }

  else if (text == "/Fund_Transfer"){
    if(Logged_in == true){
      String keyboardJson = "[[\"/Exit\"]]"; // keyboard used to
      ↪  make user experience more interactive
      bot.sendMessageWithReplyKeyboard(chat_id, "Enter the
      ↪  reciever's name to whom the money is to be transferred",
      ↪  "", keyboardJson, true); // Bot sends this message
      ↪  along with a reply keyboard
    }
    else if (Logged_in == false) {
      String keyboardJson = "[[\"/Login\",
      ↪  \"/New_User\"],[\"/Exit\"]]"; // keyboard used to make
      ↪  user experience more interactive
      bot.sendMessageWithReplyKeyboard(chat_id, "You are not
      ↪  logged in", "", keyboardJson, true); // Bot sends this
      ↪  message along with a reply keyboard
    }
  }
  else if (prev_text == "/Fund_Transfer"){
    reciever = text;
    Serial.println("Fund Transfer for reciever : " +reciever);
    String keyboardJson = "[[\"/Exit\"]]"; // keyboard used to
    ↪  make user experience more interactive
    bot.sendMessageWithReplyKeyboard(chat_id, "Enter the amount to
    ↪  be transferred", "", keyboardJson, true); // Bot sends
    ↪  this message along with a reply keyboard
    FlagFT = true;
  }
  else if (FlagFT == true){
    FlagFT = false;
    funds = text;
    Serial.println("Fund Transfer for funds : " +funds);
    String url_write = "https://script.google.com/macros/s/" +
    ↪  GOOGLE_SCRIPT_ID + "/exec?sender="+ String(user)
    ↪  +"&reciever="+String(reciever) +"&fund="+String(funds);
```

```
    HTTPClient http;
    http.begin(url_write.c_str());
    http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
    int conf = http.GET();
    String result = String(http.getString());
    Serial.println("Result : " +result);
    if(result=="0"){
      String keyboardJson = "[[\"/Debit\",
      ↪  \"/Credit\"],[\"/Balance_Inquiry\",\"/PIN_Change\"],
      [\"/Fund_Transfer\",\"/Exit\"]]"; // keyboard used to make
      ↪  user experience more interactive
      bot.sendMessageWithReplyKeyboard(chat_id, "Insufficient
      ↪  funds", "", keyboardJson, true); // Bot sends this
      ↪  message along with a reply keyboard
    }
    if(result=="1"){
      String keyboardJson = "[[\"/Debit\",
      ↪  \"/Credit\"],[\"/Balance_Inquiry\",\"/PIN_Change\"],
      [\"/Fund_Transfer\",\"/Exit\"]]"; // keyboard used to make
      ↪  user experience more interactive
      bot.sendMessageWithReplyKeyboard(chat_id, "Successfully
      ↪  transferred Rs. "+funds+ " from " + user + " to
      ↪  "+reciever , "", keyboardJson, true); // Bot sends this
      ↪  message along with a reply keyboard
    }
    if(result=="2"){
      String keyboardJson = "[[\"/Debit\",
      ↪  \"/Credit\"],[\"/Balance_Inquiry\",\"/PIN_Change\"],
      [\"/Fund_Transfer\",\"/Exit\"]]"; // keyboard used to make
      ↪  user experience more interactive
      bot.sendMessageWithReplyKeyboard(chat_id, "No such username
      ↪  found for reciever", "", keyboardJson, true); // Bot
      ↪  sends this message along with a reply keyboard
    }
    reciever = "";
    funds = "";
  }
  else if (text == "/Exit"){ //Exit used to log out of the
  ↪  process and reset all flags and global variables
    user = "";
```

```
      PassPIN = "";
      text = "";
      prev_text = "";
      FlagU = false;
      FlagP = false;
      FlagNU = false;
      Logged_in = false;
      String keyboardJson = "[[\"/start\"]]"; // keyboard used to
      ↪ make user experience more interactive
      bot.sendMessageWithReplyKeyboard(chat_id, "Thank you for using
      ↪ our service!", "", keyboardJson, true); // Bot sends this
      ↪ message along with a reply keyboard
    }

  }
}

void setup() {
  Serial.begin(115200);

  // Connect to Wi-Fi
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  #ifdef ESP32
    client.setCACert(TELEGRAM_CERTIFICATE_ROOT); // Add root
    ↪ certificate for api.telegram.org
  #endif
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }
  // Print ESP32 Local IP Address
  Serial.println(WiFi.localIP());
  Serial.println(ssid);
}

void loop() {
  prev_text = bot.messages[0].text;
  if (millis() > lastTimeBotRan + botRequestDelay)  {
```

```
    int numNewMessages = bot.getUpdates(bot.last_message_received +
    ↪  1);

    while(numNewMessages) {
      Serial.println("got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received +
      ↪  1);

      Serial.println("Text: " + text);
      Serial.println("Previous Text: " + prev_text);
    }
    lastTimeBotRan = millis();
  }
}
```

Apps Script code with comments:

```
// Accessing the spreadsheet BT20ECE046
var sheet_id = "1XyvBCBYRGnV_Emr8ptMl2o-UvdIp8roN99rJ1QONGSQ";
 ↪  //Sheet ID for the current sheet
var SS = SpreadsheetApp.openById(sheet_id); // Opening the sheet

// Accessing the sheet in which the usernames and passwords are
 ↪  stored
var sheet1 = SS.getSheetByName('credentials'); // credentials
 ↪  spreadsheet accessed
// We will create alternate sheets by using the username as sheet
 ↪  name eg:var sheet1 = ss.getSheetByName('Charmander');

// var values = SS.getDataRange().getValues(); // a 2 dimensional
 ↪  array of all the data in the sheet indexed by rows and
 ↪  columns.

function doGet(e){

// for verification
  if(e.parameter.namelogin !== undefined &&
   ↪  e.parameter.pinlogin!==undefined){ //unique variables checked
    var values = sheet1.getDataRange().getValues(); // a 2
     ↪  dimensional array of all the data in the sheet indexed by
     ↪  rows and columns.
    var name = String(e.parameter.namelogin);
    var pin = String(e.parameter.pinlogin);

    // iterate in a for loop(in the first i.e. 'A' column)
    for(n=0;n<values.length;++n){
      var cell = values[n][0] ; // 0 is the index of the column

      // checking whether any value matches with the entered
       ↪  username
      if(String(cell)== String(name)){

        // Checking whether the password matches
        if(String(values[n][1]) == pin){
          return ContentService.createTextOutput(String(0)); // The
           ↪  password matches
```

```
        }
        else{
          return ContentService.createTextOutput(String(1)); // The
          ↪  password doesn't match
        }
      }
      else{ // The entered username doesn't exist
        continue;
      }
    }
    return ContentService.createTextOutput(String(3)); // No
    ↪  Username
  }

// for new user
  if(e.parameter.usernew !== undefined &&
  ↪  e.parameter.usernewpass!==undefined){//unique variables
  ↪  checked
    var name2 = String(e.parameter.usernew);
    var pin2 = String(e.parameter.usernewpass);
    var values = sheet1.getDataRange().getValues(); // a 2
    ↪  dimensional array of all the data in the sheet indexed by
    ↪  rows and columns.

    for(n=0;n<values.length;++n){
      var cell = values[n][0] ; // 0 is the index of the column

      // checking whether any value matches with the entered
      ↪  username
      if(String(cell)== String(name2)){
        return ContentService.createTextOutput(String(0));  // The
        ↪  username matched
      }
      else{
        continue;
      }
    }
    // The username doesn't match
    sheet1.appendRow([name2,pin2]);
    SS.insertSheet(String(name2));
```

```
    var sheet2 = SS.getSheetByName(String(name2));
    sheet2.getRange("A1:C1").setValues([["amount","command",
    "balance"]]);
    sheet2.getRange("A2:C2").setValues([["15000","start","15000"]]);
    return ContentService.createTextOutput(String(1)); //
     ↪  Successfully created a new user
  }

// for withdrawal
  if(e.parameter.withdraw !== undefined &&
   ↪  e.parameter.userid!==undefined){//unique variables checked

    var name = String(e.parameter.userid);
    var amt = Number(e.parameter.withdraw);
    var sheet2 = SS.getSheetByName(name); // Accessing the sheet in
     ↪  ehich this corresponding user transactions are stored

    var values = sheet2.getDataRange().getValues();
    var n = Number(values.length);
    var bal = Number(values[n-1][2]); // balance of the user

    var newbal = bal - amt; // Subtracting the withdrawn amount
     ↪  from balance
    if(newbal<0){ // Checking if the remainder is negative
      return ContentService.createTextOutput(String(0)); //
       ↪  Insufficient Balance
    }
    else{ // Successful
      sheet2.appendRow([amt,"debit",newbal]); // Successfully
       ↪  debited
      return ContentService.createTextOutput(String(1));

    }

  }

// for credit
  if(e.parameter.credit !== undefined &&
   ↪  e.parameter.userid!==undefined){//unique variables checked
```

```
    var name = String(e.parameter.userid);
    var amt = Number(e.parameter.credit);
    var sheet2 = SS.getSheetByName(name); // Accessing the sheet in
    ↪ which this corresponding user transactions are stored

    var values = sheet2.getDataRange().getValues(); // a 2
    ↪ dimensional array of all the data in the sheet indexed by
    ↪ rows and columns.
    var n = Number(values.length);
    var bal = Number(values[n-1][2]); // balance of the user

    var newbal = bal + amt; // new balance is updated by adding
    ↪ credited amount

    sheet2.appendRow([amt,"credit",newbal]);
  }

// for balance checking
  if(e.parameter.balancecheckuser !== undefined){//unique variables
  ↪ checked
    var name = String(e.parameter.balancecheckuser);
    var sheet2 = SS.getSheetByName(name); // Accessing the sheet in
    ↪ which this corresponding user transactions are stored

    var values = sheet2.getDataRange().getValues(); // a 2
    ↪ dimensional array of all the data in the sheet indexed by
    ↪ rows and columns.
    var n = Number(values.length);
    var bal = Number(values[n-1][2]); // balance of user

    return ContentService.createTextOutput(String(bal)); // The
    ↪ current balance is shown
  }

// for reseting password
  if(e.parameter.passreset !== undefined &&
  ↪ e.parameter.userid1!==undefined){//unique variables checked
    var newpassword = String(e.parameter.passreset);
    var name = String(e.parameter.userid1);
```

```
  var values = sheet1.getDataRange().getValues(); // a 2
  ↪ dimensional array of all the data in the sheet indexed by
  ↪ rows and columns.

  for(n=0;n<values.length;++n){
    var cell = values[n][0]; // 0 is the index of the column

    // checking whether any value matches with the entered
    ↪ username
    if(String(cell)== String(name)){
      var target = Number(n) + 1;
      var range = sheet1.getRange("B"+String(target));
      range.setValue(String(newpassword));
      return ContentService.createTextOutput(String(0)); //The
      ↪ password was reseted
    }

}
}

// for fund transfer
if(e.parameter.sender !== undefined &&
↪ e.parameter.reciever!==undefined &&
↪ e.parameter.fund!==undefined){//unique variables checked
  var values = sheet1.getDataRange().getValues(); //// a 2
  ↪ dimensional array of all the data in the sheet indexed by
  ↪ rows and columns.
  var sender = String(e.parameter.sender);  // sender name
  var reciever = String(e.parameter.reciever); // reciever name
  var fund = String(e.parameter.fund);
  var sheetsender = SS.getSheetByName(sender); //sheet of sender

  var valuesSender = sheetsender.getDataRange().getValues(); // a
  ↪ 2 dimensional array of all the data in the sheet indexed
  ↪ by rows and columns.



  for(n=0;n<values.length;++n){
```

```
var cell = values[n][0]; // 0 is the index of the column

// checking whether any value matches with the entered
↪ reciever
if(String(cell)== String(reciever)){ //Enter here if reciever
↪ match
  var sheetreciever = SS.getSheetByName(reciever); //sheet of
  ↪ reciever
  var valuesReciever =
  ↪ sheetreciever.getDataRange().getValues(); // a 2
  ↪ dimensional array of all the data in the sheet indexed
  ↪ by rows and columns.
  // Balance of Sender
  var s = Number(valuesSender.length);
  var balsender = Number(valuesSender[s-1][2]);
  var newbalsender = Number(balsender) - Number(fund);

  if (Number(newbalsender)<0){

    return ContentService.createTextOutput(String(0));
    ↪ //Insufficient Funds
  }
  else {
    var sheetreciever = SS.getSheetByName(reciever);
    var valuesReciever =
    ↪ sheetreciever.getDataRange().getValues(); // a 2
    ↪ dimensional array of all the data in the sheet
    ↪ indexed by rows and columns.
    // Changing the sheet of sender
    sheetsender.appendRow([fund,"debit",newbalsender]);
    // Changing the sheet of reciever

    // Balance of reciever
    var r = Number(valuesReciever.length);
    var balreciever = Number(valuesReciever[r-1][2]);

    var newbalreciever = Number(balreciever) + Number(fund);
    sheetreciever.appendRow([fund,"credit",newbalreciever]);
```

```
            return ContentService.createTextOutput(String(1)); //
             ↪  Fund Transfer successfull
          }

        }
      }
      return ContentService.createTextOutput(String("2")); // No Such
       ↪  User
}
}
```

Explanation along with output:

Basic Structure

- Here, I have used a Telegram Bot named 'BT20ECE046_ATMBot' for input acquisition from the user. Telegram is an instant messaging app which can be used to create bots that can be configured to send and recieve messages. For creating a Telegram bot, 'BotFather' bot was used. For taking the user inputs, reply keyboards were used to improve the user experience.
- The data containing the usernames, their corresponding passwords and the transaction history of the users was stored in a Google Spreadsheet. A Google Apps Script code enabled us to edit and retrieve required data from this spreadsheet.
- The ESP32 was programmed using Arduino IDE. It acted as a interface for connecting the user inputs from telegram bot and the Google Apps Script.

Commands

The ATM machine implemented using ESP32 and offers the following commands:

- /start
  This start command is used to begin the operation of the ATM. The user is greeted with a welcome message and is given further instructions.

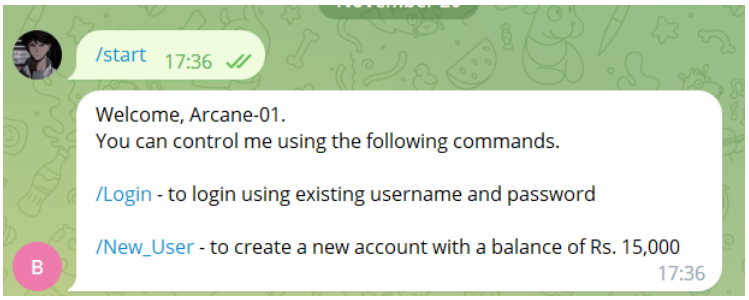

Figure 1: /start command response

- /Login and /Enter_PIN
  The Login command along with Enter PIN allows the user to access the bank account by entering the username and password.

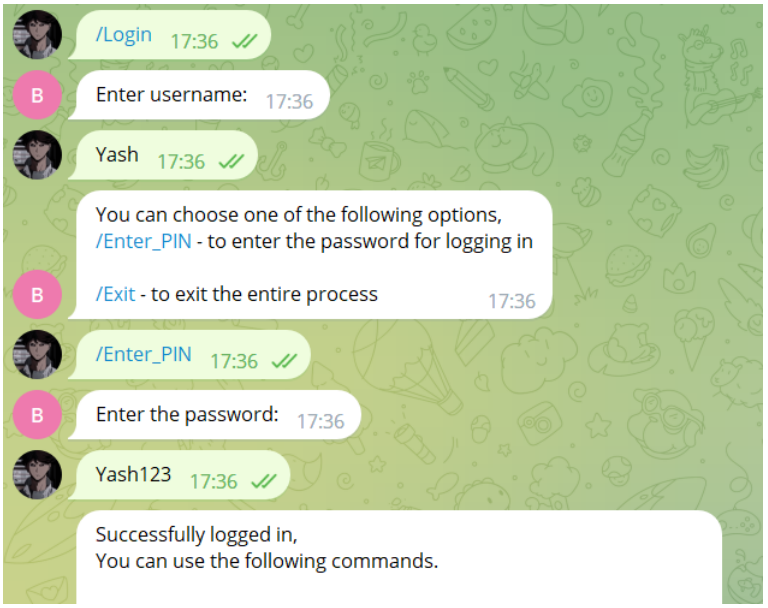

Figure 2: /Login command response

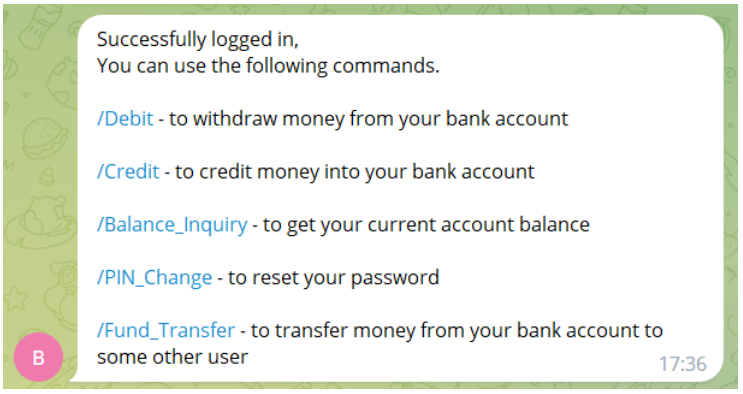

Figure 3: Login successful message

- /New_User
  This command can be used to create a new account by registering the username and password. The condition to be met for a new account creation is a unique username i.e. two bank accounts with the same username can not exist.

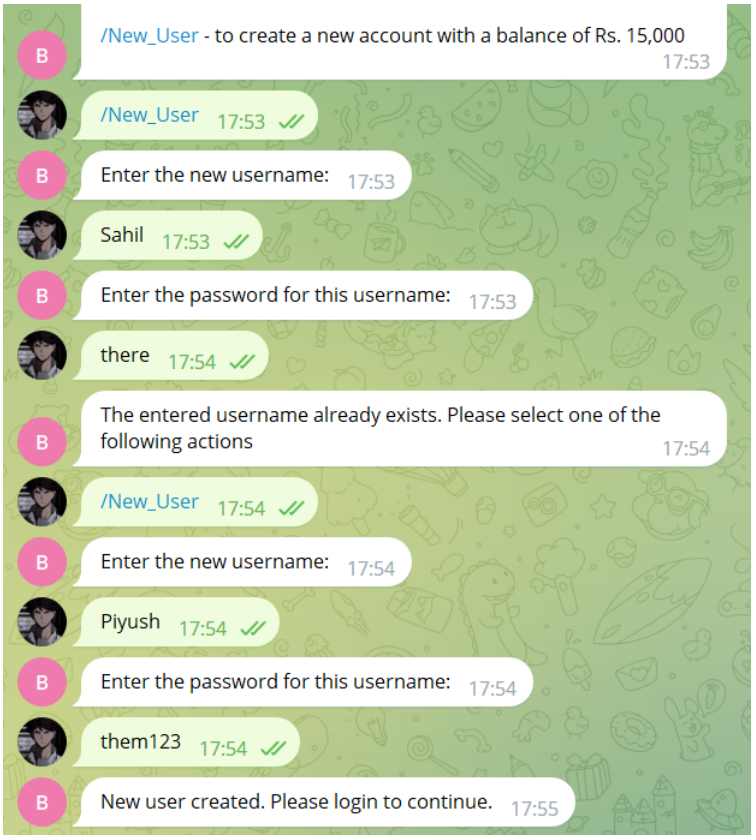

Figure 4: /New_User command response

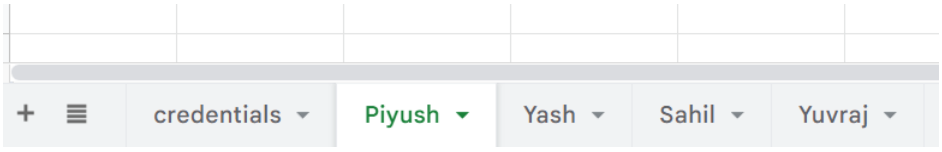

Figure 5: /New_User creates a new sheet

- /PIN_Change
  After logging in, the user can reset his/her password using this command. After updating the password the user is asked to login into the system again using this new password.

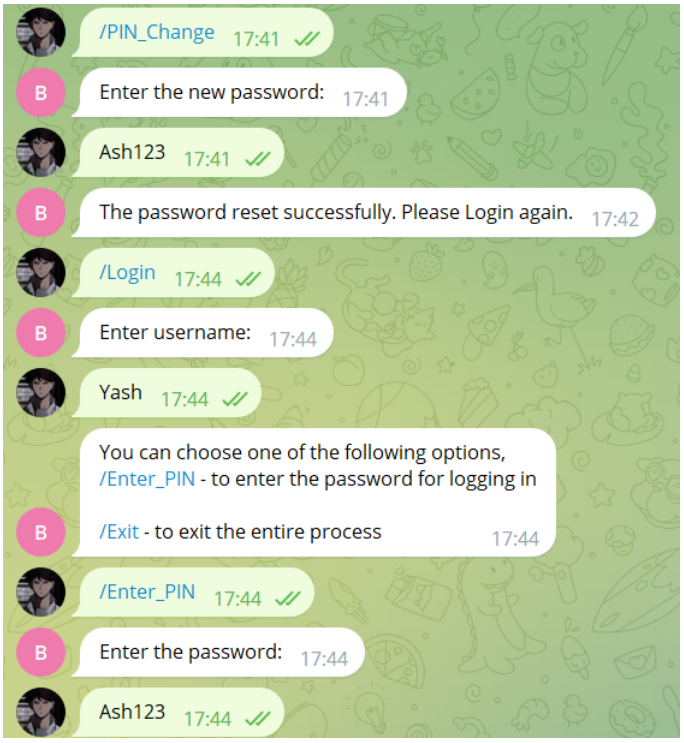

Figure 6: PIN_Change command response

- /Debit
  The Debit command allows a user to withdraw money from the bank account. If the amount to be withdrawn is found to be greater than the available balance, the user is given a warning of 'Insufficient Balance'. Whereas, if it is found to be less than the bank balance, the amount is successfully withdrawn and corresponding updates are made in the google spreadsheet corresponding to the transaction history of the user.

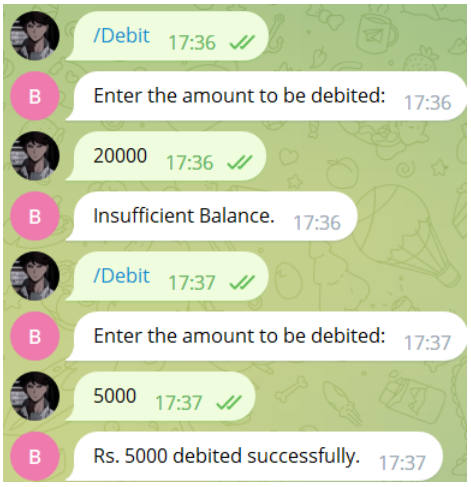

Figure 7: /Debit command response

- /Credit
  This command can be used to credit money into the user's bank account. Similar to the Debit command, the user's transaction spreadsheet gets updated.

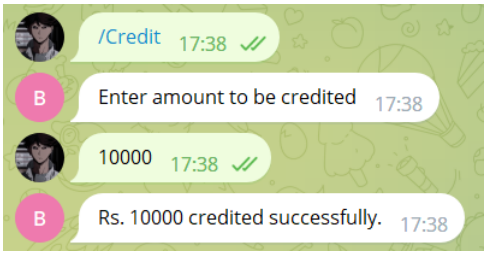

Figure 8: /Credit command response

- /Balance_Inquiry
  If a user wants to know their current bank balance, the balance inquiry command can be used.

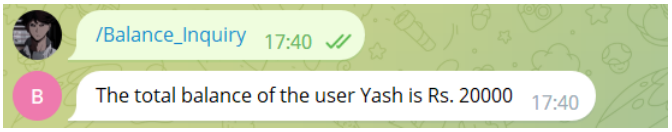

Figure 9: /Balance_Inquiry command response

- /Fund_Transfer

This command allows the user to transfer money from his/her bank account to some other account. The reciever's username will be required for this purpose. If the entered username of the reciever is not present in the database, a warning will be issued to the user. Similarly, if the amount to be transferred exceeds the bank balane of the sender, a warning about insufficient balance would be issued. If the fund transfer is successful, the transaction spreadsheets of the sender are reciever are updated.

Figure 10: /Fund_Transfer: Username not found

Figure 11: /Fund_Transfer: Insufficient Funds

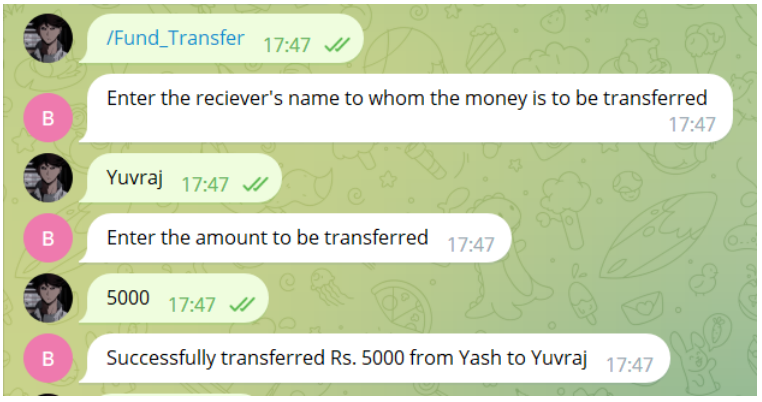

Figure 12: /Fund_Transfer: Successful

amount	command	balance
15000	start	15000
5000	debit	10000
10000	credit	20000
5000	debit	15000

Figure 13: /Fund_Transfer: Amount debited in sender's sheet

amount	command	balance
15000	start	15000
5000	credit	20000

Figure 14: /Fund_Transfer: Amount credited in reciever's sheet

- /Exit
  This command acts as log out and allows to exit all functionalities.

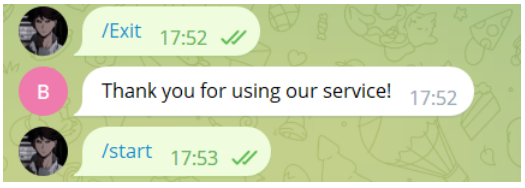

Figure 15: /Exit command response

Approach

- To take the user input from the Telegram bot the latest string which the user has entered is stored in the variable 'text' and the second last user input is stored in the variable 'prev_text'(previous text).
- If and else if loops for the different commands are run by checking the value of the text and prev_text variable. For example, if the text variable has the value /start, the if loop corresponding to start is run.
- For some commands like New_User, Fund_Transfer; multiple if-else loops were required for taking the input for username and amount. To run these loops subsequently flags were used.
- During the execution of these loops, the values corresponding to amount, username etc. were sent to Google Apps Script with the help of unique variables.
- In the Google Apps Script for the execution of the If-else blocks corresponding to the respective functions it was checked whether the unique variables are undefined or not.
- For accessing the elements of a sheet '.getDataRange().getValues()' was used. This returns a two-dimensional array containing all the elements present in the rows and columns of the sheet. This was iterated for accessing usernames, their corresponding passwords from the 'credentials' spreadsheet which contained the database of all users. A similar two dimensional array was used for accessing the total balance of a user from the sheets which contain the transaction history.
- Accessing and updating the elements in the sheet using these methods helped me execute all the commands.

Conclusion:

- The ATM machine was successfully made with the ESP32 using a Telegram bot and Google Apps Script.

Problems Faced:

- I was not able to burn my code on ESP32. Pressing the BOOT button during the uploading of the code resolved this error.
- While returning a String from the Google Apps Script I got a DataType error. While debugging the code, I observed that the creating a 2D array for a sheet

name which didn't exist resulted into this error. Refining my code in Apps Script resolved this issue.

- I observed that on adding multiple users the code slowed down. To overcome this difficulty I tried to reduce the number of for loops in my Arduino code and the Google Apps Script.
- During few trial uses of the bot, the bot had slowed down considerably. The response time had increased upto about sixty seconds. This was due to multiple devices being connected to a same WiFi source.

Links:

Youtube video link

here is the link to the YouTube video - Link

GitHub link

here is the link to the GitHub repo - Link

References:

1. https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot
2. https://developers.google.com/apps-script/guides/sheets
3. https://iotdesignpro.com/articles/esp32-data-logging-to-google-sheets-with-google-scripts
4. https://stackoverflow.com/questions/14991030/get-value-in-one-column-in-spreadsheet-using-google-apps-script