ATM MANAGEMENT SYSTEM

A MINI PROJECT REPORT
18CSC207J - ADVANCED PROGRAMMING PRACTICE

Submitted by

TEJESH [RA2111003010264]
VAISHNAV [RA2111003010265]

Under the guidance of
Jayapradha J
Assistant Professor, Department of Computing Technologies

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"ATM MANAGEMENT SYSTEM"** is the bona fide work of **TEJESH(RA211003010264)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
JAYAPRADHA	Dr. M. Pushpalatha
GUIDE	**HEAD OF THE DEPARTMENT**
Assistant Professor	Professor & Head
Department of Computing Technologies	Department of Computing Technologies

ABSTRACT

The ATM System is the project which is used to access their bank accounts in order to make cashwithdrawals. Whenever the user need to make cash withdraws, they can enter their PIN number (personal identification number) and it will display the amount to be withdrawn in the form of 50’s,100’s and 500’s. Once their withdrawn was successful, the amount will be debited in their account. The ATM will service one customer at a time. A customer will be required to enter ATM Card number, personal identification number (PIN) –both of which will be sent to the database for validation as part of each transaction. The customer will then be able to perform one or more transactions. Also customer must be able to make a balance inquiry of any account linked to the card. The ATM will provide the customer with a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, account(s), amount, and ending and available balance(s) of the affected account (“to” account for transfers).

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

IOT	Internet of Things
PIR	Passive Infrared
LCD	Liquid Crystal Diode
DHT	Distributed hash table
IR	Infra red
UART	Universal Asynchronous Receiver/Transmitter
IDE	Integrated Development Environment

CHAPTER 1

INTRODUCTION

A ATM has been acknowledged as a networking technology suitable for various environments, local or wide area, private or public, carrying any type of information from simple data packets to complex multimedia streams. It integrates features such as quality of service and traffic loss prioritization and allows for scalability and technological longevity. For these reasons, it has been proposed as a target technology for creating the B-ISDN. However, it is widely accepted that ATM has not been developed to the point where every issue is resolved, although much work has been undertaken towards its standardization.

Among the challenges currently faced by ATM technology, a primary one is certainly network management. One major reason for this is that proper exploitation of many of ATM's unique features requires advanced manipulation of abstract entities at a logical level. Thus, in ATM, while there is still a need for the ‘conventional’ management tasks (e.g. for monitoring and managing the physical level, with a view to dealing with defects, failures, and malfunctions in wires, switches, and other hardware), the main focus of the management system is set at an abstract level, with goals such as assuring QoS guarantees and exploiting, to the maximum possible extent, the versatility and the potential for multiplexing gain inherent to this networking technology. Along this view, it becomes clear that management procedures used in traditional shared-media LANs are not sufficient when it comes to managing virtual channel connections at a much higher speed.

Tasks related to ATM management are further complicated due to the complex and heterogeneous nature of the networking environment. Indeed, most of the present networks are composed of diverse pieces of ATM equipment from different vendors, as well as of non-ATM devices. Moreover, new technologies and services are introduced and integrated with existing ones at a rapid pace. Clearly, ATM management applications should support the complex procedures imposed by ATM technology itself, while at the same time being suited to composite networks.

Unfortunately, and for intrinsic reasons, end-to-end management of ATM networks cannot be fully handled by the existing protocols and systems developed for managing non-ATM networks. To draw a primary example, the Simple Network Management Protocol (SNMP), an easy to implement, low-overhead foundation for the management of multi-vendor networks of routers, servers, workstations, and other resources [35], is the most widely deployed protocol for the management of IP-based networks. Although SNMP-based solutions have been used extensively for providing a portable interface to many of the simpler ATM-related management tasks, the current solutions do not have the potential for addressing the more sophisticated management operations.

These inevitable complexities of ATM management have been identified by all parties who envisage the provision of B-ISDN through extensive use of the ATM technology. In an attempt to address the issue, the Network Management Working Group of the ATM Forum defined a five-layer network management model that provides a structured approach towards managing ATM networks [6], [7]. This model is applicable to both public and private networks, by featuring interfaces to existing SNMP, CMIP (Common Management Information Protocol), and proprietary management systems. In addition, it allows distribution of management information over ATM networks through OAM (Operations and Maintenance) cells. However, even though this standard is in place now, only a few practical implementations exist.

CHAPTER 2

LITERATURE SURVEY

The purpose of this paper is to review literature on the impact of adoption of ATM technology from the perspective of customers, banks and suppliers. As per the changing demand of the customers, innovative software solutions are regularly released. Similarly, banks are deploying CRM technology to facilitate personalized needs of customers on one to one basis. Suppliers of ATMs are also under pressure to provide ATMs to banks which can meet the latest customer needs. Design/methodology/approach - This paper is primarily based on a review of literature. Findings - The articles were categorized under two main themes- Adoption of ATM Technology and Impact of Adoption of ATM Technology .Impact of ATM Technology is further categorized into three sub themes- Customer‟s Perspectives, Bank‟s Perspectives and Supplier‟s Perspectives. This study reveals that there is a dearth of academic literature on Multi Vendor ATM Technology in developing countries .The ATM Software is to be customized with the help of personalized Technology option so as to fulfill the needs of multiple cultures in countries like India. Popularizations of biometric and multilingual ATMs are also required in Rural Areas. Originality/Values – Review on this topic has not been published in any academic journal to the best of the Author‟s knowledge. Keywords: Biometric and Multi lingual ATMs, CRM Technology, Innovative SoftwareSolutions, Multi vendor ATM Technology, Personalized Technology.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

The requirements for a bank management system provide a complete description of the system behavior and are based on the business expectations. The functioning of the system must comply with the laws and regulatory acts of the country. In the United States, for example, Consumer Financial Protection Bureau, Federal Reserve Board, Federal Deposit Insurance Corporation, and Financial Crimes Enforcement Network are the institutions that govern banking activities.

The key requirements that need to be offered by the Bank Management System can be classified into functional and non-functional requirements.

Functional Requirements

Functional Requirements describe the service that the banking management system must offer, they are subdivided into three access levels: Admin Mode, Teller Mode, and Customer Mode:

- Customer:

1. Sign in with login and password.
2. Update personal details.
3. Change password.
4. View balance.
5. View personal history of transactions.
6. Transfer money.
7. Withdraw.
8. Submit Cash.

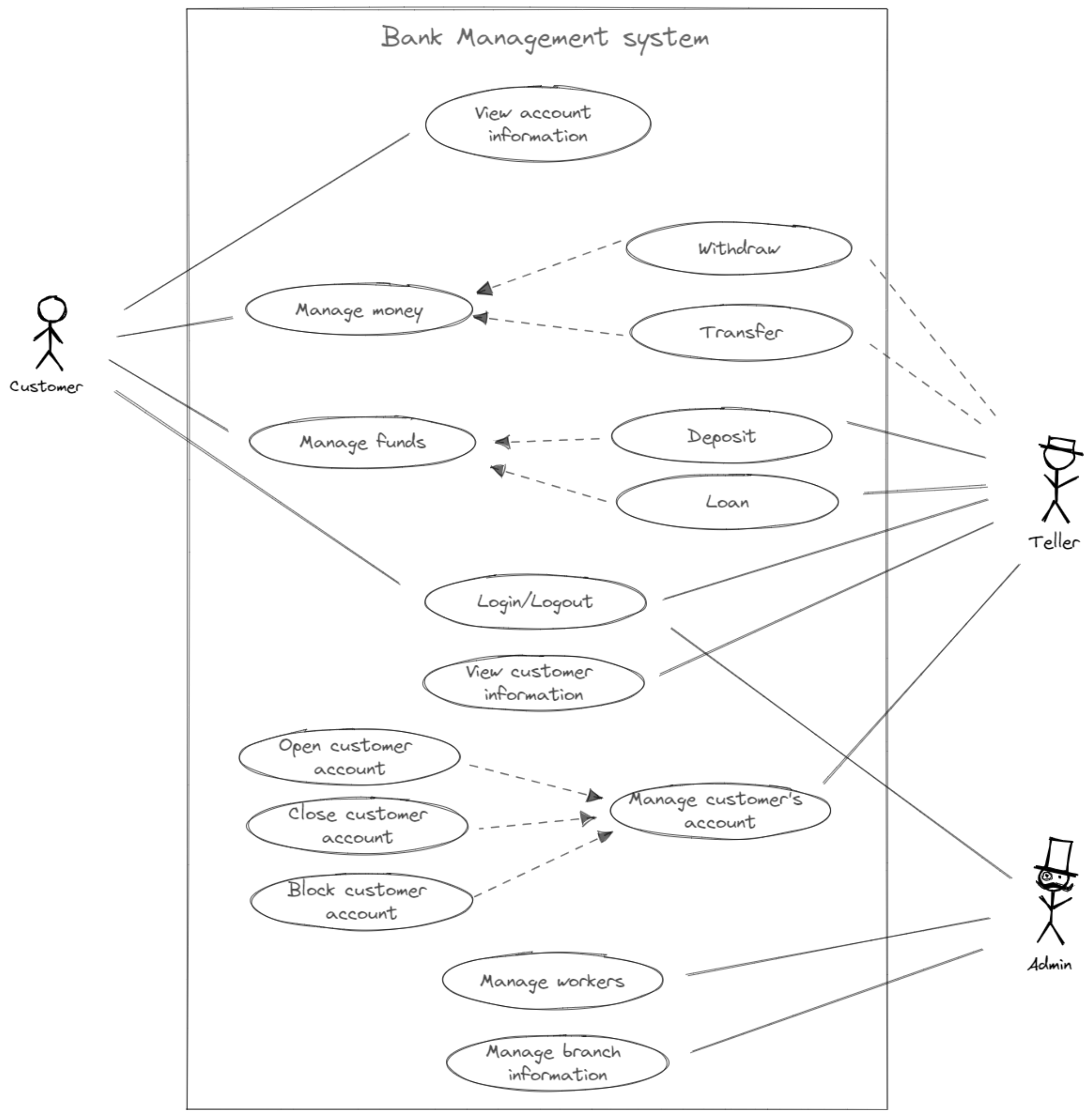
- Teller:

1. Sign in with login and password.
2. Change password.
3. Register new bank customers.
4. View customer information.
5. Manage customer accounts.

- Admin:

1. Sign in with login and password.
2. View manager and customer details.
3. Add or update bank branch details.
4. Add or update manager details.

Case diagram:



Non-Functional Requirements

Non-functional requirements specify criteria that can be used to judge the operation of a system as a whole rather than specific behaviors. They describe emergent properties like security, performance, and availability and, unlike the functional requirements that can be worked around, are essential to fulfill for a usable system. The estimation of whether the product fulfills the non-functional requirement or not usually reduces to a boolean answer: yes or no.

For a bank management system, the most important non-functional requirements include security, performance, usability, and availability.

Security

Bank management systems are notorious for being subject to malicious attacks, so security is the major requirement for the system. Unauthorized access to the data is not permissible. The data must be backed up daily and stored in a secured location, at a distance from different facilities of the system.
Online transactions and stored digital files must be encrypted according to 128-bit or 256-bit AES encryption standards. The system also must employ firewall software as a defense against network attacks.

From the client-side, the system must provide an automatic log-out after an inactivity period, accept only secure passwords that have sufficient length and non-alphabetic characters, and block login attempts after several unsuccessful trials.

Performance

The bank management system is a multi-client system that must reach response time targets for each of the clients during simultaneous calls and must be able to run a target number of transactions per second without failure. The system must effectively utilize the hardware and energy resources to minimize operational costs.

Usability

The system must provide different graphical interfaces for customers, tellers, and admins. All system interfaces must be user-friendly and simple to learn, including helping hints and messages and intuitive workflow, especially in a client interface: the client must be able to fast learn and use the interface without prior knowledge of banking terminology or rules.
The interfaces must automatically adjust to devices with different screen sizes, and allow to change typeface size and color scheme to improve readability.

Availability

The system must be available during bank working hours. The mobile banking and ATM must be available round-the-clock with minimal maintenance times, reaching 99.999% availability time per year.

CHAPTER 4

METHODOLOG
Y

Requirements gathering: Identify the requirements of the ATM management system by consulting with stakeholders such as ATM owners, operators, and users. This includes understanding the business goals, operational constraints, security requirements, and user needs.

Design: Create a high-level design of the ATM management system that takes into account the requirements gathered in step 1. This includes defining the system architecture, interfaces, data models, and algorithms.

Implementation: Develop the ATM management system based on the design created in step 2. This involves writing code, configuring hardware and software components, and integrating various subsystems.

Testing: Conduct rigorous testing of the ATM management system to ensure that it meets the requirements specified in step 1. This includes functional testing, integration testing, performance testing, and security testing.

Deployment: Deploy the ATM management system to production environment, which includes installing hardware, software, and other required components in the ATM machines.

Maintenance: Provide ongoing maintenance and support for the ATM management system, which includes monitoring system performance, addressing system issues, and implementing updates and upgrades as required. Throughout the methodology, it is important to consider security and risk management aspects of the ATM management system, as these are critical to ensuring the safety and privacy of users and their financial transactions. It is also important to follow industry standards and regulations, such as PCI DSS, to ensure compliance and minimize legal and financial risks.

CHAPTER 5

CODING AND TESTING

```
import random
import time


class Accounts:
    # Defining Account instance variables.

    def __init__(self, pin, balance,
        annualInterestRate=3.4): self.pin = pin
        self.balance = balance
        self.annualInterestRate = annualInterestRate

    # Class function to return the monthly interest rate.
    def getMonthlyInterestRate(self):
        return self.annualInterestRate / 12

    # class function to calculate difference between the balance and the amount
withdrawn.
    def withdraw(self, amount):
        self.balance -= amount

    # class function to calculate the sum between the balance and the amount
deposited.
    def deposit(self, amount):
        self.balance += amount

    # Class function to calculate the product of the balance and the annual
interest rate.
    def getAnnualInterest(self):

        return self.balance * self.annualInterestRate

    # Class function to calculate the product of the balance and the monthly
interest rate.
    def getMonthlyInterest(self):
```

```
return self.balance * self.getMonthlyInterestRate()
```

```
# Revieves pin from user input and validates input.
def getAccountPin():
    while True:
        pin = input("\nEnter four digit account pin: ")
        try:
            pin = int(pin)
            if pin >= 1000 and pin <= 9999:
                return pin
            else:
                print(f"\n{pin} is not a valid pin... Try again")
        except ValueError:
            print(f"\n{pin} is not a vaild pin... Try again")


# Recieves user input for option selection and validates selection.
def getSelection():
    while True:

        selection = input("\nEnter your selection: ")
        try:
            selection = int(selection)

            if selection >= 1 and selection <= 4:
                return selection
            else:

                print(f"{selection} is not a valid choice... Try again")
        except ValueError:
            print(f"{selection} is not a valid choice... Try again")



# Returns the current working accounts balance.
def viewBalance(workingAccount):
    return workingAccount.balance



# Recieves user input and validates if input is either yes, y, no, or n.
def correctAmount(amount):
    while True:
        answer = input(f"Is ${amount} the correct ammount, Yes or No? ")
        try:
```

```
answer = answer.lower()
if answer == "y" or answer == "yes":
```

```
                return True
            elif answer == "n" or answer == "no":
                return False
            else:
                print("Please enter a valid response")
        except AttributeError:
            print("Please enter a valid response")


# Recieves user input on amount to withdraw and validates inputed value.
def withdraw(workingAccount):
    while True:
        try:
            amount = float(input("\nEnter amount you want to withdraw: "))
            try:
                amount = round(amount, 2)
                if amount > 0 and ((workingAccount.balance) - amount) > 0:
                    answer = correctAmount(amount)
                    if answer == True:
                        print("Verifying withdraw")
                        time.sleep(random.randint(1, 2))
                        return amount
                elif (((workingAccount.balance) - amount) < 0):
                    print("\nYour balance is less than the withdraw amount")
                elif amount == 0:
                    answer = correctAmount(amount)
                    if answer == True:
                        print("Canceling withdraw")
                        time.sleep(random.randint(1, 2))
                        return amount
                else:
                    print("\nPlease enter an amount greater than or equal to 0")
            except TypeError:
                print("\nAmount entered is invalid... Try again")
        except ValueError:
            print("\nAmount entered is invalid... Try again")



# Recieves user input on amount to deposit and validates inputed value.
def deposit(workingAccount):
    while True:
        try:
            amount = float(input("\nEnter amount you want to deposit: "))
            try:
                amount = round(amount, 2)
                if amount > 0:
                    answer = correctAmount(amount)
                    if answer == True:
                        print("Verifying deposit")
                        time.sleep(random.randint(1, 2))
                        return amount
                elif amount == 0:

                    answer = correctAmount(amount)
                    if answer == True:
                        print("Canceling deposit")
                        time.sleep(random.randint(1, 2))
                        return amount
                else:

                    print("\nPlease enter an amount greater than or equal to 0")
            except TypeError:
                print("\nAmount entered is invalid... Try again")
        except ValueError:
            print("\nAmount entered is invalid... Try again")


# End of program to print out account information and return false to end main
loop
def exitATM(workingAccount):
    print("\nTransaction is now complete.")
    print("Transaction number: ", random.randint(10000, 1000000))
    print("Current Interest Rate: ", workingAccount.annualInterestRate)
    print("Monthly Interest Rate: ", workingAccount.annualInterestRate / 12)
    print("Thanks for using this ATM")
    return False



def main():
    # Creating all accounts possible, could be stored or read from a file/database
instead for better functionality overall.
    accounts = []
```

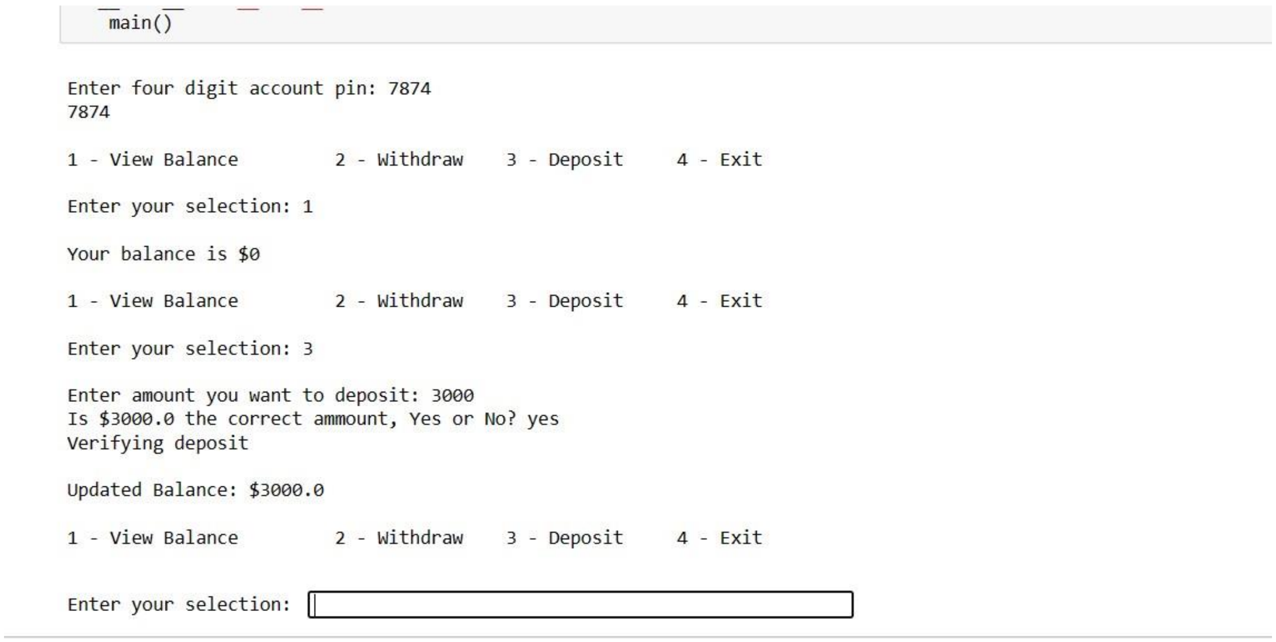
```
for i in range(1000, 9999):
    account = Accounts(i, 0)
    accounts.append(account)

# ATM Processes loop
loop = True
while loop == True:

    pin = getAccountPin()
    print(pin)
    # Account session loop
    while loop == True:
        # Menu Selection
        print("\n1 - View Balance \t 2 - Withdraw \t 3 - Deposit \t 4 - Exit ")

        selection = getSelection()

        # Getting working account object by comparing pins
        for acc in accounts:
            # Comparing user inputted pin to pins created
            if acc.pin == pin:
                workingAccount = acc
                break

        # View Balance
        if selection == 1:
            print(f"\nYour balance is ${viewBalance(workingAccount)}")
        # Withdraw
        elif selection == 2:
            workingAccount.withdraw(withdraw(workingAccount))
            print(f"\nUpdated Balance: ${workingAccount.balance}")
        # Deposit

        elif selection == 3:
            workingAccount.deposit(deposit(workingAccount))
            print(f"\nUpdated Balance: ${workingAccount.balance}")
        # Exit
        elif selection == 4:
            loop = exitATM(workingAccount)
        # Invalid input
        else:
            print("Enter a valid choice")
```

```
if __name___== "__main__":
```

```
main()
```

CHAPTER 6

SCREENSHOTS AND RESULTS



```
    main()

Enter four digit account pin: 7874
7874

1 - View Balance         2 - Withdraw    3 - Deposit     4 - Exit

Enter your selection: 1

Your balance is $0

1 - View Balance         2 - Withdraw    3 - Deposit     4 - Exit

Enter your selection: 3

Enter amount you want to deposit: 3000
Is $3000.0 the correct ammount, Yes or No? yes
Verifying deposit

Updated Balance: $3000.0

1 - View Balance         2 - Withdraw    3 - Deposit     4 - Exit


Enter your selection:
```

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, a ATM Management System project in Python can be a useful tool for managing banking operations and transactions. By defining classes and implementing the logic, the system can provide various features such as creating and managing accounts, depositing and withdrawing funds, checking balances, and viewing transaction history.

REFERENCES

[1] Zhang, X., Davidson, E. A,"Improving Nitrogen and Water Management in Crop Production on a National Scale", American Geophysical Union, December, 2018.How to Feed the World in 2050 by FAO.

[2] Abhishek D. et al., "Estimates for World Population and Global Food Availability for Global Health", Book chapter, The Role of Functional Food Security in Global Health, 2019, Pages 3-24.Elder M., Hayashi S., "A Regional Perspective on Biofuels in Asia", in Biofuels and Sustainability, Science for Sustainable Societies, Springer, 2018.

[3] Zhang, L., Dabipi, I. K. And Brown, W. L, “Internet of Things Applications for Agriculture". In, Internet of Things A to Z: Technologies and Applications, Q. Hassan (Ed.), 2018.

[4] S. Navulur, A.S.C.S. Sastry, M.N. Giri Prasad,"Agricultural Management through Wireless Sensors and Internet of Things" International Journal of Electrical and Computer Engineering (IJECE), 2017; 7(6) :3492-3499.

[5] E. Sisinni, A. Saifullah, S.Han, U. Jennehag and M.Gidlund, "Industrial Internet ofThings: Challenges,Opportunities, and Directions," in IEEE Transactions on Industrial Informatics, vol. 14, no. 11, pp. 4724-4734, Nov. 2018.

[6] M. Ayaz, M. Ammad-uddin, I. Baig and e. M. Aggoune, "Wireless Possibilities: A Review," in IEEE Sensors Journal, vol. 18, no. 1, pp. 4-30, 1 Jan.1, 2018.