Science and Technology Publishing (SCI & TECH)
ISSN: 2632-1017
Vol. 6 Issue 1, January - 2022

Development Of Mechanism For Cluster-Based Grant Registration And Disbursement By Charitable Organizations

Ntuen, Bassey Cletus
Department Of Electrical/Electronic And Computer Engineering, University of Uyo, Akwa Ibom State Nigeria

Philip M. Asuquo
Department Of Electrical/Electronic And Computer Engineering, University of Uyo, Akwa Ibom State Nigeria

Kalu Constance
Department Of Electrical/Electronic And Computer Engineering, University of Uyo, Akwa Ibom State Nigeria

***Abstract*— The development of mechanism for cluster-based grant registration and disbursement by charitable organizations is presented. The emphases in this paper is development of a mechanism that will be used to organize the beneficiaries in clusters during grant registration process under the coordination of different charitable organizations spread across the nation. The mechanism entails module for registration of every cluster that belongs to a given charitable organization. Furthermore, mechanism is also presented for the registration of the directors and beneficiaries in each cluster, production of the pay schedule, handling of those beneficiaries and directors without bank account, disbursement of funds to accredited beneficiaries and directors and handling of those accredited beneficiaries and directors that are omitted or not correctly paid in the disbursement process. Accordingly, the functional decomposition of the Cluster-Based Grant Registration and Disbursement (ABGRD) mechanism for charitable organization is presented along with the detailed algorithm for the implementation of each of the functional units.**

Keywords — Cluster-Based Grant Management, Third Party NIN Validation, Grant Beneficiary, Pay Schedule, Grant Registration, Third Party Validation Bank Account, Charitable Organizations, Grant Disbursement

1. Introduction

A charitable organizations or charitable are nonprofit organization which focus on addressing different aspects of societal challenges by generating or acquiring funds and deploying the funds to beneficiaries and projects that touch lives and address those challenges [1,2,3,4,5,6,7,8,9,10]. Well established charitable organizations do get grants to be disbursed to various categories of beneficiaries. However, one of the challenges of the grant management is the registration and validation of the beneficiaries as well as the disbursement of the funds to those validated beneficiaries [11,12,13,14,15].

Mostly, the charitable organizes the grant beneficiaries into manageable groups called clusters and the registration of beneficiaries are none at cluster level [16,17, 18,19, 20,21, 22, 23]. The use of web application in the grant registration is generally low and the challenges of ensuring that multiple registration by one beneficiary is rarely guaranteed. As such, some beneficiaries defraud the system by engaging in multiple registration. Also, some charitable organization managers register fictitious names so that they can recoup the money meant for the beneficiaries by using their own account to collect the money from different beneficiaries. Accordingly, this paper presents detailed design of a mechanism that can be used to address all the stated challenges of grant registration and disbursement by charitable organizations. The mechanism verifies account details, national identification numbers and phone number of beneficiaries and checks for duplicate entries at the registration point. This eliminates the occurrence of multiple registration by beneficiaries. In all, by adopting the mechanism, the charitable organizations are equipped with a means to ensure equitable distribution of the grant across the clusters and beneficiaries.

2. The functional decomposition the mechanism

The mechanism entails module for registration of every cluster that belongs to a given charitable organization. Registration of the directors and beneficiaries in each cluster, production of the pay schedule, handling of those beneficiaries and directors without bank account, disbursement of funds to accredited beneficiaries and directors and handling of those accredited beneficiaries and directors that are omitted or not correctly paid in the disbursement process. Accordingly, the functional decomposition of the Cluster-Based Grant Registration and Disbursement (ABGRD) mechanism for charitable organization is as given in Figure 1.

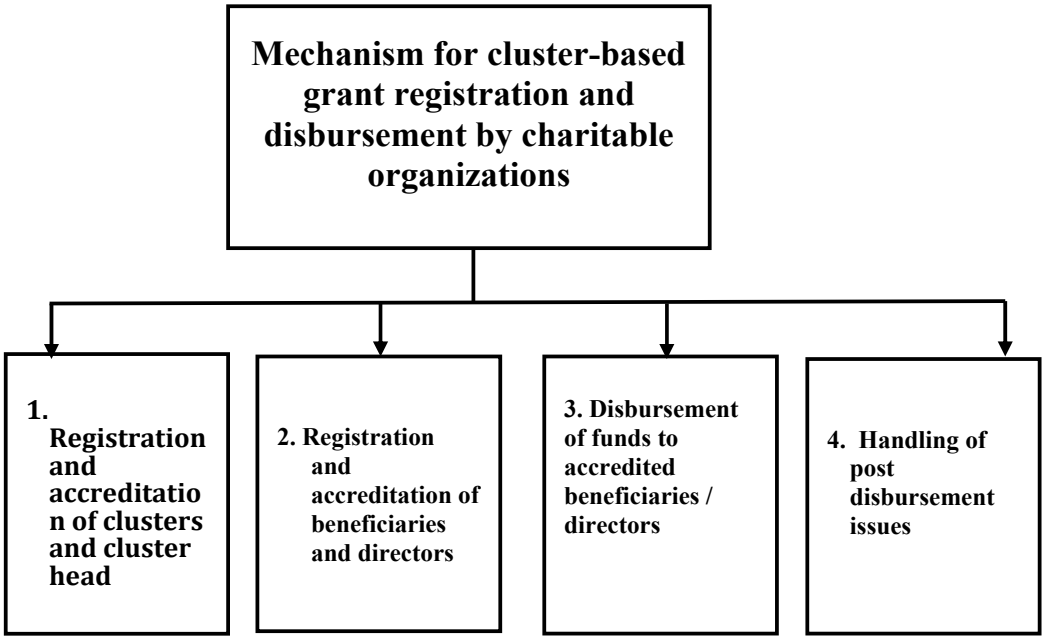

Figure 1. The functional decomposition of the Cluster-Based Grant Registration and Disbursement (ABGRD) mechanism for charitable organization is as given in

3. Registration and accreditation of clusters and cluster heads by the charitable organizations

3.1 The required parameters for clusters and cluster heads registration and accreditation

In order to disburse grant money to the beneficiaries, the charitable organizations are expected to form clusters of beneficiaries. In this work, each beneficiaries is expected to register for the grant through the charitable organizations. In turn, each charitable organization is expected to register clusters and each cluster should have a cluster head. The required parameters for clusters and cluster heads registration and accreditation are shown in Table 1.

Table 1: The required parameters for clusters and cluster heads registration and accreditation using Algorithm 1 and Algorithm 2 respectively

Parameter Description	*Parameter Name for programming purpose*
Cluster Name	*ClName*
Cluster Number	*ClNum*
Cluster State and local government of operation	*ClState and ClLGA*
Beneficiary counter is *denoted as an array where Benef*Count *[1] is for cluster head, Benef*Count *[2] is for directors and Benef*Count *[3] is for ordinary beneficiaries*	*Benef*Count[1] *Benef*Count[2] *Benef*Count[3]
A maximum of cluster head, director and ordinary beneficiaries denoted as an array where NBmax [1] is for cluster head, NBmax [2] is for directors and NBmax [3]is for ordinary beneficiaries	*NBmax [1]* *NBmax [2]* *NBmax [3]*
Cluster head counter	ClHdCount
Cluster head Name	*ClHdName*
Beneficiary Status (1 for cluster head, 2 for director, 3 for other beneficiaries)	*BenefStatus*
Cluster head national identification number	*ClHdNIN*
Cluster head Phone	*ClHdPhone*
Cluster head email	*ClHdMail*
Cluster head State of origin	*ClHdStateOrg*
Cluster head LGA of origin	*ClHdLGAOrg*
Cluster head contact address	*ClHdAddress*
Cluster head passport size facial photo	*ClHdPhoto*
Cluster head number	*ClHdNum*
Cluster head account number	*ClHdAccNum*
Cluster head Bank name	*ClHdBankName*

3.2 The process flow and algorithm for cluster registration and accreditation

The process flows or sequence of actions for creation of a cluster on the charitable organization web portal for grant management (WP4GM) is as follows:

i. Access the charitable organization web portal for grant management (WP4GM).
ii. Create a cluster name (ClName) and check for duplicate names; reject name if duplicate exists, otherwise retain the name
iii. Use the WP4GM to generate unique cluster identification number for the cluster, hence duplicate cluster number is not permitted and will not exist in the WP4GM database.
iv. Input the following cluster parameters listed in Table 1: *NBmax [1], NBmax [2], NBmax [3]*, ClState , ClLGA
v. Initialize the following BenefCount[1],BenefCount[2],BenefCount[3]
vi. End process

The algorithm for the cluster creation and validation process is given in Algorithm 1.

Algorithm 1 Cluster Creation and Validation Algorithm

1: Input: ClName *// Input and validate cluster name*
2: If UniqueTestClName (ClName) = "false" **then**
3: Reject ClName
4: **Else**
5: Accept ClName
6: Endif
7: Input: *NBmax*
8: Input: *NDmax*
9: Input: ClState and ClLGA
10: CreateClnumber(*ClNum)*
11: Inpt NBmax [1]; NBmax [2] ; NBmax [3] // *Input maximum number of cluster head counter ,* NBmax [1]; *maximum number of directors,* NBmax [2]; *and maximum number of beneficiaries ,* NBmax [3]
12: BenefCount[1]=0; BenefCount[2] =0 ; BenefCount[3]=0 // *Initialize current the cluster head counter , drector counter and beneficiary counter, all set to 0*
13: End *// End cluster creation process*

Note for Algorithm 1. The following functions are used in Algorithm 1 which is cluster creation and validation algorithm:

I. UniqueTestClName(ClName) is a function that is used to search the WP4GM database if there is no duplicate of the cluster name (ClName) in the database. The function returns "false" if a duplicate of ClName already exists in the WP4GM database otherwise it returns "true".
II. CreateClnumber(*ClNum)* is a function that is used to create unique cluster number and assign it to the variable *ClNum*

3.3 The process flow and algorithm for cluster head registration and validation

The process flows or sequence of actions for cluster head registration and validation on the charitable organization web portal for grant management (WP4GM) is as follows:

i. Input cluster number (ClNum) and cluster head count (ClHdCount)
ii. Check if head already exists for the cluster (that is ClHdCount > 0) and abort the cluster head registration and validation otherwise if ClHdCount ≤ 0 continue with the cluster head registration and validation
iii. Set cluster head counter (ClHdCount) to 1 if it is zero otherwise abort the cluster head registration. More than one cluster head is not allowed.
iv. Access the charitable organization web portal for grant management (WP4GM).
v. Input cluster head name (ClHdName) which consists of surname, first name and other name.
vi. Input cluster head national identification number (ClHdNIN).

a) Check for duplicate ClHdNIN in the WP4GM database, reject ClHdNIN if duplicate exists, and otherwise retain the name.
b) Validate that the name, ClHdName matches with the name on the ClHdNIN, reject ClHdName and ClHdNIN if the two names do not match otherwise retain the ClHdName and ClHdNIN.

vii. Input cluster head phone number (*ClHdPhone*).
a) Check for duplicate *ClHdPhone* in the WP4GM database; reject *ClHdPhone* if duplicate exists, and otherwise retain the *ClHdPhone*.
b) Check if cluster head phone number (ClHdPhone) is linked to ClHdNIN; reject *ClHdPhone* if it is not linked to ClHdNIN otherwise retain the *ClHdPhone*.
c) Check if the *ClHdPhone is active*; reject *ClHdPhone* if it is not active otherwise retain the *ClHdPhone*.

viii. Input cluster head bank account number (*ClHdAccNum*).
a) Check for duplicate *ClHdAccNum* in the WP4GM database, reject *ClHdAccNum* if duplicate exists, and otherwise retain the name.
b) Validate that the name, ClHdName matches with the name on the *ClHdAccNum*, reject ClHdName and *ClHdAccNum* if the two names do not match otherwise retain the ClHdName and *ClHdAccNum*

ix. Input cluster head email (ClHdMail)
x. Input cluster head State of origin (ClHdStateOrg)
xi. Input cluster head LGA of origin (ClHdLGAOrg)
xii. Input cluster head contact address (ClHdAddress)
xiii. Upload cluster head passport size photo (*ClHdPhoto*)
xiv. BenefBankSerNum[1] =1
xv. Assign unique number to the cluster head number relative to the cluster number as listed in Table 1.
xvi. Set the beneficiary status to 1 (BenefStatus =1)
xvii. Send successful registration message to the cluster head along with the profile and cluster head number.

The algorithm for the cluster head registration and validation process is given in Algorithm 2.

Algorithm 2 Cluster Head Registration and Validation Algorithm

```
1: Input: Cluster number (ClNum) and cluster head count (ClHdCount)
2: If: ClHdCount =0 then
3:         ClHdCount =1
4:  else
5:         Abort cluster head registration // Cluster head already exists
6: endif
7: Input: ClHdName  //Input surname, first name and other names
8: Input: ClHdNIN  // Input and validate cluster head national identification number
3: If  UniqueTestNIN(ClHdNIN) = "false" then
4:       Reject ClHdNIN
5:     Else
6:        If  NINNameCheck(ClHdNIN) = "false" then
7:           Reject ClHdNIN and ClHdName
8:        Else
9:           Accept ClHdNIN and ClHdName
10:       Endif
11: Endif
12: Input: ClHdPhone // Input and validate cluster head phone number
13:  If  UniqueTestPhone(ClHdPhone) = "false" then
14:      Reject ClHdName and ClHdPhone
15:   Else
16:       If  PhoneNINCheck(ClHdPhone) = "false" then
17:          Reject ClHdName and ClHdPhone
18:       Else
19:          If  PhoneActivCheck(ClHdPhone) = "false" then
20:             Reject ClHdName and ClHdPhone
21:          Else
22:             Accept ClHdName and ClHdPhone
23:          Endif
24:       Endif
25: Endif
26: Input ClHdAccNum // Input and validate cluster head bank account number
27: If  UniqueTestAcc(ClHdAccNum) = "false" then
28:      Reject ClHdAccNum
29:    Else
30:      If  AccNameCheck(ClHdAccNum) = "false" then
31:         Reject ClHdAccNum and ClHdName
32:      Else
33:         Accept ClHdAccNum and ClHdName
34:      Endif
35: Endif
36:  Input: ClHdMail
37:  Upload: ClHdPhoto
38:  CreateClHdnumber(ClHdNum)
39:  BenefStatus =1
40:  Notify the custer head of the "successful" or "unsuccessful" registration
41: End            // End cluster head registration and validation process
```

Note for Algorithm 2. The following functions are used in Algorithm 2 which is cluster head registration and validation algorithm:

I. *UniqueTestNIN(ClHdNIN)* is a function that is used to search the WP4GM database if there is no duplicate of the *cluster head national*

identification number (ClHdNIN) in the database. The function returns "false" if a duplicate of ClHdNIN already exists in the WP4GM database otherwise it returns "true".

II. *NINNameCheck(ClHdNIN)* is a function that use a third party API (application program interface) to bring out the name of the person that has the NIN number (ClHdNIN). It returns "false" if the name supplied by the cluster head is different from the name returned from the third part API for verification of ClHdNIN otherwise it returns "true".

III. *UniqueTestPhone(ClHdPhone)* is a function that is used to search the WP4GM database if there is no duplicate of the *cluster head phone number* (*ClHdPhone*) in the database. The function returns "false" if a duplicate of *ClHdPhone* already exists in the WP4GM database otherwise it returns "true".

IV. *PhoneNINCheck(ClHdPhone)* is a function that use a third party API to bring out the phone number that is linked to a NIN number. It returns "false" if the phone number (ClHdPhone) supplied by the cluster head is different from the phone number returned from the third part API for verification of ClHdNIN otherwise it returns "true".

V. *PhoneActivCheck(ClHdPhone) is a function that is used to check if the phone number (ClHdPhone) is active (capable of receiving calls and SMS messages). The function generates a code and sends it as SMS message to the phone. The phone user is prompted to input the code into the PhoneActivCheck(ClHdPhone) validation form. If the code entered by the user is correct, then the phone is considered active and the function returns "true" otherwise it returns "false".*

VI. *UniqueTestAcc(ClHdAccNum)* is a function that is used to search the WP4GM database if there is no duplicate of the *cluster head bank account number* (*ClHdAccNum*) in the database. The function returns "false" if a duplicate of *ClHdAccNum* already exists in the WP4GM database otherwise it returns "true".

VII. *AccNameCheck(ClHdAccNum)* is a function that use a third party API to bring out the name of the person that has the bank account number (*ClHdAccNum*). It returns "false" if the name supplied by the cluster head is different from the name returned from the third part API for verification of *ClHdAccNum* otherwise it returns "true".

VIII. *CreateClHdnumber(ClHdNum)* is a function that is used to create unique cluster head identification number and assign it to the variable *ClHdNum.*

4. Registration and accreditation of cluster beneficiaries and directors by the Charitable Organizations

4.1 The required parameters for registration and accreditation of cluster beneficiaries and directors

Every beneficiary is expected to register for the grant through the charitable organizations. Notably, the registration and accreditation of cluster beneficiaries and directors by the charitable organizations is very similar to that of the cluster head, except that the cluster head is of status value of 1 and the maximum number of allowed cluster head per cluster is different from those of the directors and ordinary beneficiaries. The required parameters for registration and accreditation of cluster beneficiaries and directors are shown in Table 2.

Table 2: The required parameters for registration and accreditation of cluster beneficiaries and directors using Algorithm 3

Parameter Description	*Parameter Name for programming purpose*
Cluster Number	*ClNum*
Beneficiary counter is denoted as an array where BenefCount [1] is for cluster head, BenefCount [2] is for directors and BenefCount [3] is for ordinary beneficiaries	*Benef*Count[1] *Benef*Count[2] *Benef*Count[3]
A maximum of cluster head, director and ordinary beneficiaries denoted as an array where NBmax [1] is for cluster head, NBmax [2] is for directors and NBmax [3]is for ordinary beneficiaries	*NBmax [1]* *NBmax [2]* *NBmax [3]*
Beneficiary Status (1 for Beneficiary , 2 for director, 3 for other beneficiaries)	*BenefStatus*
Beneficiary Name	*ClHdName*

Beneficiary national identification number	*ClHdNIN*
Beneficiary Phone	*BenefPhone*
Beneficiary email	*BenefMail*
Beneficiary State of origin	*BenefStateOrg*
Beneficiary LGA of origin	*BenefLGAOrg*
Beneficiary contact address	*BenefAddress*
Beneficiary passport size facial photo	*BenefPhoto*
Beneficiary number	*BenefNum*
Beneficiary account number	*BenefAccNum*
Beneficiary Bank name	*BenefBankName*
Beneficiary cluster serial number	*BenefBankSerNum[1]* *BenefBankSerNum[1]* *BenefBankSerNum[1]*

4.2 The process flow and algorithm for registration and accreditation of cluster beneficiaries and directors

The process flows or sequence of actions for beneficiaries and directors registration and accreditation on the charitable organization web portal for grant management (WP4GM) is as follows:

i. Access the charitable organization web portal for grant management (WP4GM).
ii. Input cluster number (ClNum) and *BenefStatus* (as listed in Table 2)
iii. Check for availability of slot, SlotsAvailable(*BenefStatus)*. If SlotsAvailable(*BenefStatus) is "false" then* abort the beneficiary and director registration and validation otherwise continue with the beneficiary registration and validation
iv. Input beneficiary name (BenefName) which consists of surname, first name and other name.
v. Input beneficiary national identification number (BenefNIN).
   a) Check for duplicate BenefNIN in the WP4GM database, reject BenefNIN if duplicate exists, and otherwise retain the name.
   b) Validate that the name, BenefName matches with the name on the BenefNIN, reject BenefName and BenefNIN if the two names do not match otherwise retain the BenefName and BenefNIN.
vi. Input beneficiary phone number (*BenefPhone*).
   a) Check for duplicate *BenefPhone* in the WP4GM database; reject *BenefPhone* if duplicate exists, and otherwise retain the *BenefPhone*.
   b) Check if beneficiary phone number (BenefPhone) is linked to BenefNIN; reject *BenefPhone* if it is not linked to BenefNIN otherwise retain the *BenefPhone*.
   c) Check if the *BenefPhone is active*; reject *BenefPhone* if it is not active otherwise retain the *BenefPhone*.
vii. Input beneficiary bank account number (*BenefAccNum*).
   a) Check for duplicate *BenefAccNum* in the WP4GM database, reject *BenefAccNum* if duplicate exists, and otherwise retain the name.
   b) Validate that the name, BenefName matches with the name on the *BenefAccNum*, reject BenefName and *BenefAccNum* if the two names do not match otherwise retain the BenefName and *BenefAccNum*
viii. Input beneficiary email (BenefMail)
ix. Input beneficiary State of origin (BenefStateOrg)
x. Input beneficiary LGA of origin (BenefLGAOrg)
xi. Input beneficiary contact address (BenefAddress)
xii. Upload beneficiary passport size photo (*BenefPhoto*)
xiii. Increment Beneficiary counter (*Benef*Count) by 1 , that is *Benef*Count = *Benef*Count+1
xiv. BenefBankSerNum [BenefStatus] = *Benef*Count.
xv. Send successful registration message to the beneficiary along with the profile and beneficiary number.

The algorithm for the beneficiary registration and validation process is given in Algorithm 3.

Algorithm 3 Beneficiaries and Directors Registration and Validation Algorithm

Input: Cluster number (*ClNum)* and *BenefStatus*
If: *BenefCount[BenefStatus] ≥ NBmax [BenefStatus]* **then**
    Abort beneficiaries and directors registration and validation
else
    Continue with beneficiaries and directors registration and validation
endif
Input: *BenefName //Input surname, first name and other names*
Input: *BenefNIN // Input and validate cluster head national identification number*
If *UniqueTestNIN(BenefNIN) = "false"* **then**
    Reject BenefNIN
Else
    If *NINNameCheck(BenefNIN) = "false"* **then**
        Reject BenefNIN and BenefName
    Else
        Accept BenefNIN and BenefName
    Endif
Endif
Input: *BenefPhone // Input and validate cluster head phone number*
If *UniqueTestPhone(BenefPhone) = "false"* **then**
    Reject BenefName and BenefPhone
Else
    If *PhoneNINCheck(BenefPhone) = "false"* **then**
        Reject BenefName and BenefPhone
    Else
        If *PhoneActivCheck(BenefPhone) = "false"* **then**
            Reject BenefName and BenefPhone
        Else
            Accept BenefName and BenefPhone
        Endif
    Endif
Endif
Input *BenefAccNum // Input and validate cluster head bank account number*
If *UniqueTestAcc(BenefAccNum) = "false"* **then**
    Reject BenefAccNum
Else
    If *AccNameCheck(BenefAccNum) = "false"* **then**
        Reject BenefAccNum and BenefName
    Else
        Accept BenefAccNum and BenefName
    Endif
Endif
Input: *BenefMail*
Upload: *BenefPhoto*
BenefCount[BenefStatus] = BenefCount[BenefStatus] +1
BenefBankSerNum [BenefStatus] = BenefCount[BenefStatus]
Notify the custer head of the "successful" or "unsuccessful" registration
End *// End cluster head registration and validation process*

Note for Algorithm 3. The following functions are used in Algorithm 2 which is Beneficiary registration and validation algorithm:

IX. *UniqueTestNIN(BenefNIN)* is a function that is used to search the WP4GM database if there is no duplicate of the *Beneficiary national identification number* (BenefNIN) in the database. The function returns "false" if a duplicate of BenefNIN already exists in the WP4GM database otherwise it returns "true".

X. *NINNameCheck(BenefNIN)* is a function that use a third party API (application program interface) to bring out the name of the person that has the NIN number (BenefNIN). It returns "false" if the name supplied by the Beneficiary is different from the name returned from the third part API for verification of BenefNIN otherwise it returns "true".

XI. *UniqueTestPhone(BenefPhone)* is a function that is used to search the WP4GM database if there is no duplicate of the *Beneficiary phone number* (*BenefPhone*) in the database. The function returns "false" if a duplicate of *BenefPhone* already exists in the WP4GM database otherwise it returns "true".

XII. *PhoneNINCheck(BenefPhone)* is a function that use a third party API to bring out the phone number that is linked to a NIN number. It returns "false" if the phone number (BenefPhone) supplied by the Beneficiary is different from the phone number returned from the third part API for verification of BenefNIN otherwise it returns "true".

XIII. *PhoneActivCheck(BenefPhone) is a function that is used to check if the phone number (BenefPhone) is active (capable of receiving calls and SMS messages). The function generates a code and sends it as SMS message to the phone. The phone user is prompted to input the code into the PhoneActivCheck(BenefPhone) validation form. If the code entered by the user is correct, then the phone is considered active and the function returns "true" otherwise it returns "false".*

XIV. *UniqueTestAcc(BenefAccNum)* is a function that is used to search the WP4GM database if there is no duplicate of the *Beneficiary bank account number* (*BenefAccNum*) in the database. The function returns "false" if a duplicate of *BenefAccNum* already exists in the WP4GM database otherwise it returns "true".

XV. *AccNameCheck(BenefAccNum)* is a function that use a third party API to bring out the name of the person that has the bank account number (*BenefAccNum*). It returns "false" if the name supplied by the Beneficiary is different from the name returned from the third part API for verification of *BenefAccNum* otherwise it returns "true".

XVI. *CreateBenefnumber(BenefNum)* is a function that is used to create unique Beneficiary identification number and assign it to the variable *BenefNum.*

5. The disbursement of funds to accredited beneficiaries and directors

The disbursement process is accomplished by preparing a pay schedule which contains the cluster information as well as the information about the various categories of beneficiaries and the amount of money due to each of the beneficiaries. The required parameters for pay the schedule are presented in Table 3. The pay schedule is sent to the bank which waits for specific instruction on which date and time frame the beneficiary accounts will be credited.

Before the banks are instructed to credit the beneficiaries account, the beneficiaries are notifies through SMS message or email about the disbursement and the amount they are due to receive. Also, the period the beneficiaries will expect the fund will be communicated in the same notification message.

Finally, once the notification is done, the banks are given the final instruction to pay, then, the banks credit each of the beneficiaries account with the amount listed for the beneficiary and then forwards the payment details to the WP4GM database. The payment details from the bank is used to address post-disbursement issues.

Table 3 The required parameters for pay schedule Algorithm 3

Parameter Description	*Parameter Name for programming purpose*
Cluster Number	*ClNum*
Cluster Name	*ClName*
Beneficiary Name	*ClHdName*
Beneficiary Status (1 for Beneficiary , 2 for director, 3 for other beneficiaries)	*BenefStatus*
Beneficiary cluster serial number	*BenefBankSerNum*
Beneficiary unique number	*BenefNum*
Beneficiary account number	*BenefAccNum*
Beneficiary Bank name	*BenefBankName*
Beneficiary Grant Total Sum	*BenefGTS*
Beneficiary Phone	*BenefPhone*
Beneficiary email	*BenefMail*
Beneficiary passport size facial photo	*BenefPhoto*

6. Handling of post disbursement issues and other issues not covered by the strategy presented in this paper

Post disbursement issues are handled with the help of the cluster heads who will organize meetings for the cluster members and collect all their complaints and forwards same to the charitable organization which will treat the issues on case-by-case bases.

At this point, the charitable organization can address the issue of grant disbursement to those beneficiaries that do not have bank account. Some beneficiaries are also too old or sick that they are not able to participate in the grant registration process. The charitable organization can also improvise other strategies to address such category of beneficiaries.

Also, some beneficiaries need items rather than money while some need to be assisted to manage the grant by investing it in some business ventures. These also, require other strategies and the charitable organization will have to improvise the strategies.

7 Conclusion

An approach for managing the registration of beneficiaries for a grant is presented. The work also presented the mechanism for the disbursement of the grant to the beneficiaries. The emphases in this paper is development of a mechanism that will organize the beneficiaries in clusters during grant registration process under the coordination of different charitable organizations spread across the nation. The disbursement of the grant is also implemented based on the cluster framework. In this paper, about four different functions are identified for the mechanism and the process flow and algorithm for each of the functions are presented.

References

1. Salamon, L. M. (Ed.). (2014). *New frontiers of philanthropy: a guide to the new tools and actors reshaping global philanthropy and social investing*. Oxford University Press, USA.
2. Komatsu, T., Deserti, A., Rizzo, F., Celi, M., & Alijani, S. (2016). Social innovation business models: Coping with antagonistic objectives and assets. In *Finance and economy for society: Integrating sustainability*. Emerald Group Publishing Limited.
3. Moraes, C., Daskalopoulou, A., & Szmigin, I. (2020). Understanding individual voluntary giving as a practice: Implications for regional arts organisations in the UK. *Sociology*, *54*(1), 70-88.
4. Coppi, G., & Fast, L. (2019). *Blockchain and distributed ledger technologies in the humanitarian sector*. HPG Commissioned Report.
5. Salamon, L. M. (2015). *The resilient sector revisited: The new challenge to nonprofit America*. Brookings Institution Press.
6. Holt, D., & Littlewood, D. (2015). Identifying, mapping, and monitoring the impact of hybrid firms. *California Management Review*, *57*(3), 107-125.
7. Thornley, B., Clark, C., & Emerson, J. (2014). *The impact investor: Lessons in leadership and strategy for collaborative capitalism*. John Wiley & Sons.
8. Bellucci, M., & Manetti, G. (2017). Facebook as a tool for supporting dialogic accounting? Evidence from large philanthropic foundations in the United States. *Accounting, Auditing & Accountability Journal*.
9. Meyers, D., Alliance, C. F., Bohorquez, J., Cumming, B. F. I. B., Emerton, L., Riva, M., ... & Victurine, R. (2020). Conservation finance: a framework. *Conserv. Finance Allian*, 1-45.
10. Esposito, A., & Besana, A. (2018). US community foundations: Building a generous society in challenging times. *Journal of Nonprofit & Public Sector Marketing*, *30*(2), 200-227.
11. Coppi, G., & Fast, L. (2019). *Blockchain and distributed ledger technologies in the humanitarian sector*. HPG Commissioned Report.
12. Gelb, A., & Metz, A. D. (2018). *Identification revolution: Can digital ID be harnessed for development?*. Brookings Institution Press.
13. Agbenyo, F., Galaa, S. Z., & Abiiro, G. A. (2017). Challenges of the targeting approach to social protection: An assessment of the Ghana Livelihood Empowerment against Poverty Programme in the Wa Municipality of Ghana. *Ghana Journal of Development Studies*, *14*(1), 19-38.
14. Kemal, A. A. (2019). Mobile banking in the government-to-person payment sector for financial inclusion in Pakistan. *Information Technology for Development*, *25*(3), 475-502.
15. Lo-Ciganic, W. H., Huang, J. L., Zhang, H. H., Weiss, J. C., Wu, Y., Kwoh, C. K., ... & Gellad, W. F. (2019). Evaluation of machine-learning algorithms for predicting opioid overdose risk among medicare beneficiaries with opioid prescriptions. *JAMA network open*, *2*(3), e190968-e190968.
16. Mehrotra, S., & Verma, S. (2015). An assessment approach for enhancing the organizational performance of social enterprises in India. *Journal of Entrepreneurship in Emerging Economies*.
17. Aldashev, G., & Navarra, C. (2018). Development NGOs: basic facts. *Annals of public and cooperative economics*, *89*(1), 125-155.
18. Song, E. E. (2022). How Outsourcing Social Services to NGOs Bolsters Political Trust in China: Evidence from Shanghai. *Chinese Political Science Review*, 1-27.
19. Cordery, C. J., Sim, D., & van Zijl, T. (2017). Differentiated regulation: The case of charities. *Accounting & Finance*, *57*(1), 131-164.
20. Mbuthia, L. W., & Muturi, J. (2021). INFLUENCE OF STRUCTURE ON THE FINANCING STRATEGY OF NON-

GOVERNMENTAL ORGANIZATIONS:(THE CASE OF VSO JITOLEE IN KENYA). *International Research Journal of Business and Strategic Management*, *2*(1).

21. Desie, S., & Ismail, M. O. (2017). Accountability to affected populations: Somalia nutrition cluster experiences. *Field Exchange 56*, 5.
22. Ambrose-Oji, B., Lawrence, A., & Stewart, A. (2015). Community based forest enterprises in Britain: Two organising typologies. *Forest Policy and Economics*, *58*, 65-74.
23. Sharma, E., & Morwitz, V. G. (2016). Saving the masses: The impact of perceived efficacy on charitable giving to single vs. multiple beneficiaries. *Organizational Behavior and Human Decision Processes*, *135*, 45-54.