TOWARDS BETTER CONTROL OVER THE DISTRIBUTION OF
SUBSIDIZED COMMODITIES

Osman Ibrahim
Faculty of Informatics & Computer Science, British University in Egypt (BUE),
El Shourouk City, Egypt, Osman.ibrahim@bue.ed.eg

ABSTRACT

This paper presents a mobile based solution as an emerging direction in e-Government to
tackle the problem of fairly and transparently distributing subsidized food. The proposed
solution provides visibility, and control over the distribution process. The solution and its
operational process introduce a better mechanism for leakage to black market which is
common with traditional means of subsidy distribution. This mechanism improves the
effectiveness and efficiency of the distribution process and ensures high usability Thus, the
needs and competencies of the beneficiaries of the subsidies distribution system are met. The
solution also improves the delivery of such an important government service to benefit
citizens. Analysis, design, and implementation of such a solution are introduced too.

Keywords: m-Government, Subsidy distribution systems, Mobile-based solutions, System
development using UML.

I. INTRODUCTION

 Governments everywhere provide consumption subsidies in a number of ways: by
providing use of government assets, property, or services at lower than the cost of provision,
or by providing economic incentives to purchase or use such goods [1]. In Egypt, traditional
means for the distribution of food subsidies suffer from many shortcomings such as: unfair
distribution of goods, lengthy distribution process with many leakage opportunities along the
way. One of the conclusions in a recent report published by the World Bank [2] was “If
leakages are eliminated and coverage is narrowed, the government of Egypt could save up to
73 percent of the cost of food subsidies”.

Consequently, better mechanisms for subsidies distributions are needed in order to
improve the effectiveness and efficiency of the distribution process and to ensure high
usability that meets the needs and competencies of the typical stakeholders of the subsidies
distribution system.

The growing rate of subsidy beneficiaries in Egypt has made the development of a fair
and efficient subsidies distribution system a real challenge [3]. Moreover, a major sector in
the beneficiary population suffers from technology-illiteracy that renders current
technological solutions for subsidies management, e.g. smart cards inadequate to scale-up to the magnitude of the problem. Some of these problems are reported in [4]

A. Short comings of the current Subsidy Model

In specific, the current subsidizing model suffers from the following main limitations:

1) The subsidizing entity is isolated from the control of the distribution activity. This isolation leaves the door wide open for corrupted merchants to sell the subsidized goods in the black market.

2) The distribution process is a chain with the subsidizing entity at one end and the target consumers at the other. Along this chain, leakage may cause subsidized goods to not reach eligible consumers.

3) The subsidizing system is complex to be used by various stakeholders. This complexity reflects on the usability of the system by the subsidizing entity as well as the consumer.

4) It is impossible to effectively and efficiently track and report accurate data regarding the current status of the subsidized goods. This reduces the effectiveness of the distribution process, and leaves the subsidizing entity with uncertainty to plan future demand.

B. Our solution

Mobile technologies have deeply penetrated into the typical life-style of the Egyptian culture at its all population sectors and levels. According to the ICT indicators in brief of Egypt Ministry of Communications and Information Technology released February 2011 [5], mobile penetration reached around 92% of the population with 27% increase from the year before. This means that mobile based approaches to tackle this problem will have ground success factors.

In this paper we present a solution based on mobile technologies to tackle the subsidies distribution in Egypt, and for any developing country in general. Accordingly, the main purpose of our solution is to design, develop, and implement a Prototype to demonstrate a solution for the subsidies distribution problem using simple and popular mobile phones.

Despite the fact that Smart Mobile Phones gained rapid adoption among mainstream consumer segments across markets [6], we chose to design and implement our system so that any java-enabled mobile device is suffice to use the system.

Our solution is a type of Mobile government (m-government) which may be defined as [7] “a strategy and its implementation involving the utilization of all kinds of wireless and mobile technology, services, applications, and devices for improving benefits to the parties involved in e-government including citizens, businesses, and all government units”. The solution provides the software along with the enabling connectivity to avail the following functions.

1) Enable consumers to acquire and obtain their subsidized rations in smooth way using a mobile device regardless of the device’s sophistication (first generation mobile will suffice)

2) Enable Merchants to distribute rations to beneficiaries and have visibility on their accounts using a mobile-based point of sale that hosts domain software and is interfaced to variety of terminals (and possibly systems) such as Smart Cards, Mobile devices, and Back-end database server/servers

3) Enable the subsidizing entity (hereafter is termed 3rd party) to establish, manage, and control subsidy elements including items, merchants, and consumers, and be able to align these elements based on indicators provided by the system.

This solution intends to transform the entire subsidy distribution process into a technology assisted that is usable by end beneficiaries with minimum investment.

The rest of the paper is organized as follows: In section 2, we discuss the main requirements that govern the solution starting from a high level usage scenario that shows the main concept of the solution along with the use case modeling of these requirements.
solution architecture with its different views is given in section 3. Section 4 is dedicated to detailed design summary. Section 5 presents some related design and implementation issues. Section 6 is a discussion that highlights the current status of the solution. We conclude in section 6.

II. SOLUTION KEY REQUIREMENTS

In this section we present the key requirements of the solution. We start by introducing a high level usage scenario that abstracts the main idea of the system and is used to derive the system major requirements. We follow by presenting the key system requirements and constraints. Functional requirements of the system modeled as use cases are discussed next.

A. Basic Usage Scenario

To have some high level idea of how the system will support the subsidy distribution process and lay the ground for comprehending the technical discussion that follows, we briefly describe a usage scenario of our solution when it is up and running. In this scenario we assume that a consumer has her/his own mobile device and is willing to have her/his ration (typically monthly) from a merchant who is previously registered with the system. The consumer has also been registered with the system as well as his monthly quota is known. The scenario takes the following steps which are also shown on Fig. 1 below:

1) Customer sends her/his id code, merchant code, and choices
2) The system verifies the customer’s and the merchant’s codes, and balance availability
3) The system sends a transaction number to the customer and to the merchant
4) Both customer and merchant should approve the transaction in order for the system to commit the transaction

Fig. 1 shows the common case in which we assumes that all consumers have their own mobile devices to post her/his ration request. Our solution takes into consideration a less common case where the consumer does not own a mobile. In this latter case the consumer may use the merchant’s mobile device for the same purpose (of course with a secret PIN).

Figure 1 Basic Usage Scenario

A variation of the device used in executing the scenario is also possible. One possibility is the use of Near Field Communication (NFC) enabled mobile phones [8]. The merchant can also use a PC with a USB mobile modem, a point of sale or his mobile phone. There is still the possibility to use a mobile point of sale too.

B. Requirements Collection and Constraints

Our approach of collecting user requirements adopted a User Centered Design methodology [9] and has sought to understand user needs from the users directly through
observation and user interviews. This user data gathering has focused on the usage of existing (smart card based) and prior (paper based) systems, and their perception of the important requirements of any future systems. This information has then been used to determine the basic requirements and even assisted in designing an interface proposal for our solution.

One main determining factor which governs our solution is minimizing potential sources of corruption and maximizing the added value for users. A set of basic functional and non-functional requirements [10] must be met that represents the heart of the system. We may view some of these requirements as constraints or assumptions in some sense as given by the following sample. We present functional requirements as use cases (to follow shortly).

1) System must be mobile-based
2) Traditional Mobile devices must suffice to interact with the system.
3) A consumer can request all or part of his quota. This is very different from limitations imposed by the existing system (or the system before) where a consumer is required to request all items of his monthly quota.
4) A consumer can request quota from any merchant. This is also very different from limitations imposed by the existing system (or the system before) where a consumer is registered with and tied to one specific merchant.
5) Consumers own java enabled Mobile to run simple application (or otherwise she/he may use the merchant mobile device)
6) Merchant owns java enabled Mobile to run simple application.
7) Customer can respond to simple confirmation messages, and can reply with his pin.

In addition to the above basic scenario and the set of constraints/assumptions that represent very basic mandatory requirements, the following section provides discussion about the functional services expressed as a set of the UML use cases [11, 12]. This set of use cases concretizes the basic scenario, and lists the basic services to be provided by the solution. These services achieve the goals of the actors of the system: the Consumer, Merchant, and the 3rd party. If the system can deliver the services that achieve goals of its actors, then such a system to a large extent meets the requirements of the system. Summarization of these services is given in the next section in the form of the system use cases.

C. Use Case Model

As a whole the system comprises twenty three main use cases and three main actors. Use cases included cover the main functionalities (services) to be provided by the solution to its actors. These use cases are grouped functionally as follows:

1) Consumer use cases: Use Cases that serve consumers.
2) Merchant use cases: Use Cases that serve Merchants
3) 3rd Party use cases: Use Cases that serve the 3rd party.
4) Data Maintenance: Use Cases that provide services to enable building and maintaining the basic data of the system
5) Authentication uses cases: Use Cases that provide authentication and validation services

Please note that Data Maintenance Use Cases are mainly used by the 3rd party (subsidizing entity) actor for building and maintaining the basic data of the system.

Three main actors are identified: Consumer, Merchant, and 3rd party. There were suggestions to consider mobile operator as a fourth actor but this suggestion is ruled out at this stage as mobile operator has no clear role within the current scope except passing different interactions between the three main actors.

Each use case is described using a simple template including commonly used attributes for use cases. In addition to the use case name and actors, the description includes
for each use case, the basic scenario, alternative scenarios (if any), preconditions and post conditions.

Fig. 2 is an example of the Lookup Consumer Information use case description following this template.

<table>
<thead>
<tr>
<th>Lookup Consumer Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
</tr>
<tr>
<td>3rd Party</td>
</tr>
</tbody>
</table>

### Basic Flow of Events

1. The actor enters ID of the Consumer he/she wants to look up and click Search.
2. The system retrieves the Consumer information from the database.
3. The system displays the information about the Consumer such as ID, Name, Status, Subsided Quantity, Location, and the Account Balance Status…etc.
4. The system allows the actor to print out the report.

### Alternative Flows

1. If the Actor wants to see the reports for all the consumers in the database, he/she does not fill in the ID of the Consumer and just clicks Search button.
2. The system displays the information about all Consumers such as ID, Name, Status, Subsided Quantity, Location, and the Account Balance Status…etc.
3. The system allows the actor to filter data by location, status,…etc.
4. The system allows the actor to print out the report.

### Preconditions

1. actor is logged on to the system
2. actor is authorized to record new consumer
3. Consumer already exists in the database

### Post conditions

A report of information about a particular Consumer is displayed /printed.

**Figure 2 Lookup Consumer Information Use Case**

### III. ARCHITECTURAL DESIGN

In this section we discuss the architectural design of our solution from different views [13]. Views to be discussed here are: the logical, style (pattern), and deployment views.
A. Logical View

The system could be simply viewed as 3 main interacting sub-systems linked by a wireless network. The first subsystem is a very simple application hosted by the consumer mobile device. The second is also a very simple application that is hosted by the merchant mobile device (or simple point of sale). Most of the business logic and transactional data are within the back end where the system database resides too. Fig. 3 shows this high level logical view of the system.

![Figure 3 Architecture logical view](image)

B. System Architectural Pattern

Our solution architecture adopts the well-known layered architectural pattern (style) to organize the solution into a number of layers [14- 17]. Layered architecture is chosen in order to ensure flexibility and modularity. Organizing software architecture into layers is a very common architectural style used in various industrial systems [18]. Our solution architecture consists of 5 layers in addition to the user interface layer that we do not have space to cover.

1) Terminal Interface Layer (TIL): This layer handles user inputs from various possible user interfaces (e.g., smart card, desktop terminal, cell phone terminal). This layer also converts user requests into commands that can be processed by the Business Logic Layer (BLL). Finally, this layer is also responsible for appropriately formatting the output for display on various possible channels. In the current design, two presentation layers do exist. The first layer resides within thin clients running normal web browser for subsidy control at 3rd party (subsidizing entity) side. The second presentation layer is implemented with java to run on merchant and consumer mobile phones.

2) Business Logic Layer (BLL): This layer has the following responsibilities:

   a) Processes the business logic commands passed from the TIL layer and composes security requests that will be passed to the Security Layer (SL).

   b) Interprets the server responses for the TIL layer.

   c) Provides support for application specific business processes and the enforcement of business and data integrity rules.

3) Translation Layer (TL): This layer is responsible of converting the high level commands which are created in the presentation and business logic layers into a stream of byte arrays to be sent to the communication layer.
4) **Security Layer (SL):** This layer performs the basic security operations needed in order to keep all commands and their associated data secure. This layer performs encryption/decryption operations, and it filters out unauthorized responses.

5) **Wireless Communication Interface Layer (WCIL):** This layer is responsible for preparing encrypted commands and data passed by the SL for transmission according to a specific data transmission protocol used by the underlying wireless network backbone.

### C. Deployment View

The Deployment View of the solution describes the likely physical network and hardware configurations on which the system will be deployed. This view has been informed by a number of the architecturally significant decisions such as: Centralized Database, Layered Architecture Style, Wireless Network, and Mobile Technology.

The deployment view of the system is represented as UML deployment diagram (not shown for size limitation). Deployment diagrams are large-scale instance diagrams and are important for modeling software architectures in UML [19]. Our deployment view includes four UML nodes where each node is connected to the other nodes (mostly) through a wireless network. Each node hosts a subsystem of the solution. The first three subsystems encapsulate the services provided to our principal actors; namely: Customer, Merchant, and the 3rd Party.

The fourth subsystem encapsulates the central database that holds subsidy and subsidy-related information along with a large part of the business logic. We may refer to this part as the solution server. Please note that the first and the second nodes abstract mobile devices processing elements that will host software to be used by Consumers and Merchants. The 3rd party may use thin clients or mobile device too.

### IV. DETAILED DESIGN SUMMARY

Our solution is designed using the common practiced Object Oriented approach where main system entities are perceived as objects. In this respect the detailed design took the following summarized steps:

1) Elaborate business scenarios into more technical (concrete) scenarios.
2) Reformat use cases identified into a set of business operations that represent the minimum system functional commitments.
3) Elaborate business operations identified in 2 above into design elements.
4) Parallel to steps 1-3, develop the class diagram assisted by other UML models (e.g. Sequence and activity diagrams).
5) Decide on persistence and hence lay out the database design
6) Develop user interface

In this section we discuss some of these detailed design activities. For space limitation we are unable to give full details. Details given here are intended for shedding light on the system as a whole. In subsequent papers we dedicate greater details that encompass these activities.

#### A. Typical Business Scenario From a design perspective

In this design activity we elaborate the main business oriented usage scenario into system-wide technical interaction. Compare the following scenario (from a design perspective) to the business scenario shown on Fig. 1.

The consumer runs the application from her/his mobile device and connects to the back end.
1) The system prompts the consumer to enter ID and PIN code to login, if it’s the first time to use the system, the consumer is directed to a registration form.
2) The consumer’s information is validated at the server side.
3) Consumer sends a request asking for her/his monthly ration.
4) The server responds at the Merchant side by sending a list of consumer names who wants their rations
5) The Merchant selects a consumer from the list to service by sending to the server the consumer and his ration data including consumer ID, quota type, and details of required quota items.
6) The server responds with a summary of the interaction to the consumer that includes the consumer and merchant identities as well as quota details (including items, ration’s month and year, and the total to be paid by the consumer)

When the consumer commits this information, the database is updated with this finalized transaction.

B. Mapping key Business Operations into Design Elements

Here use cases are analyzed, and grouped into a set of key business operations (Command is the corresponding technical term used later) that the solution is committed to provide. Identified business operations are then mapped into design elements using more than one model. For example:

1) UML Sequence and activity diagrams are employed in order to capture timing and sequencing of the various commands and activities for each system operation.
2) Low level Behavior modeling where each operation is specified in terms of: Command Trigger, Server Actions, and Command Response.

Fig. 4 depicts a high-level grouping of the key business operations in the proposed design.

As shown in the figure, business operations are grouped into four main categorizes:

- User Account Management operations: These operations deal with the registration of the customers and merchants to the system. The operations also cover various activities related to the management of the created account, such as the change of the user PIN.
- Subsidize-Out operations: These operations present the core of the solution. They cover the activity of claiming subsidized items from a merchant. The Subsidize-Out operations include the customer request, merchant approval, and customer approval processes.
- Management and Control operations: these operations focus on the activities related to the supply of subsidized items to the merchants and monitoring of the status of these items throughout their life-cycle from their arrival to the merchant until they are claimed by consumers.
• Enquiry-related operations: these operations cover all relevant enquiries that are useful for various stakeholders in the system. This includes, for example, operations about enquiries related to merchant information, customer available balance, list of customer transactions, etc.

Dealing with each of the leaf business operations as a command, we show on Fig. 5 how the command is triggered, and how it climbs up or down the solution layers, and how each layer manipulates the command until its success or failure. The figure provides an example of how one of the business operations Purchase-Customer request traverses the different solution layers from the moment it is triggered and how it is manipulated along different layers. The role of each layer in processing the command is apparent.

C. Class Diagram

Our class diagram encompasses 2 main categories of classes: business domain classes and utility classes. A major part of the business domain classes are persistent classes too. These include classes such as Items, ration quota, consumer, merchant, and other related classes e.g. organizations, governorates, cities, and subscription. This set of classes forms the core of the system database. No space to any further details of this group of classes.

Utility classes span all layers of the system architecture to provide system wide support functionalities. These can further be divided into the following subgroups:

1) **General Utility Classes:**

These classes model abstractions such as user request/response, Translation class (responsible of converting command data into byte array to be sent to the server), and encoding/decoding classes

2) **Client Interaction Classes**

In addition to the pure user interface classes, the solution includes a set of other client classes that works just below the user interface classes. We call this category of classes Form Action classes. The main responsibility of this class is to instantiate a command based on user entry and sends the created command to the translation layer or receive the response of previously sent command and handle errors if found. This set of classes is designed in a way to implement business operations as discussed earlier and account for any future additions to these operations. Currently there is a number of classes of this type equals to the number of currently implemented operations.

3) **Server Interaction Classes**

We discuss here one main class that shields the server database called Database Wrapper. Any interaction with the back end database is done through this class. It contains a set of handlers that employ Java Database Connectivity (JDBC) technology [20] to connect and interact with the database for different database operations. Each handler is a method that communicates with a database stored procedure to perform a specific task in the database side. These tasks implement business operations that require database access. Each handler gets parameters required by the database stored procedure from the client command and call the appropriate stored procedure with these parameter.
**Command Triggering**

| Input from TIL | A customer-purchase command code with parameters: Merchant ID, Item list |
| BLL Action | Append Mobile phone, Customer PIN and compose the command |
| Output to SL | A request to encode the composite command |

**Server Actions**

| If the submitted data is valid |
| Generates a new transaction number |
| Store the Transaction information in the transaction database with transaction status marked ‘not-confirmed’. |
| Send the transaction number to Customer. |
| Else |
| Send error code to Customer |
| End if |

**Customer Response**

| Input from SL | Server response containing transaction number or error code |
| BLL Action | Store the purchase transaction number if found, On error, interpret the error code |
| Output to TIL | Trigger the Transaction-Started event, or Trigger the Transaction-Rejected-Error event |

**Merchant Response (Conditional)**

| Input from SL | Decrypted server response containing transaction information |
| BLL Action | Add the purchase transaction information to the list of pending transactions |
| Output to TIL | Trigger the Pending-Transaction-List-Updated event |

*Figure 5 Life Time of the Purchase-Customer Command*
V. DESIGN AND IMPLEMENTATION ISSUES.

In this section we briefly discuss some issues related to both design and implementation of the solution. This discussion sheds some light on our solution specifics and justifies some technical decision we made.

A. Design issues.

Heavy Use of Inheritance: Commands, Responses, and Client Action classes are examples of classes that heavily use inheritance where an abstract class is defined and a set of derived classes that are used to specialize different commands, responses, and form actions are defined. Those derived classes each corresponds to one of the different command types to implement business operations. For example, the Register Command class has the responsibility of registering the merchant information to the system and storing the information on merchant phone. The Query Quota command is used to query client ration. Purchase command is used to order the client quota. Get Pending Invoice command has the responsibility as implied by the name.

The same applies to the Response and Action Form classes where a base abstract class is defined and a set of other derived (concrete) classes inherits from this base class. In addition to being a necessity as we perceive it in our case, heavy use of inheritance in our solution agrees with the common Java programming practice [21]; the language we used to implement the solution.

Assigning responsibility to classes: in finalizing the class diagram starting from the domain model of the problem concepts, we have employed UML sequence diagrams along with the basic design patterns. This way we could concretize use case (business operation) interactions in a systematic way and hence assigning responsibilities to different classes. This leads to reduced coupling and increased cohesion of the classes [22]. We also used UML activity diagram to flesh out algorithmic logic for some of the classes operations.

B. Implementation issues

Java is used as the programming language for coding the system. Java is used since our team has experience working with; in addition it provides all we need to implement the solution. Java is also selected for its native support to network communication. As the reader may have noticed, an important part of the system relies on network communication to send commands and receive responses.

Within the context of Java, we had to make a decision whether to use Java Sockets or Java Remote Method Invocation (RMI) to deal with the network part of our solution. Despite the fact that sockets programming is more tedious, two factors enforced us to use sockets. First, we already have reusable java components written using java sockets that we have validated and used in other projects. These reusable components cover most of our needs. Second, these components (luckily) use bytes across the network which makes it reusable with other clients such as .NET and C++ if future needs arise.

For the back end implementation we are employing Oracle products to host data, business logic, and the part of the application used by the subsidizing entity. Oracle database
engine (Oracle 10g) hosts the solution persistent data briefly discussed earlier. The database engine also hosts stored procedures and database triggers that implement the core of the business logic. The part of our application that should be used by the subsidizing entity for maintaining the data and extracting indicators is implemented using Oracle development tools too.

Oracle Database was selected for scalability, security, availability and other plus features that Oracle database engine is known to have. In addition, Oracle offers development platform for Internet and traditional development that we used in implementing our solution.

VI. DISCUSSION

Our solution has passed through typical phases of development: planning, analysis, design, coding, and testing interleaved in an iterative way as the recent best practices recommend. We can say that what we have today is a prototype that has been demoed to relevant stakeholders and communities. We received many valuable comments and advices. Many of these comments are business oriented and few are technical.

Business comments are centered on extending the solution to cover different spans of subsidies such as subsidies provided by NGOs. Many comments also suggested incorporating other forms of subsidies in our solution such as gasoline.

Examples of technical comments include more thorough testing and validation, security issues, performance concerns, and solution usability require further attention.

We do not claim that the system is ready to be deployed now to a real working environment. Still more work is required to productize the solution. Luckily we have thought of many of the comments and questions raised during the demos and we have taken into consideration while designing the system both on the level of the clients and the back end that it can accommodate extensions in different dimensions. The current version of the solution supports other donors of subsidy such as NGOs not only governments. It is also true that the solution still requires more testing especially performance, security, and usability testing. Concurrency and synchronization issues require more analysis.

Currently there is undergoing discussions and negotiations with the key stakeholders of the subsidizing entities to finish the above issues and productize the prototype. This will be followed by deploying the system in a metropolitan area for experimenting with the solution. We will report later about the emerging status of our solution.

VII. CONCLUSION

In this paper we presented a complete high level picture of the requirements, design, and implementation of consumption subsidy solution that provides automated support for the subsidy distribution process using mobile technology. The application can run on any java-enabled mobile and interacts with a back end server maintained by the subsidizing entity(s). The system provides the level of control and visibility over subsidy distribution process from the side of the subsidizing entity (typically governments) that helps in minimizing leaks to black market. In addition, it makes the process easy for consumers and merchants. For
consumers it provides her/him with new services of requesting his ration from any registered merchant and he no longer tied to a specific one. Also consumers are not required to ration all (monthly) quota at the same time.

A prototype is developed, tested and demoed to stakeholders and it is now in the process of being productized. A negotiation is undergoing to deploy the system on a metropolitan area for experimenting with the solution.

VIII. Acknowledgment

This work benefited from the valuable contributions of the team members of the project C2/S1/145 of EU - Egypt Innovation Fund sponsored and funded in part by EU. The content of this paper is the sole responsibility of the author and can under no circumstances be regarded as reflecting the position of the European Union.

REFERENCES
[4] Magdy Amer “Using Smart Cards to Control the Distribution of the Subsidized Food in Egypt” 1st International Conference on Innovation and Entrepreneurship, Cairo, Egypt, 23 – 24 April 2012