

# Template Mobile Applications for Social and Educational Development

Anita Wasilewska, Jennifer L. Wong

Computer Science Department, Stony Brook University, Stony Brook, NY 11794

**Abstract**—There are over 3.5 billion mobile phones in the world and they are proliferating at astounding rates across socioeconomic and cultural boundaries. They also provide unprecedented opportunities for enabling social impact and technical activism. To most of the people in informal economies and immigrant communities, the mobile phone is the dominant computing resource which they have access to with limited income to support. In the paper we present a notion of template applications for mobile phones. By developing useful applications as templates for these communities which operate on the limited resources of their existing mobile phone, we gain the benefits of language, community, and country portability and customization. We present the template philosophy and illustrate the concept with two developed template applications and explore their transformation into other application domains.

## I. INTRODUCTION

Computer Science research for global and social development can be carried out at three levels of the economy on opposite ends of the spectrum. At the top of the spectrum are developed countries with access to state-of-the-art technology. Research here focuses on projects for using available resources to aid disabled members of the society. For example, the development of web access techniques for the blind [1]. At the other end of the spectrum are underdeveloped countries with limited, if any, technology. The most dominant form of technology in these places is typically mobile phones. In the middle, we have developing, lower-class, and immigrant communities within developed countries which have access to technology which is not necessarily state-of-the-art. While it is socially important and worthy of research to focus on innovation at the top end of the spectrum, we believe that focusing on the bottom of the spectrum, on informal economies in developing countries, poses larger computer science research challenges, mainly due to the technology and the knowledge of the users being at a much lower level, and will have much greater impact on global social development. If these challenges can be addressed at this level, then the solutions, along with improved features, can be easily adapted to advanced informal economies with more technology resources and knowledge, such as immigrant communities in highly developed countries [2], [3].

Incubator and micro-loan programs have been the predominant form of developmental and technical support for informal economies and lower-class communities. While many of these programs have proven to be beneficial they face two major drawbacks: typically there is no evaluation or feedback information obtained on the effectiveness or impact of the programs; and these programs operate for limited time periods

only and provide no sustainability within the community. Incubators typically bring technology, such as computers, to the community and teach members to use them. However when the incubator term expires the technology often leaves with it. By focusing on developing special mobile applications for existing mobile technology, such as mobile phones, the members will continue to have access to the devices after assistance has come and gone.

In this work, we propose an ideology for the development of template mobile applications. Template applications focus on language-independent interfaces to enable quick customization and portability to different countries and communities and all levels of educational development. Ideologically, the focus is on modularity and simplicity of the developed programs. The template applications are initially based on applications developed for a particular domain and community. Throughout the design process, additional considerations must be made to structure, implement, and design the interface to be template capable. In the next section we discuss the goals, concepts, and challenges of developing template applications for mobile phones. In Section II, we present two template applications and their transformations to illustrate the concepts. Before concluding, we present the challenges of working with mobile phones in the mentioned communities and the challenges in developing for mobile technology.

## II. TEMPLATE IDEOLOGY

We propose the creation of template mobile phone applications with three main goals in mind. The first goal is to develop new applications which will encourage economic growth in informal economies and new immigrant communities in developed countries, to assist incubator and micro-loan programs in developed countries (such as France [4] and Harlem, NY [5]) to support their businesses, and to assist disabled and elderly with programs which are more simplistic in use. Because the target audience of the first goal is expansive across age, countries, and cultures, our second goal is to create the applications as language independent applications. One excellent example is Africa, where there are 2,000 different languages spoken and these vary from the official and written languages. By creating the application to be language independent, each developed application can be quickly extended across all of these types of boundaries. The third goal is to address the functionality and interface of the applications to the specific needs of the users. Often applications are developed for a large target audience and



Fig. 1. Screenshots of the dictionary application.

therefore try to address the needs of the group as a whole. By doing so, often many functionalities of an application are useless to the user. When dealing with mobile phones which are a few generations old, as used by these communities, there are lack of hardware resources to support unused functionality. We believe that it is crucial to ensure that the applications reflect the direct needs of the users, even at the cost of small customizations.

A template application is an application which can easily be modified from one domain to another. This modification should mainly be performed on the language independent interface, on small underlying computation functions, or through database information replacement. A template application for mobile phones should be programmed in an efficient and compact fashion, should not require more computation resources than the mobile phone provides, and should be structured in such a way that customization is easily performed. The main screen (Figure 5) of our financial application template is a good and simple example of what is meant by a language independent interface. In this application, we use in and out arrows pointing to the money symbol to represent income and expenses categories, and the “bucket full of money symbol to represent the total income category. In the generic case, we use a stack of green 1000’s to represent money. Depending on the country, the color or the amount of each bill should be tailored. For example, in our initial application built for Senegal, the stack of bills had the symbol for Senegalese currency, only when converting the application to a template did we replace it for a more universal symbol. We do introduce written text into the interface to help educate the user if they are illiterate or semi-literate. The key is to strive to develop the application as language independent as possible, replacing words with symbols wherever possible.

What does it mean to develop a template? As we can see from the above example a template must have its source in a concrete application which is developed for a concrete need. Not all mobile applications can become a template. For an application to be able to be transformed into a template application it has to be developed in such a way that its user interface and basic functions can be made universal and also can be easily transformed to another domain. For example, as in a case of the financial applications (Section III-B) and its transformations as Food Planner and Time Planner. While thinking about and designing new applications the developer must structure their interfaces and functions with templates in mind; he/she must brainstorm from the beginning about what

new applications can be developed in template capacity.

In order to develop templates, there two main design and technology challenges. The first is to develop applications for low to middle level mobile phones which are at least a few generations older than the current technology. These phones are typically what are accessible, cost effective, and technology appropriate for the users. Providing the users with state-of-the-art mobile technology is not only cost prohibitive but is also likely to be unsupported by the mobile carrier-/infrastructure. For developers it is difficult to design, think, and program under these limitations due to their knowledge of current technology. To get past this barrier, developers need to transform their thinking away from high-level language programming principles towards highly efficient and compact code, go “back-to-basics” in programming. Programs need to be organized in a less structured programming style, but remain highly structured to aid in customization and portability of the application to other domains. For example, efficient use of macro definitions should be used to allow all customizable aspects of the program to be done uniformly. Along with this a strong compiler is required to ensure maximum code compaction to bring the maximum amount of functionality to the limited resources of the mobile phones. Also, it is a challenge to create applications which address the various limitations of the different models and platforms of the mobile phones in different countries. Therefore, the easier it is to quickly and efficiently modify the template application to work on different platforms, the faster the application can get to the user. One additional aspect that must be considered when designing the applications is the level of capabilities of the members in the communities to customize, train the users on the operation of the applications, and provide support for the template applications. In order for these templates to reach the users, on-the-ground workers from the communities who understand the culture and the language must be trained. It is crucial for the success of the applications. Therefore, all the programming must be done in such a way that someone with minimal programming experience and possibly without true understanding of the entire application can modify the program to address a specific user’s needs.

The second major challenge is to create and maintain a library of images for customizing the applications to the needs of the user. When developing template applications, presented in the next section, we spent a large amount of time determining and creating images to properly represent concept ideas. The difficulty additionally increases considering how the images would be interpreted by other cultures, countries, and educational levels. For example, in a financial application what images would one create to represent the concepts of balance, income, and savings goal?

Once an application is developed for a targeted user with specific needs, we view the application as a template if at least the interface can be ported to another domain. The application can be a template if it can be transformed through a subset of the following five template transformations: (I) visual interface modification of template, (II) visual interface

modification with basic function simplification, (III) visual interface modification with basic function manipulation, (IV) visual interface modification with basic function extension, and (V) personalization/customization. The first transformation is solely changing the images in the interface. This type of transformation keeps all functionality of the original application and changes the domain only through the images. The second transformation is when the not only the images need swapping, but portions of the applications functionality are removed or simplified. In this case, often whole components of the program are removed. When the functionality must be modified instead, this is the third transformation. The most difficult of the transformations, from a programming and development perspective, is the basic function extension. In this case the current functionality of the application remains intact but must also be enhanced to implement an additional functionality. Lastly, we have personalization or customization. This type of transformation can be a combination of the other transformations, in the sense that depending on the user's needs for the application any combination of image swapping and functionality modifications could be required. One benefit of the templates, is that with each template transformation of an application, the new application can be used as a future template. We illustrate this with the financial account book in Section III-B.

### III. TEMPLATE EXAMPLES

In this section, we present examples of the template transformations for two developed applications. The first application is an educational dictionary (or flash card) application. The dictionary is a simple application with an intuitive interface and operation. Because of its ease of understanding we use the application to show straight-forward illustrate the first two template transformations (I) and (II). The dictionary application was originally designed to help promote and encourage early language learning in developing countries. In most developing countries there are different official and spoken languages, which makes language a barrier for many young children in schools. This is only one of the factors that results in many children completing barely a few years of primary schooling and ultimately become illiterate adults.

The second application is a financial account book. Initially designed to assist small business owners in informal economies, the goal of the application is to instill awareness of profit, business expenses, and savings goals. The idea is that by bringing financial awareness to the business owners, the businesses would increase their chances of prospering, and therefore prospering the local economy. The nature of financial applications is to be complex in operation. However, the structure and the interface of the financial application hides the complexity of operation with a simple intuitive interface. While the account book is a more involved application development-wise it is an excellent candidate for illustrating the template transformations. More details about the design process, implementation and functionality of these two applications can be found in [6].



(a) Main Screen

(b) Description Screen

Fig. 2. Screenshots of the African Countries application.

#### A. Educational Dictionary

The dictionary application was originally developed to teach French to the children in Senegal [6]. In essence it is an educational flash card application which displays pictures to the user. When an item is selected the spelling of the word is displayed and an audio file pronouncing the word is played. In addition, a definition of the selected word or a sentence using the word is provided through a button on the word page. The application shuffles the order of the items on each load to encourage the user to learn different words, similar to shuffling flash cards.

Figure 1(a) illustrates the main selection screen of the application. Each of the words or items in the dictionary can be tailored to represent important or appropriate words for the particular age group. When a word/title is selected from the main screen, the screen shown in Figure 1(b) is presented. This screen displays a larger version of the picture and provides two buttons: details and play. When the play button is pressed, an audio file pronouncing the word is played. The details button presents additional information about the item. This field can be customized to display any additional information which is relevant to the education of the user (Figure 1(c)).

The dictionary program is structured in such a way that the user can change the elements of the program through a single database file without modifying the operation code. An xml file which contains the details: the word, the image file name, corresponding audio file, and the details text is used to encourage flexibility and reusability of the application.

1) *States Application: Transformation (I) & (V)*: The States application is a simple transformation of the dictionary application to educate the user about the states in the United States of America. This is an example of template transformation (I). By replacing the words/items with images of each state the dictionary has become a more generalized education tool. Figure 3(a) shows a screenshot of the main page. When a state is selected, the details and play buttons remain. When the play button is pressed, the name of the state is pronounced. The details button displays information about the state to the user (Figure 3(b) and 3(c)). The user can select the next or previous state in alphabetical order or a random state by selecting the



Fig. 3. Screenshots of the States application.

left or right arrow from the right toggle button, as shown in Figure 3(d). This application is a simple transformation. Only the images and the data information has been modified. The same type of transformation can be performed to change the required language or relevant information for the student/user.

Another example of this transformation is to change the states application into an application displaying the countries in Africa. Figure 2 illustrates this modification. Personalization is easily performed by substituting a database file. In addition, notice that all the operation buttons try to use images to represent the functionality. The bottom menu buttons can also be modified to show in any language.

2) *States II Application: Transformation (III)*: States II is an extension of the dictionary application and illustration of the template transformation (III). The main screen of States II is identical to the original dictionary and states programs. While viewing the list of states, there is a popup button at the bottom right that allows you to either show more details about the highlighted state or search through the list for a particular state. This is an extension of the original functionality and requires additional programming. However, by introducing additional functionalities the program can target the educational needs of older children. When choosing to search for a state, you type in the state you wish to find (Figure 4(b)). If it yields a result, you can select it to view more details about that state.

After choosing a state from the list, the screen with more details about that state appears. Just like in Dictionary, there is a "play" button that plays a sound file that says the name of the state. In addition a "More Info" button has been added and when selected, will briefly show a picture of the state capital as well as a sound file that says the name of the capital (Figure 4(a)). The use of capitals instead of details about the state in this version exhibits the versatility of the program. Also on this screen, at the bottom right is a button that pops up to allow the user to go to the next state, previous state, random state, or details that show the state capital. These simple modifications increase the usability of the program and extend the functionalities to allow the program to be used as a template for more advanced applications again.



Fig. 4. Screenshots of the States II application.

## B. Financial Account Book

The account book is an application that was created to help manage a user's income and spending. The main functions are: tracking the amount of sales of different goods (*income*), tracking the amount of spending on items and activities (*expenses*), quantifying the profit or loss of businesses (*total*), and providing a visual interface to track savings or progress of profits towards a long-term goal (*goal*).

Figure 5 illustrates the main screen of the application. On the main screen, four of the main functions are represented: *income*, *expenses (outcome)*, *total*, and *goal*. When *income* or *outcome* is selected the user is presented with a screen which allows them to enter the amount of money which was earned or spent in a particular category (Figures 6(a) and 6(c)). The *goal* screen, shown in Figure 6(e), presents the user with three items to choose from. These items represent long-term purchasing goals. The cost of the selected goal item is entered by the user on this screen. Figure 6(g) shows the total screen, it displays the current balance of the account to the users and illustrates the progress towards the selected goal item.

The account book is a complete application with versatility as its main feature. It has been designed to operate as any kind of resource management application through the template transformations mentioned in Section II. These transformations are illustrated in the following three applications: Ap-

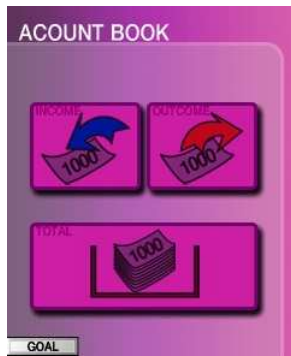


Fig. 5. Main Screen of Account Book Application.

parel Application, Meal Planner Application and Time Planner Application. Each application shows how the account book can be simply transformed.

#### C. Apparel Application: Transformation (I) & (V)

The Apparel application is a direct transformation of the Account Book through transformation (I): visual interface modification of template and (V) personalization. The principle of the application is identical to the original account book - manage the user's finances. The Apparel application is a simple example illustrating how the application can be customized to the particular needs of a business through a direct image replacement. In this case, the images for the items which the business sells, the expenses, and the motivational goal have to be changed to reflect an apparel business. In this case, no functions of the application are changed.

The main page of the application is identical to the account book, as shown in Figure 5. Variations of the *income* screen are required to represent the products sold in the apparel business. We show an example in Figures 6(a) and 6(b) of the changes from the original account book to the apparel business. A similar modification can be seen for the expenses or *outcome* screen in Figures 6(c) and 6(d). Figure 6(d) shows how the different expenditures can be changed. For instance, "Taxi" has been changed to "Transportation" and "Electricity" has been changed to "Heat". These customizations can be performed specifically to the needs of the user.

From the main menu, the user can set a goal for what they want to work towards, by selecting the goal button at the bottom left. Again, the functionality has remained the same between the Apparel application and Account Book. Figure 6(e) shows the original Account Book with three goals. In Figure 6(f) we replace these goals with various types of shoes. The transformation here is also the replacement of images with images of motivational items for the user. Each of the goals can be personalized to the user through image replacement.

When the total button is selected from the main screen, the screen shown in Figure 6(e) is displayed. The screen depicts how much money the user has accumulated as well as a progress bar towards the goal the user has set. This screen for the apparel business is identical to the original Account Book in terms of functionality. Figure 6(g) shows the original Account Book with a pair of shoes and a red "X" showing

that the user does not have enough money to buy that item yet. In Figure 6(h), we can see that the item is changed to reflect the goal a different goal.

#### D. Meal Planner: Transformation (II) & (IV)

Meal Planner is a simple to use application which is designed to educate and help the user to keep track of his consumption of the four food groups. It is a great example of how a versatile application such as an Account Book can be very easily transformed into another topic. This transformation involves both modification of the images, simplification of the application by removing functionality and extension of a function.

In Figure 5, we can see the main menu of the original Account Book. The Meal Planner application is about measuring how much food one wants to eat for a given time period. The application is simplified to contain three of the original four functions of the program: eating *income*, the total food we have eaten *total* and the amount of food we want to eat *goal*. First, we change the images for these different functions/buttons to cater to our food scheme - this is how simple it is to change the objective of the application. In this application we have used a plate and silverware to represent *income* and *total* as shown in Figure 7(a). Since the outcome function is no longer needed, the button has been removed from the main screen.

When the *income* button is selected, the different food group categories are displayed (Figure 7(b)). In Figure 6(a) of the original account book, five different categories were presented. In the meal planner since there are only four food groups, meats, breads, fruits and vegetables, the fifth option is removed. This shows how easy it is to manipulate the application to deal with the amount of items or categories desired.

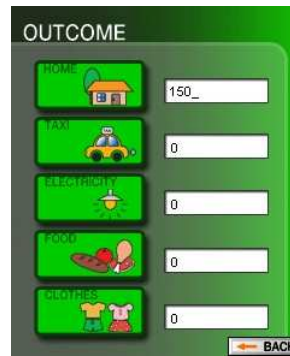
After selecting a type of food to add into the system, representing the food the user has eaten, we are presented with one of the screens. An example is shown in Figure 7(d). We can see that this screen has not changed much in the Meal Planner as opposed to the Account Book shown in Figure 7(c). The only difference is the images used to represent the scheme that is used. Figure 7(c) uses a single bill to represent the amount of money you are adding into the system, as well as a stack of bills to represent the stack that the money is being added to (the total amount). In Figure 7(d), we have changed these bill images to a particular food group. There is a loaf of bread to represent a single serving of fruit. Underneath is a group of grains representing the total amount of grains consumed by the user. The images on this page are altered depending on the selected food group the user has chosen from the income menu. The functionality of the income page has now changed to include four different totals, one for each food group, instead of the single total which was used in the account book. This extension of functionality is basic since it involves only duplication of the total amount of money. The major difference between the two applications is that the Account Book only kept track of one amount - the users' money. Meal



(a) Account Book: Income Screen



(b) Apparel: Income Screen



(c) Account Book: Outcome Screen



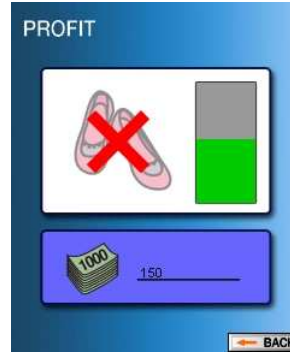
(d) Apparel: Outcome Screen



(e) Account Book: Goal Screen



(f) Apparel: Goal Screen

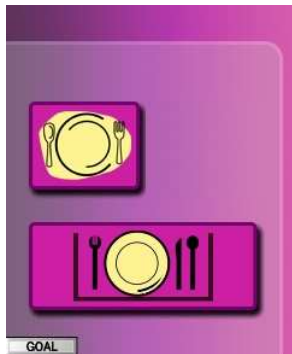


(g) Account Book: Total Screen



(h) Apparel: Total Screen

Fig. 6. Screenshots of the Apparel Application.



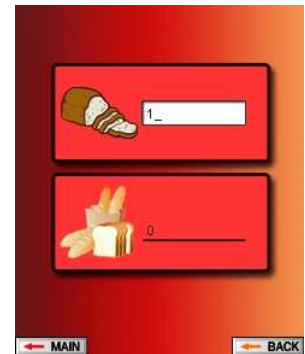
(a) Main Screen



(b) Income Screen



(c) Account Book: Income Input Screen



(d) Income Input: Grains/Bread Screen

Fig. 7. Screenshots of the Meal Planner application.

Planner has to keep track of four different amounts - the four different food groups.

When selecting the total category from the main page, the total amount of food that has been consumed by the user so far is shown (Figure 8(a)). This screen is different in appearance from the original account book shown in Figure 6(g). However it is in fact very similar. By looking at the information given for one food group, we can see it has an image of the item and a visual bar representing the total amount for that item, as it was at the bottom of the Account Book. It also has a textual number representing the total amount for that item, just like the Account Book, however it is situated to the right of the image and bar. The function that the Account Book uses to keep track of the user's money has simply been duplicated

four times to allow the Meal planner to keep track of its four food groups. The format of the screen has been modified, but the same underlying functionality remains.

Selecting the goal button from the main screen takes us to the goal screen shown in Figure 8(b) where the user can set the amount of each food group he wishes to intake - the goal he wishes to achieve. What has been done on this screen is similar to the total screen in Figure 6(g). The original Account Book kept track of three items as shown in Figure 6(e). In order to allow the user to choose between the four food groups, the function that allows the user to set the goal for an item was duplicated one additional time. The images were changed and the format was shifted to allow room for the fourth food group. The functions stayed the same as in the Account Book.

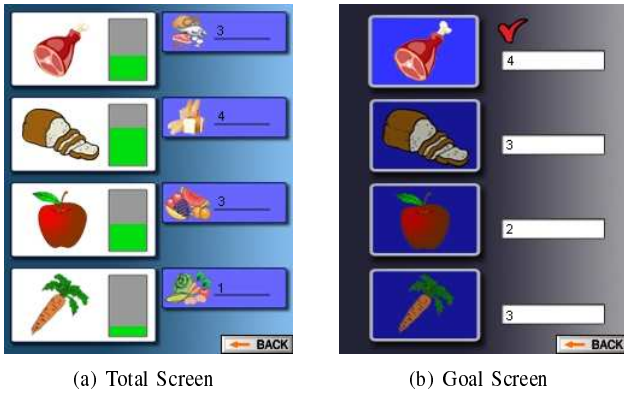


Fig. 8. Screenshots of the Meal Planner application.

This Meal Planner application shows exactly how versatile the Account Book really is, and how easy it is to manipulate it. First, the only thing that needs to be done to change the topic or objective, is to simply change the images throughout the application. Then, as Meal Planner shows, functions as well as buttons can be removed if they are not needed such as the outgoing button and functions from the original Account Book. In addition, certain duplication of functions can be done to account for additional items, such as the four total food groups in place of a single total.

#### E. Time Planner: Transformation (I) and (V)

Time planner is an application that shows how one template transformation can be modified again to implement another function. In this case, Time Planner uses the above Meal Planner as a template. This is also an illustration of the personalized transformation. The meal planner altered the Account Book by swapping images, removing unneeded functions to achieve a new scheme and a new objective, and duplication of existing function. The time planner uses the meal planner and swaps the images to get a weekly, or daily, time planning scheme. This is another example of the type (I) and (V) transformations.

The time planner application is designed to assist the user in monitoring how much time is spent on a particular activity and to determine if he has met his goal for that activity. An example of usage for this application is a parent who wants to track the amount of time his child spends on a particular activity per day. Or it could be use to help a user monitor the amount of exercise performed in a particular category, such as cardio, weights, etc.

The main screen of the time planner, shown in Figure 9(a), is identical to the meal planner with image replacement. By comparing Figure 9(a) and Figure 7(a), we see that the scheme has completely changed by simply changing just two images. The user can input the amount of hours spent on a particular activity *income*, view the total amount of hours spent on a particular activity *total* and set a goal as to how many hours the user wants to spend performing each activity *goal*.

After selecting the calendar button on the main screen, the user picks the activity that has been performed (Figure 9(b)). By comparing with the meal planner in Figure 7(b), we can see

that the format and function have remained the same. There are still four buttons, representing four activities instead of four food groups. This is where the user enters the amount of hours that were spent performing the selected activity. Again, the only thing that has changed from Meal Planner is the images that are used to represent activities instead of food groups. These screens have a field where the user inputs the hours, and a display that shows the total amount of hours the user spent doing that activity in a given time period. When selecting the total button from the main screen, or the button with numbered calendar pages, the user is presented with the total amount of hours spent on each activity, shown in Figure 9(c). It shows an image of the activity as well as a visual bar that shows how many hours were spent towards the user's goal. It also has a textual number showing how many hours were spent doing that activity. Comparing it to the original Account Book, shown in Figure 8(a), we see that the only thing that has changed are the images. Meal Planner duplicates the function used to display the total amount from the original Account Book four times in order to be able to display four different items. The Time Planner application then benefits by using the Meal Planner as a template because now it is possible to display four activities. The goal screen is shown in Figure 9(d). The user can set the amount of hours he wants to spend performing each activity in a given time period. This is identical in functionality to the Meal Planner shown in Figure 8(b). The only change is the set of images to change the application from a food scheme to an activity scheme.

#### IV. CHALLENGES

When developing application templates which are aimed to educate and promote the needs of groups such as developing informal economies, immigrant communities, and the disabled, there are three categories of challenges: social, deployment, and cultural.

Social and deployment challenges are the most critical and difficult to address. The biggest social challenge is identifying and addressing the needs of the people and targeted community. For an outside developer it is often difficult to think in terms of the local, potential users, which can result in the development of applications which are insignificant or have unrealistic or impractical functionality. As with development of any software, interviewing and testing with the users is the key. This can be a difficult task in developing countries due to the lack of technology and educational background, or in immigrant communities where the cultural behavior or trends vary. In addition, language and/or terminology can be a huge barrier in conveying the culture and needs of the user to the developers.

Given these challenges, it is crucial to have an "on the ground" team of people who are trained in the existing template application functionalities and can interact with the potential users. This is critical to conveying the types of applications which are needed by the community, the customizations needed to address the particular needs of a user, and to train and obtain feedback from the users. This also is a

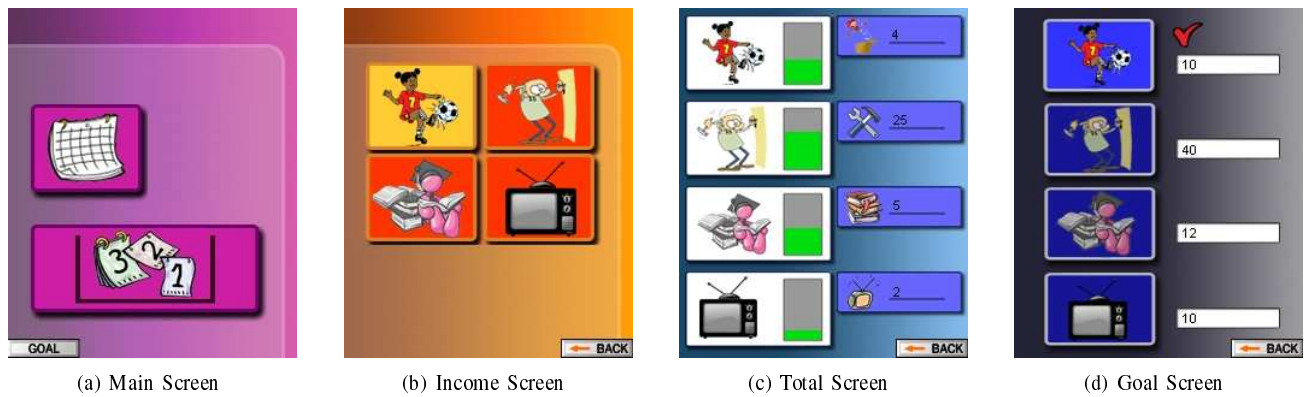


Fig. 9. Screenshots of the Time Planner application.

deployment challenge due to the fact that the "on the ground" team must be created, trained, and maintained throughout the development and deployment phases, and provide support to the users. Another deployment challenge is the deployment of application on existing mobile phones and platforms within the communities. Expensive and/or specialized mobile phones which are above the normal standard in the community can make the user stand out. In many cultures around the world, a user with this type of hardware could face political repercussions, be a target for crime, or face other negative consequences. As a result, it is important to know the types of mobile devices and the hardware features, capabilities, and limitations when developing and applying templates. This type of portability issues is a major challenge for development.

The last challenge is the set of programming and computer science challenges. Not only are platform compatibility and portability challenges, but it is a challenge to develop applications for mobile phones that are a couple of years behind the current technology standard. It can be difficult for developers to build applications for these systems when they run systems with more advanced technology. Lastly, and possibly most important to our template applications is the need to create and maintain a library of images for developers to customize applications. These images must be conceptual images which portray the concepts of the applications across cultural and country boundaries. Without the proper image, the meaning or concept of a particular function can be lost and therefore deem the application a failure.

## V. CONCLUSION

We presented the need for and the concept of template applications for mobile phones in developing informal economies, immigrant communities in developed countries, and the disabled. Template applications provide portability, customization and flexibility to address the needs of the people with applications which are language independent, highly customizable, and simplistic in operation. We illustrated the five different categories of template transformations for two main applications: an educational dictionary and a account book application.

## VI. ACKNOWLEDGEMENT

The authors wish to thank Adam Hanover and Raksik Kim, students at Stony Brook University, for their development of the template transformation applications and figures. In addition, we wish to thank the all the students of our 2008-2009 seminars on "Mobile Computing for Informal Economies" [7] for their hardwork, ideas, and enthusiasm.

## REFERENCES

- [1] Y. Borodin, J. Mahmud, I. Ramakrishnan, and A. Stent, "The hearsay non-visual web browser," in *International cross-disciplinary conference on Web accessibility*, 2007, pp. 128–129.
- [2] M. Nowak, *On ne prie pas qu'aux riches*. Editions J.-C. Latts, 2005.
- [3] "European microfinance network (EMN)," <http://www.european-microfinance.org>.
- [4] "Agoranov," [http://www.agoranov.com/index\\_uk.htm](http://www.agoranov.com/index_uk.htm).
- [5] "Harlem business incubator," <http://www.panix.com/~smsinc/hrp.html>.
- [6] T. Giridher, R. Kim, D. Rai, A. Hanover, J. Yuan, F. Zarinni, C. Scharff, A. Wasilewska, and J. L. Wong, "Mobile applications for informal economies," 2009.
- [7] A. Wasilewska and J. L. Wong, "Seminar on mobile computing for informal economies," <http://www.cs.stonybrook.edu/~cse651>, 2008-2009.