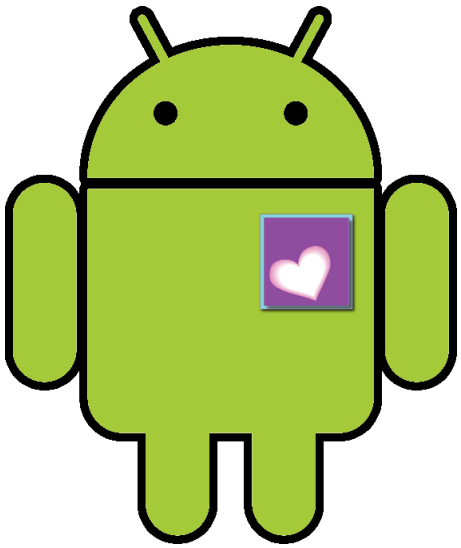
JCA: Code Archons

ModsDesigns

Final Report



Josh Lamson
Caitlin Hurley
Aaron Lebahn
June 23, 2011

1. Abstract

In the working world, too often parents are overly busy and therefore unable to give their children and loved ones the attention they deserve. The ModsDesigns project is to design an Android application based on the client's website, www.textcards4families.com, which provides "a simple yet fun way for anyone, mainly parents, to communicate with their loved ones, especially their kids." By converting this website to a mobile format, parents will be able to readily send messages of love and encouragement.

The primary purpose of this application is to allow users to browse, search, and send textcards™, which are greeting cards sent via picture message, to their loved ones that quickly and easily provide the love and encouragement a busy daily life can prevent. These textcards™ can all be displayed or searched and displayed by category, based on the user's search criteria, selected, then sent to a chosen recipient immediately or at a later time. The application also allows the user to view account information, such as a list of their recipients and their current subscription plan, and general information about the website, which includes the FAQ's, Terms of Use, and Privacy Policy pages.

The solution to creating this application was to develop using Java, Google's Android development tools, and XML as well as modify the existing PHP code. The languages used to develop the application allowed for formatting, user interface design, and to POST and GET information to and from the website. In order to communicate with the existing database, modifications and additions were made to the website's existing PHP code. These modifications and additions were also added to allow our application access to the existing functionality of the website, such as creating an account, logging in, and searching for textcards™.

The result of this project is a functional, portable version of the website that users can access "on the go". However, due to time constraints, there are still minor bugs and additions that were unable to be addressed. These include memory leaks, utilizing in-app billing, creating notifications to inform the users of the status of their textcard™, and allowing for customizable application settings. Fortunately, the application in its current state does meet all of the client's primary goals.

Table of Contents

1. Abstract 2
2. Introduction 4
3. Requirements 4
3.1 Functional 4
3.1.1 Mobile Application 4
3.1.2 Textcard™ Interactions 6
3.1.3 Database 6
3.1.4 Information Syncing 6
3.2 Non-Functional Requirements 7
3.2.1 Android Market 7
3.2.2 Advertising 7
4. Detailed Design 7
4.1 Architecture Design 7
4.2 Application Flow and State Diagrams 8
4.3 UML Diagrams 11
4.4 Database Schema 14
5. Implementation Details and Results 15
5.1 Language Choice 15
5.2 Library Usage 16
5.3 Tool Usage 16
5.4 Issues 16
6. Conclusions and Future Directions 17
6.1 Lessons Learned 17
6.2 Future Directions 17
7. Glossary 18
8. References 18

2. Introduction

Our client, Jill Modesitt, has asked our group to create a mobile application based on her current website, textcards4families.com. This website provides a fun, simple way for people to communicate with their loved ones using textcards™, greeting cards sent via picture message. Previously, the only way to access these textcards™ was through the website; however, the mobile application creates a more accessible method to send them.

This application was created for the Android platform with the goal to make it as similar to the website as possible. Users are able to register, log in, send and search textcards™, and change many settings on their account as necessary. These changes are then reflected on both the Android application and the website, allowing these two mediums to be used interchangeably.

Another goal for this application was to easily and effectively integrate with the current database (Figure 3) with minute changes. This goal was set in place with the knowledge that the database can only be accessed by the website. To account for this obstacle, additions were made to the PHP code running the website. By making these modifications, the application is able to use the website to access the needed information from the database.

By creating a mobile application that mimics the website, we will enable users to send these textcards™ at any time and any place. This will create a boom in our client's business by moving the service to a medium that is currently on an upswing of popularity.

3. Requirements

3.1 Functional

The four main components of this project are: to create a mobile application that can replace or supplement the running website, to allow for interaction between the user and the textcards™, to enable the application to access the current database, and to permit the application to sync with account and informational aspects of the website. Throughout each of these, POST and GET requests were used to communicate with the website. This means that when the application needs information from the website, it GETs this through a GET request. When the application needs to send information to the website, which can result in the website sending information back to the application through GET requests, the app POSTs the necessary information to the website.

3.1.1 Mobile Application

The first and most important requirement of this project was the creation of an Android application that can act either as a nearly complete replacement of the website or simply as a

supplement to the website. The application and website have linked interactions which allows users to make changes to either medium and see those changes displayed on the other. The application has the following Activities:

- Log In
    - This page has two options: enter a user name and password to log in or go to the register page to create an account.
    - After the initial log in, the app then automatically logs the user in each time the app is opened.
- Register - the user is prompted to enter the required information and agree to the Terms of Use. After they have submitted their information, it is sent to the website which then sends a confirmation email to the email provided. After confirming, the user is able to log in on the Log In page.
- Home - this page displays the available free textcards™, the user's last sent textcard™, and the user's queued textcards™ (up to 5 are displayed on this screen)
- Gallery - when this page is initially opened, a gallery view of about one hundred textcards™ (loaded one-by-one) is shown. The user can scroll through these cards or go to the Search Textcards™ page, accessed from the Menu, to look for textcards™ by category.
    - Search Textcards™ - this page GETs the list of categories and sub-categories from the database, then fills in the titles and drop-down boxes from the acquired list. The user can then choose which categories they would like to search, submit, and see the results back in the Gallery.
- My Account - this activity has all information concerning the user's personal account
    - My Subscription - this page displays the list of available subscriptions and descriptions about each of them. If the user has a subscription, the number of textcards™ and time remaining are displayed.
    - My Recipients - a list of the user's contacts and their information are displayed on this page. The user is also able to delete contacts, and add contacts, accessed through the Menu.
        - Delete Recipients – the user is able to delete contacts simply by tapping on the recipient's information. A dialog appears to ensure this is what the user wants to do. If so, the contact is deleted from the My Recipients page and the database. Otherwise, it remains in place.
        - Add Recipients - the user simply enters contact information for their recipient, submits, and the app POSTs information to the website which adds the recipient to the database. When the user goes back to the My Recipients page, their new recipient is displayed.
    - Queue and History - the app GETs lists of up to ten of the user's queued, sent, and failed textcards™ and displays them on this page.

    - Change Password - this page allows the user to enter their current and new password they wish to change to.
- Information - this activity has all general information about the website and product.
    - About Us - a brief description about the goal of the website and application is displayed on this page.
    - FAQs - The FAQs page shows the Frequently Asked Questions, retrieved directly from the website.
    - Charities - this page shows a list of the current charities users are able to donate to. Each charity has its own logo, mission statement, and description displayed.
    - Privacy Policy - the Privacy Policy page describes the privacy policy concerning the user's personal information.
    - Terms of Use - this page displays the legal information regarding the Terms of Use
    - Contact Us - the Contact Us page allows users to contact us with questions, comments, or request for textcards™.

3.1.2 Textcard™ Interactions

Interactions between the user and textcards™ is the primary function of the website and application. Users have the ability to interact with textcards™ in the following ways:

- View free textcards™ on the Home page
- View a gallery of about one hundred free and subscription-only textcards™ in the default order
- Search for textcards™ by category
- Send textcards™ to a chosen recipient instantly or at a scheduled time

3.1.3 Database

Database access was required throughout the entire application. The following components and functionality acquired information from the database:

- All aspects of textcards™
- Accessing and editing account information
- Accessing general information about the application and website

3.1.4 Information Syncing

In addition, there are the less complex requirements to simply sync information from the website to the application. These pages are synced solely from the website, with no interaction with the database at all:

- FAQs
- Terms of Use
- Privacy Policy

3.2 Non-Functional Requirements

The non-functional requirements included the process of publishing the Android application and advertising the application via the website.

3.2.1 Android Market

Uploading the application to the Android Market has a step-by-step routine that requires attention to detail.

1. Step 1: create a digital signature, it was required to sign the application with a cryptographic private key and ensure this key's validity period will end *after* October 22, 2033.
2. Step 2 was to define a version code, used for identifying the application internally, and a version name, to display to users as the application's version. These are the more technical steps in this uploading process.
3. Step 3: create a details page, is used so users know what exactly the application is and why they should use it. The details page contains an application description and screenshots that show the main aspects of the application. A proper logo and label that met Google's requirements for application upload was also created for this process.
4. The final step, Step 4, was to compile a complete .apk file for the market. The .apk file is what is uploaded to the market and made available to users.

3.2.2 Advertising

Advertising the application on the current website is also very important and still being worked on.

A QR Code was generated to allow users to easily access the application via mobile device and will be placed on the website as soon as possible. A URL to the application may also be placed on the website for user's who wish to access the application on their computer.

These methods of advertisements will be especially beneficial for current users of the website. They will quickly be informed of this new development and will easily be able to access the application.

4. Detailed Design

4.1 Architecture Design

The majority of this architecture was set in place by a previous group of developers. The only new contribution to the system is the use of an Android Application acting as the medium between the user and the functionality of the website. Everything beyond the Website (Database, Event Scheduler, etc) is unmodified by the application, and can only be accessed through POST requests to the website. The website will then return XML for the Android to parse and display

to the user. The website had to be modified in order for it to return XML when being accessed by the App. The website distinguishes it's response by looking at a POST variable called "response_type." This variable can either be "html" or "xml". The PHP files look at this POSTed variable, and either returns a browser readable HTML or parsable XML. Many of the objects retrieved in code were given a function called toXML(). This function takes all the information about a single object (A contact, a textcard™, etc.), and outputs it to the returned file in an XML format. From these, we can create objects in Java, and use them to the application's needs.

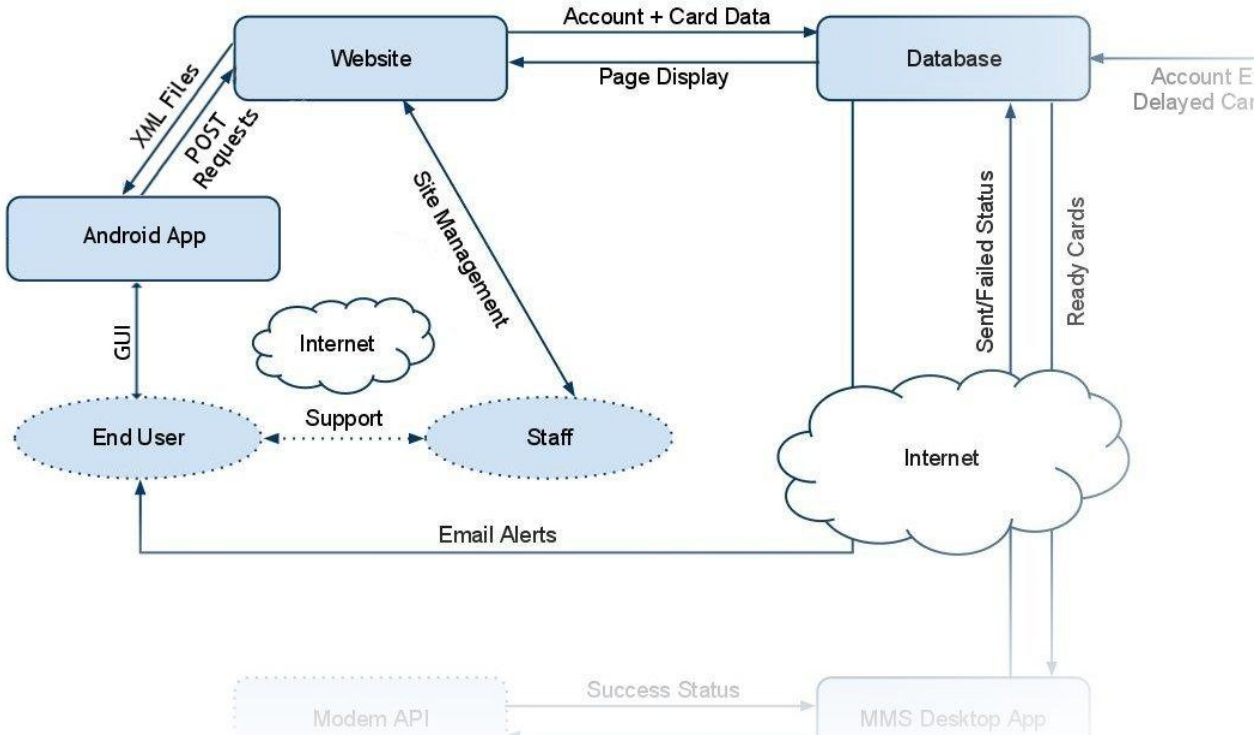


Figure 1: System Architecture

The System Architecture design for the Textcards4families System

4.2 Application Flow and State Diagrams

The following state diagrams describe the Activity flow of the application. An Activity is a single, focused thing the user can do. More often than not, an Activity is a single screen that can be viewed on the Android device. The first diagram, Figure 4, describes the entry flow of the program. The application is opened to a login screen. From there, the user can either register and create an account, or login using a previously created account. While registering, the user must fill out a form, and agree to the Terms of Use, which can be access from the menu on that page. If the user is returning to the app after logging in and closing the app, it will remember the users login information, and permit access to the main application automatically. Once logged in the main application begins.

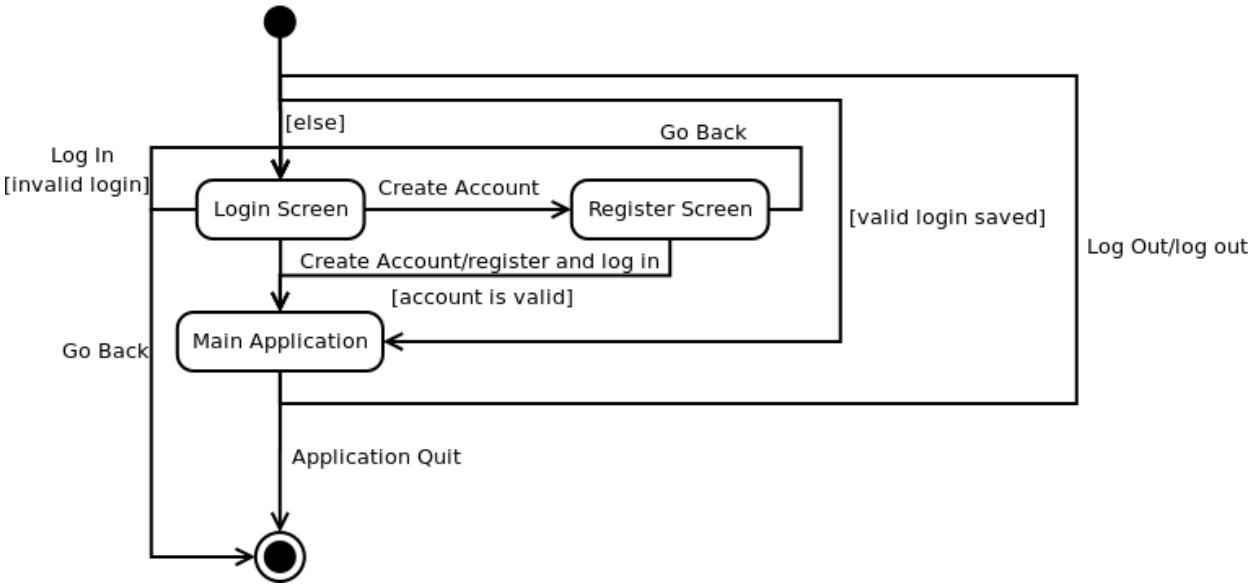


Figure 2: Overview State Diagram

This is an overview of the entire application structure.

Figure 5 shows the main application flow. It consists of four tabs that can be navigated between by pressing the corresponding tab. The home tab displays the currently offered free textcards™, the last textcard™ sent, and a shortened queue for the user. The gallery tab initially displays the first 100 textcards™. To see additional cards, the user may open the menu, and be redirected to a search page. From here, they use numerous drop-down boxes to select search criteria. Any one of the textcards™ can be clicked and sent. Once a textcard™ is clicked, a larger preview of that textcard™ is sent. Under the large preview is a button to send the textcard™. The user is then greeted by a form involving drop-down boxes, radio buttons, and text boxes. The user can then send the card given their inputted criteria.

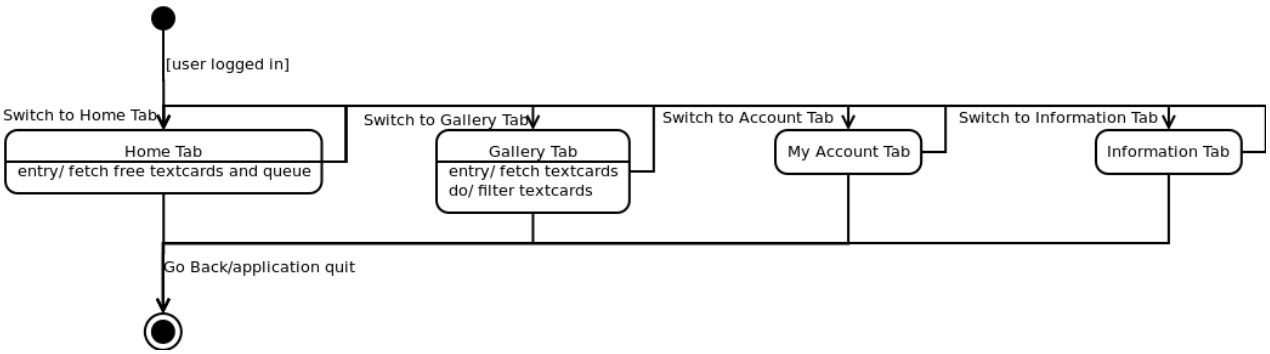


Figure 3: Tabs State Diagram

This shows the transitions between separate tabs

Figure 6 shows the sub-menus of the “My Account” tab. From this screen, you can view a more detailed Queue and History or your subscription plans. the user can also view, add, or

delete your recipients, and change your password. The Subscription page displays the user's current active subscription plan, and the available subscription plans for purpose. The My Recipients allows the user to view, delete, and add contacts to their account. The Queue and History page displays up to ten queued textcards™, the ten most recently sent cards, and ten failed sends. Last, the change password screen show a very simple form to change their password.

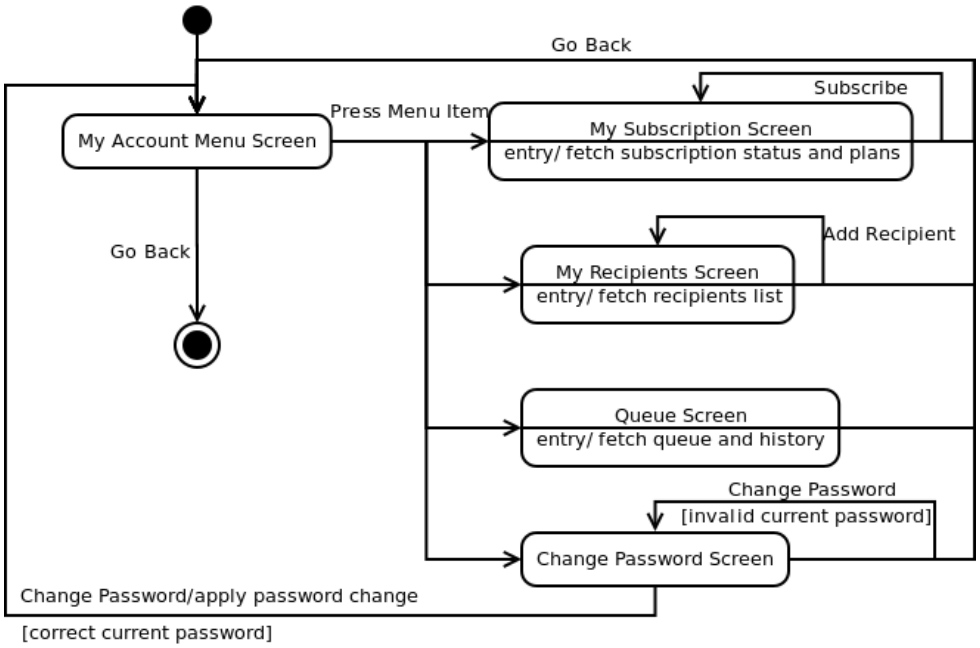


Figure 4: "My Account" State Diagram

This shows the transitions between menu items in the "My Account" tab.

Figure 7 shows the sub-menus for the Information tab. From there, you can view site information, FAQs, partnering charities information, Terms of Use, and Privacy policy. All of this information is fetched from the website, and displayed to match the web content exactly. The user can also contact textcards4families with questions, concerns, or textcard™ requests by filling out a simple form and submitting.

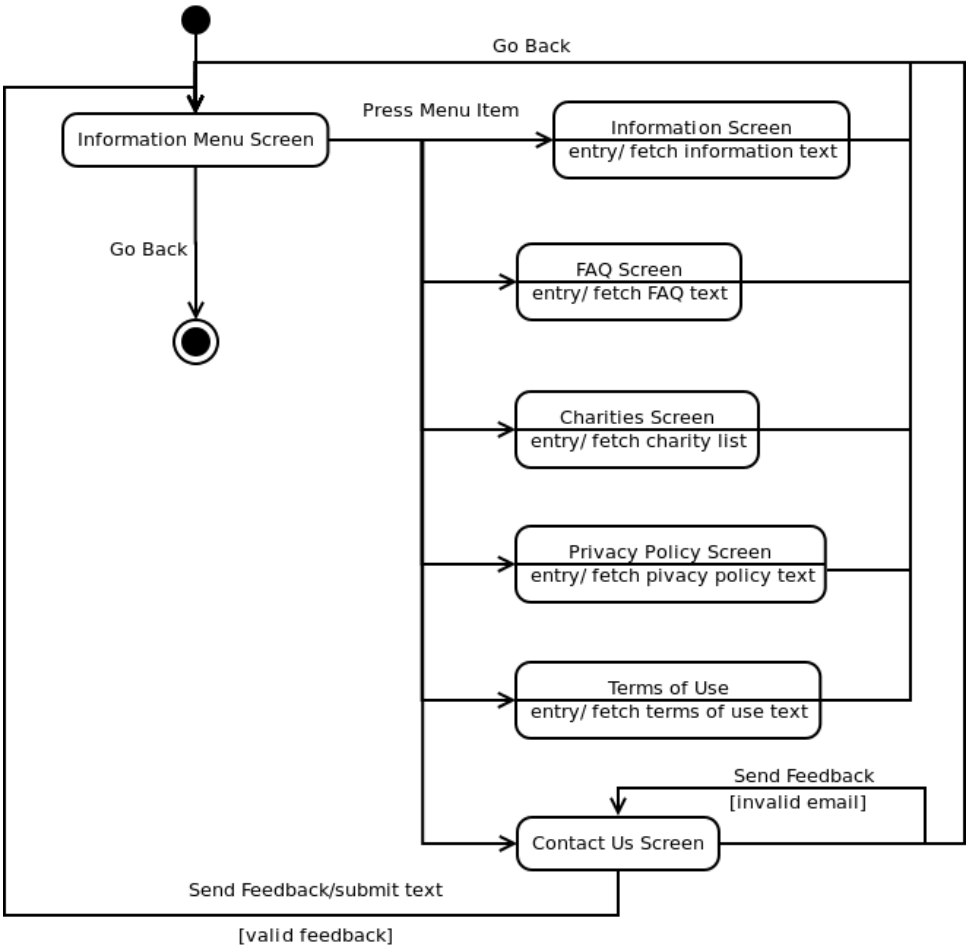


Figure 5: "Information" Tab State Diagram

This shows the transitions between menu items in the information tab.

4.3 UML Diagrams

Besides one class for each activity shown above in the state diagrams, there are frequent helper classes that are used for retrieving and displaying information from the website and displaying. Figure 8 is an example of this process, specifically for the charity information activity. The process begins when the InformationCharsActivity is started. When this activity is created, it creates an UpdateCharitiesUI object, and calls execute() on it. This class extends an abstract class named UpdateUI, which extends what is called an AsyncTask. An AsyncTask "enables proper and easy use of the UI thread. This class allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers." (from developer.android.com reference). This is important because it allows the app to (a) multitask, and (b) do work without the UI freezing and becoming un-usable. UpdateUI takes an Activity for reference, data to POST, a website URL to send the request to, and lastly a DataHandler used to parse XML responses from POST requests. It uses these to perform the fetching and parsing operations in a separate thread, while displaying a progress dialog in the

foreground to alert the user of background activity. The UpdateCharitiesUI specifically is passed a CharityParsingHandler. UpdateCharitiesUI then POSTs to the correct PHP file, and receives an XML response, like this one:

```
<?xml version="1.0" encoding="utf-8"?><!DOCTYPE CHARITY [<!ENTITY nbsp
" ">]>
<Charities>
	<Charity id="1" name="Families First"
	url="http://www.familiesfirstcolorado.org"
	logoPath="charities/families_first.png">
		<strong>Mission:</strong>  To provide services which
		strengthen families, empower parents, nurture children, and
		end the cycle of child abuse and neglect.
	</Charity>
	<Charity id="6" name="Women&#039;s Crisis &amp; Family Outreach
	Center (WCFOC)" url="http://www.twcfoc.org/"
	logoPath="charities/wcfoc.jpg">
		<strong>Mission:</strong>  WCFOC is dedicated to
		reducing domestic violence and family conflict through
		advocacy, empowerment, violence prevention and
		intervention, services and education. Since its inception,
		WCFOC has provided programs and services to any domestic
		violence victim requesting those services and to the region
		at-large.<br/><br/>If you would like to make a donation
		straight to WCFOC, please click here: <a
		href="https://www.blacktie-
		colorado.com/online_sales/nonprofit_donation.cfm?id=1974"
		target="_blank"><br/><img src="images/donate.jpg"
		width="80" height="20" border="0" /></a>
	</Charity>
</Charities>
```

Based on this XML response, the passed CharityParsingHandler will create a new Charity object for every “Charity” tag. All of these objects are placed into an ArrayList of Charities, which is then used by UpdateCharitiesUI. Based on information in the Charity, and hard-coded formatting information, the charities are placed into certain Views and added to the Activity, so they can be viewed by the user. This process is used throughout the entire application, for syncing any displayable information, such as FAQs, Contacts, Terms of Use, Privacy Policy, Subscription plans, and just about anything that can be printed to the screen.

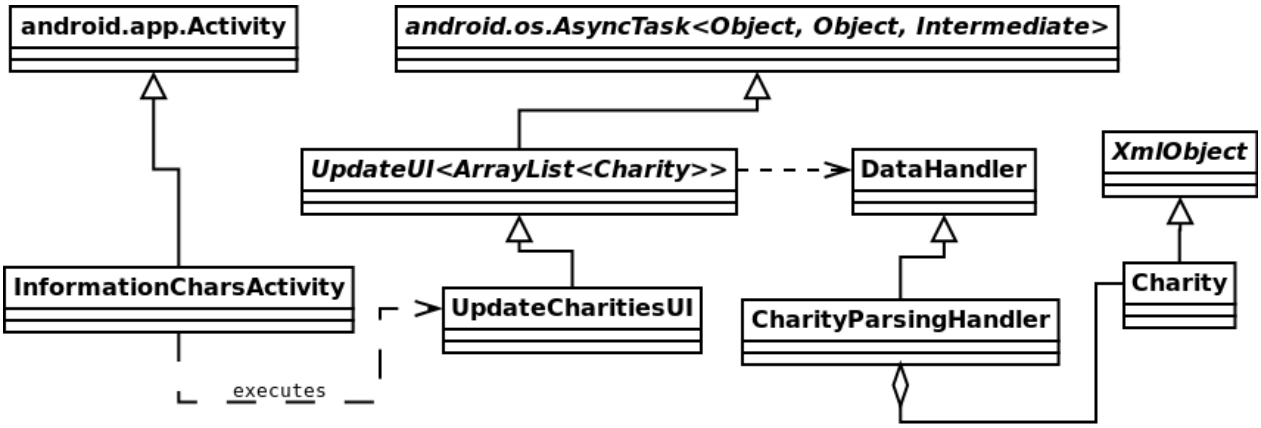


Figure 6: Website Communication Framework (Charity)

This UML encompasses all classes used in requesting, parsing, storing, and displaying Charity information

There are two slight modifications to the process taken above. The first of these is when data needs to be placed inside of a specialized layout, such as a GridView, ListView, or Spinner. This process is started by setting the Adapter of the View being loaded (In the below UML, the Gallery GridView). The Adapter for each case extends an Abstract XmlObjectAdapter, which is standardized to work with the Views mentioned above. This adapter accepts a new UpdateAdapter object, which is executed similarly to the UpdateCharityUI above. From here the processes become more alike. The UpdateAdapter is handed a DataHandler which parses XML and creates an ArrayList of XmlObjects, in this case Textcard objects. The ArrayList is then used by the adapter, which in this case, loads the data about the textcards™ all at once, then loads their thumbnails one at a time.

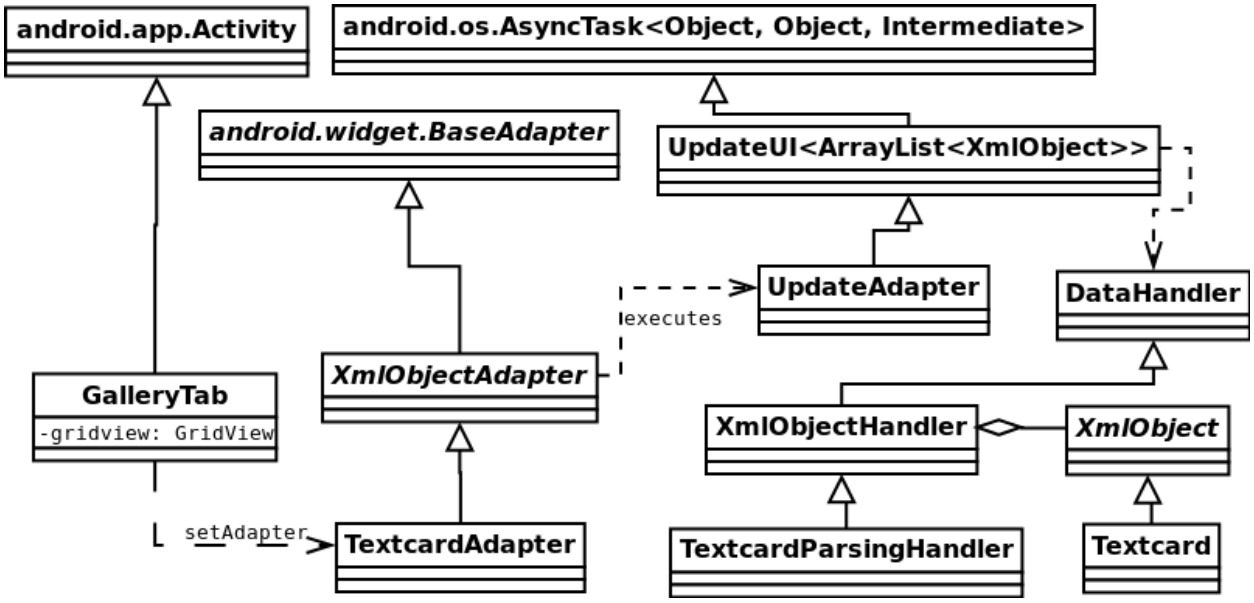


Figure 7: Website Communication Framework (Gallery Adapter)

This UML encompasses all classes used in requesting, parsing, storing, and displaying textcards™ in the Galley Tab

The other modification to the above examples is requests that only receive a status message in return. Actions like logging in or out, sending feedback, sending textcards™, and the like, only receive either a <success> tag or an <error> tag, and some message describing the response, like "Invalid Email." These executed actions are handled in a very similar manner, but instead of creating ArrayLists of options, they return only a string result, and allow access to a Boolean that indicates the success or failure of the result.

4.4 Database Schema

The database has been put in place by the developers of the website. It stores everything from user information, time zone information, and contact information to non-user based data such as textcard™ information and account transactions. All of the relationships between data is shown using "Crow's Foot" notation, which is described in the Glossary. The Database is only accessed through what are called Stored Procedures. These are called through special PHP functions defined in a Database PHP class. They can be used to obtain lists of IDs, or insert data into a relation, or do anything that can be done with MySQL.

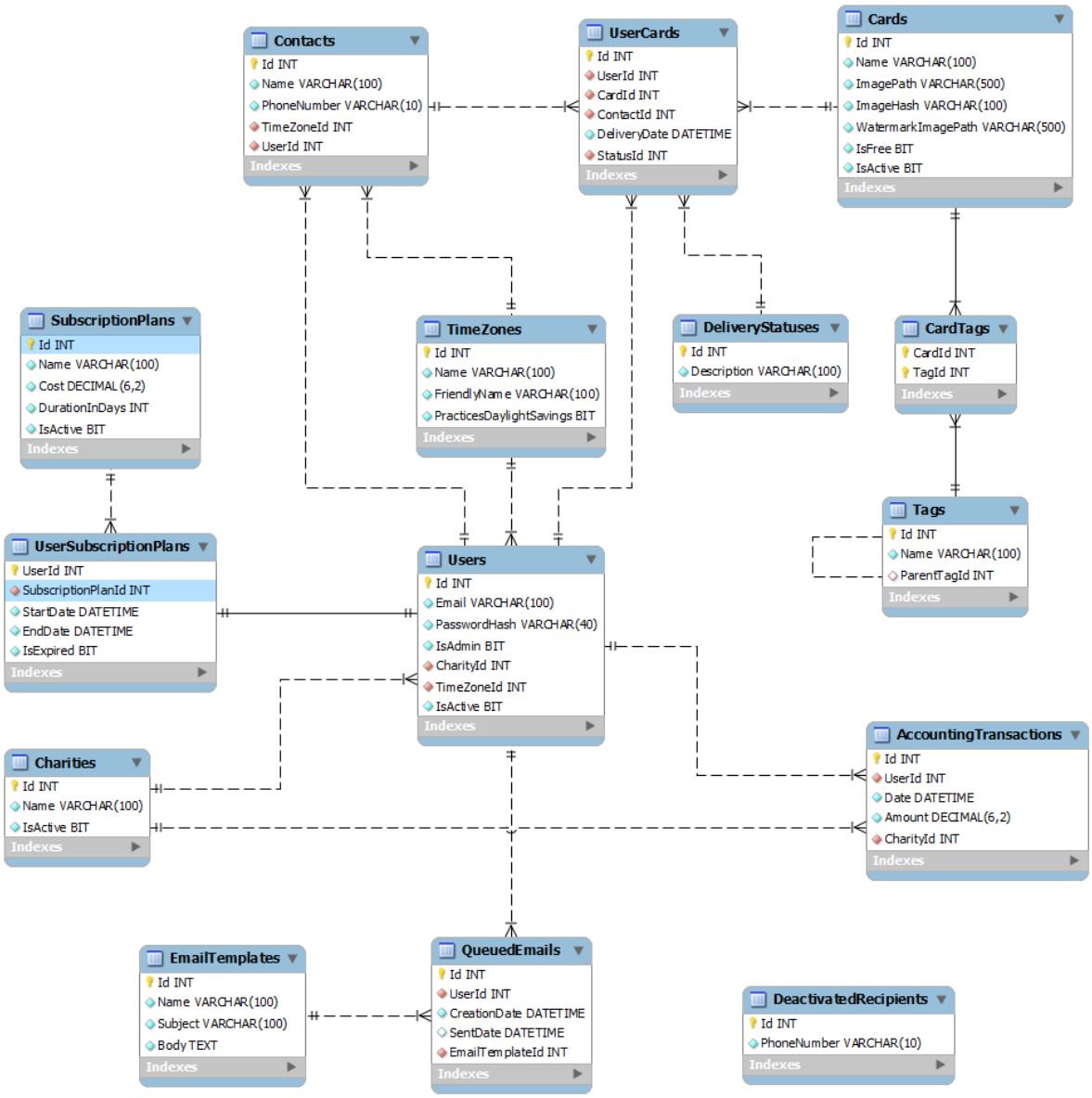


Figure 8: Existing Database

This is a diagram of the current database that the website is utilizing. Our goal is to allow the Android application to use this database with zero changes.

5. Implementation Details and Results

5.1 Language Choice

Several languages were used in the project. The primary language used was Java. This choice was directly related to the fact that the application was developed for the Android platform. The client did not specify whether the application must be developed for Android or

iOS. Since everyone on the development team was familiar with Java and not Objective C, developing an Android application was chosen.

XML was also used to serve two functions. The first was to provide layouts for the graphical user interface. This is the method that Android requires in order to specify static GUI design. The Java code for Android also allows for programmatic construction of the GUI; however, XML provides a simpler and more stable method to accomplish the same result.

The second use of XML was to communicate between the website and the app. Several reasons exist for this. First, the use of XML within the Android environment simplified learning. Second, XML was not only easily generated by the website, but it could also be parsed simply by the phone.

On the website side, PHP and MySQL were used. This was an easy decision. The app needed to communicate with the existing database, so MySQL was a must. In addition, since there was existing PHP code to interface with the database, the additional code was also written in PHP.

5.2 Library Usage

No libraries were explicitly added to the project, although several libraries were used that were included in Android. Google provided excellent libraries for graphical design and thread management that were used. In addition, the Android platform includes an HTTP client library written by Apache and an XML parser library called SAX. Since we required a method to send requests to the server and receive responses over HTTP, we needed a library that acted as a HTTP client, and the Apache client met our needs very well. Secondly, we needed a method to parse the XML response, and SAX was the only such library included with Android.

5.3 Tool Usage

Eclipse was used as the primary development environment. This was an easy choice as Google provides a full Android package for eclipse, including emulators for developers who don’t have android devices. There are other IDEs for Android applications, but Eclipse was the most highly recommended. Beyond its easy integration with Android, eclipse is also a very reliable IDE for Java, and most other languages that were involved with the project, including XML and PHP.

Initially, git and github were to be used as version control for the Android project, but due to poor eclipse integration, the standby option of subversion was used. Git was still used to make changes to the website’s PHP code. This was due to the code already existing on github, and the need for collaboration with the site’s existing webmaster.

5.4 Issues

One of the final issues with deploying the product onto the Android market was getting good compatibility with previous Android versions. An Android app must have both a build

target and minimum compatible Android Package Index (API). When development began, more or less random values were picked. When deployment came closer, these values were more closely observed and changed to reflect the compatibility goals. Changing these values had errors occurring in about a quarter to a third of the files. Issues included unknown constants for the minimum API, functions that couldn't be used anymore, GUI issues, etc.

6. Conclusions and Future Directions

6.1 Lessons Learned

An ample amount of time was spent working with POSTing and GETting data to and from the website. All but two of the activities in the application use GET or POST requests in some way. This was a new experience for our team and took some effort to get familiar with. One page that uses both GET and POST requests is the SendTextcardActivity. When this activity is started, the application sends a GET request to the website to retrieve the list of recipients and greetings used on the website. After the user has filled out the required information and clicks the submit button, the entered information is POSTed to the website where all actions to send the textcard™ are completed. However easy the concept of GET and POST requests was to grasp, getting a handle on the methods to do so required time and effort.

Another obstacle during development that pushed the team's limits was the occasional lack of an elegant solution. More often than not, Google has provided a plethora of tools to be used, and to account for any situation. Despite this, there were moments where there was no perfect solution to a problem. For example, we had some issues with backwards compatibility. For older Android versions and smaller less dense screens, textcards™ would appear large, overlap, and appear incredibly unsightly. After several hours of research, looking for a library or function to fix our problem, we eventually settled on getting the width of the screen, and doing some number crunching to set the size of out images. This was hard to do, but we learned the important lesson that sometimes, things just need to be brute forced.

6.2 Future Directions

The primary functions of the application are set in place and working properly. The application very closely mimics the website in design and functionality and meets the primary goals of the client. Additional features the application may implement are:

- More instructions throughout the App to make the user's experience with the application simpler and more enjoyable
- Notifications to inform the user when their textcard™ has successfully sent
- Implement in-app billing
- The ability to customize personal settings
- The ability to add recipients from the phone's contact book

7. Glossary

Activity: A single, focused thing the user can do, that almost always interacts with the user.

Crow's Foot Notation: A notation used in database diagrams. A line with a crow's foot on one end describes the relationship between two tables. The table without the foot contains many of the tables with the foot.

GET: An HTTP request which asks for the specified representation of an object. This request is used solely to retrieve information.

POST: An HTTP request which submits data to be processed to an identified resource.

Spinner: a widget similar to a drop-down list for selecting items.

Textcard™: A digital picture, similar to a greeting card, that is delivered as a picture message to mobile phones or emails**.**

View: This class represents the basic building block for user interface components. A View occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for *widgets*, which are used to create interactive UI components (buttons, text fields, etc.).

8. References

Android development information:
http://developer.android.com/index.html

Textcards4families website:
http://www.textcards4families.com