ANDROID APPLICATION FOR SCHOOL OF ART AND DESIGN

A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Sushruth Chandrashekar
Spring 2014
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the
Thesis of Sushruth Chandrashekar:
Android Application for School of Art and Design

__Carl Eckberg__
Carl Eckberg, Chair
Department of Computer Science

__Kris Stewart__
Kris Stewart
Department of Computer Science

__Leigh Conoir__
School of Art and Design

March 25, 2014
Approval Date
DEDICATION

To family and friends.
ABSTRACT OF THE THESIS

Android Application for School of Art and Design
by
Sushruth Chandrashekar
Master of Science in Computer Science
San Diego State University, 2014

According to “Steve Jobs”, Apple sold its one-millionth iPhone 3G, in just three days after its unveiling, and now they sell ten mobile devices for each laptop or desktop computer sold. In the past few years, the telecommunications field has experienced incredible advancements in network coverage, speed, and technological improvement throughout the world.

In this thesis I aimed to create an Android application for SDSU School of Art and Design that provides all the features that a student would often use when he or she visits the SDSU School of Art and Design website at their finger tips. This is a one-stop solution for getting all school of art and design related information like events, advising hours, scholarship, directory and course schedule.

Students can access information about all the events happening in School of art and design and also set reminder. The application sends out a push notification on the event day from which they can access the details of the events. Also students can get faculty and staff information like their email addresses, phone number and even check out their artwork. Accessing all the course details, student work and student organizations, has never been so easy.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT .................................................................................................................. v</td>
</tr>
<tr>
<td>LIST OF FIGURES ......................................................................................................... ix</td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS ............................................................................................... x</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS ..................................................................................................... xi</td>
</tr>
</tbody>
</table>

## CHAPTER

### 1 INTRODUCTION ........................................................................................................ 1

1.1 Background .............................................................................................................. 1

1.1.1 Why Should Universities/Schools Go Mobile? ................................................. 1

1.1.2 Outcomes ............................................................................................................. 3

1.2 Comparing iPhone and Android .............................................................................. 4

1.3 Functional Requirements ......................................................................................... 5

### 2 AN OVERVIEW OF ANDROID SYSTEM ............................................................... 6

2.1 Architecture of Android .......................................................................................... 6

2.1.1 Layer One: Applications .................................................................................... 6

2.1.2 Layer Two: Application Framework ................................................................. 8

2.1.3 Layer Three: Libraries ....................................................................................... 8

2.1.4 Layer Four: Android Runtime ........................................................................... 9

2.1.4.1 Dalvik Virtual Machine ............................................................................... 9

2.1.4.2 Core Java Libraries ..................................................................................... 9

2.1.5 Layer Five: Linux Kernel .................................................................................. 10

2.2 Android Applications .............................................................................................. 10

2.2.1 Android Application Components ...................................................................... 11

2.2.1.1 Content Providers ..................................................................................... 11

2.2.1.2 Activities .................................................................................................. 11

2.2.1.3 Services .................................................................................................... 14

2.2.1.4 Broadcast Receivers .................................................................................. 15

2.2.2 Activating Components ..................................................................................... 15
6 FUTURE ENHANCEMENTS ................................................................. 46
REFERENCES .................................................................................... 47
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2.1.</td>
<td>Android architecture.</td>
<td>7</td>
</tr>
<tr>
<td>Figure 2.2.</td>
<td>Compilation of java code into .dex code.</td>
<td>10</td>
</tr>
<tr>
<td>Figure 2.3.</td>
<td>Activity lifecycle.</td>
<td>13</td>
</tr>
<tr>
<td>Figure 3.1.</td>
<td>Showing the overall architecture.</td>
<td>17</td>
</tr>
<tr>
<td>Figure 3.2.</td>
<td>JSON structure.</td>
<td>19</td>
</tr>
<tr>
<td>Figure 3.3.</td>
<td>Home screen.</td>
<td>25</td>
</tr>
<tr>
<td>Figure 3.4.</td>
<td>Directory with FastScroll and alphabetical section index.</td>
<td>27</td>
</tr>
<tr>
<td>Figure 3.5.</td>
<td>Faculty details and email activities.</td>
<td>30</td>
</tr>
<tr>
<td>Figure 3.6.</td>
<td>Faculty portfolio and portfolio detail activities.</td>
<td>31</td>
</tr>
<tr>
<td>Figure 3.7.</td>
<td>Events list and event detail activities.</td>
<td>32</td>
</tr>
<tr>
<td>Figure 3.8.</td>
<td>Event notification.</td>
<td>33</td>
</tr>
<tr>
<td>Figure 3.9.</td>
<td>Gallery.</td>
<td>34</td>
</tr>
<tr>
<td>Figure 3.10.</td>
<td>Student advising.</td>
<td>35</td>
</tr>
<tr>
<td>Figure 3.11.</td>
<td>Course description.</td>
<td>36</td>
</tr>
<tr>
<td>Figure 3.12.</td>
<td>Course details.</td>
<td>37</td>
</tr>
<tr>
<td>Figure 3.13.</td>
<td>Organizations.</td>
<td>38</td>
</tr>
<tr>
<td>Figure 3.14.</td>
<td>Scholarships.</td>
<td>39</td>
</tr>
<tr>
<td>Figure 3.15.</td>
<td>Art supplies.</td>
<td>40</td>
</tr>
<tr>
<td>Figure 3.16.</td>
<td>Giving.</td>
<td>41</td>
</tr>
<tr>
<td>Figure 3.17.</td>
<td>Art council.</td>
<td>41</td>
</tr>
<tr>
<td>Figure 3.18.</td>
<td>Contact info.</td>
<td>42</td>
</tr>
<tr>
<td>Figure 3.19.</td>
<td>SDSU.</td>
<td>43</td>
</tr>
</tbody>
</table>
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CDMA</td>
<td>Code Division Multiple Access</td>
</tr>
<tr>
<td>GL</td>
<td>Graphics Library</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>HyperText Transfer Protocol Secure</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>NMT</td>
<td>Nordic Mobile Telephone</td>
</tr>
<tr>
<td>OS</td>
<td>Operating Systems</td>
</tr>
<tr>
<td>OEM</td>
<td>Original Equipment Manufacturer</td>
</tr>
<tr>
<td>RIM</td>
<td>Research In Motion</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SDSU</td>
<td>San Diego State University</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to Leigh Cotnoir (SDSU School of Art and Design) for giving me the opportunity to work on this thesis and also for her continuous guidance, support and advice throughout my thesis.

I also want to take this opportunity to express my deepest gratefulness to Professor Carl Eckberg, for being my thesis chair and giving me priceless inputs and guidance from time to time, just not on my thesis but also throughout my graduate work.

I am also very grateful to Dr. Kris Stewart, Professor of Computer Department for being so kind to accept to be a member of my thesis committee and her co-operation throughout my thesis.
CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Since the invention of the telephone there have been several advancements in the telecommunication field. The biggest break-through in communication came in the late ’80s with the arrival of mobile phone networks. Making use of various radio standards named with acronyms such as CDMA and NMT, handsets could connect with antennas. So many technology hurdles had to be eliminated to make mobile telephony possible. For example, it is no longer possible to relate a phone with a given switch, as the phone can easily roam through the country. In 21st century people could make a phone call from anywhere as long as they were within range (the more technical term for range is “cell”).

These mobile phones were not capable of doing anything other than placing calls. Then the explosion of the World Wide Web (Internet) happened, which changed the way people communicated with each other. The Internet became a very integral part of each and everyone’s day to day today life; slowly mobile phones makers slowly started increasing the computing capabilities of these devices so that people can connect to the Internet through their phones. These new devices were called smartphones. These devices became smarter and smarter; they started featuring music players, cameras, games and both entertainment and productivity applications. Since Smartphones had less computing power, there arose a need for operating systems designed specially for these devices.

Several companies came out with their version of mobile OS, which could efficiently make use of available computing power and perform well. Examples of such operating systems are Palm OS by Palm, Inc and RIM. However, two solutions stand out, and they are Google’s Android Operating System and Apple’s IPhones running iOS.

1.1.1 Why Should Universities/Schools Go Mobile?

By adopting mobile technology we can easily reach underprivileged communities easily. Studies show that cell phones can help bridge the digital gap by giving Internet access
to underprivileged teens. 41% of teens from households earning less than $30,000 yearly say they browse with their cell phone [1]. 2011 Horizon Report says that Mobile offerings are swiftly becoming a selling point for possible students considering educational options [2].

There are many opportunities of improving and increasing accessibility, communication and learning in universities. This is based on the fact that today, almost all students have a mobile device at all times. One crucial challenge one has to deal with while developing applications for mobiles is “Usability”. For the application to be a hit, the interface of the app should be highly efficacious, user friendly and usable mobile interface to support dynamicity of mobile and handheld devices. While the opportunities that mobile and wireless devices present us are new, however the challenges are kind of old, like smaller screen sizes and reduced input competences. All of these mean that accommodating the already available e-Learning services and contents to m-Learning is not a trivial task [3]. Delivering information on real time when needed improves user accessibility and fulfillment of the offered service in real time. Thus, based on the student feedback we believe that developing a mobile accessible learning environment improves the accessibility of the electronic learning content. To determine the effectiveness of m-Learning an experiment was conducted in 2008 with focus group consisting Communication Sciences and Computer Technologies faculty at South East European University [3]. In order to achieve that, they developed an application intended for the students of computer science to view the core and elective subjects they can choose, view the announcements and import dates. The development of this application is based on student’s needs, preferences and context. The users were using their mobile devices in order to access the content. The students logged onto a web server using their ID and from there they had three options:

1. View their exam results.
2. To see the new announcements and news.
3. To see the teaching content for a particular subject.

The focus group consisted of 10 participants age range 20-22 years, all of them students. Experts were 5 of them and other 5 were novices.
1.1.2 Outcomes

It was found that mobile access of the e-content and provided services at real time is far more effective vehicle in dissemination of knowledge and knowledge transfer primarily based on the user motivation and recent popularity the mobile devices are gaining in general. Students generally agreed that this approach with mobile devices is more intrinsically motivating and it encourages collaboration [3].

Mobile devices have become omnipresent in the today's world. With all the computing power of portable devices in everyone’s hand, it is time to start using mobile devices for education [4].

Since mobile devices are increasing exponentially in developing countries, Brown (2003) [5] claims that it's is about time to start thinking our future in which mobile devices play a very important role in all aspects of human life including education. Although there are many people who use mobile technologies every day, majority agree that mobile learning will play a major role in electronic learning. The ability of transmitting learning modules and all the administrative data wirelessly and securely to enable quick and smoother communication between students and faculty is phenomenal.

By expanding our education system to wireless medium, the educational institutions can bring great convenience to the learners who cannot always find an Internet enabled computers to obtain all the crucial information from their schools/universities. Mobile educational services are going to benefit both the learners and teachers, as they can access the services offered by school anytime and anywhere they want [6]. An experiment conducted in 2005 with a mobile prototype system, which was designed to provide students and teachers with educational services had the following functionalities:

1. News: This option provided all updated campus news for teachers and students.
2. Help: Where students can find basic operation guides by clicking on “help topics”
3. Contact List: Would give contact details of the department.
4. Classified Ads: Both the students and teachers can use this to post advertisements.
5. Academics: Students can access timetable, course center, coordinators contact.

Twenty-five participants who expressed their opinions in the form of a questionnaire used this prototype.
The participants included: One Academic administrator, One Associate Professors, Two Lecturers, Nine Postgraduate Students and Twelve Undergraduate Students. After all the participants went through the prototype, feedback was collected. The feedback given on the usefulness of the prototype was very positive. 80% of all the participants acknowledged that the prototype was a good complement to the existing web-based systems [6].

- Students can get all the information pertaining to their studies through one interface.
- The services were available wherever and whenever you need them.
- It will be heavily useful for those who use mobile phone and desktop education system a lot.
- All-important functionalities of a desktop system were provided in the mobile system.
- Handy way to access all essential information.
- Simple and user friendly.
- Information can be delivered to users more quickly.

From all of the above research it was evident that the School of Art and Design was going to benefit heavily from this and in turn would help students to consume knowledge and information from anywhere and anytime.

1.2 COMPARING iPHONE AND ANDROID

While iOS and Android both deliver efficient solutions for fulfilling the user’s computing needs, there are some differences between them. Currently iOS version 7 is the newest on iPhone and version 4.3 on Android. The primary difference between these two operating systems is the platforms on which they run. iOS is custom-made to run on devices manufactured by Apple such as iPhones, iPads, etc. only. Android on the other hand, since it is based on open sourced Linux OS, it can be run on array of devices. Thus, a lot of OEM firms chose Android for their devices, thereby boosting its share in market.

The second difference is that, Apple forces very strict application development rules and screening procedure to let any application to be made available to the public. Apple strictly controls tools that are used for app development on iOS. The testing and debugging tools available on Android are very good compared to iOS’s Xcode.
On the other hand Android OS is open sourced, supports many open development platforms and has the ability to connect or run third party tools to improve functionality of applications. Google has partnered with many handset vendors and service providers, giving consumers choices the iPhone does not have. HTC, Samsung, and LG each have at least one android phone offering. This is all because Google offers Android as an open source solution, thus making it possible for any handset to use it as software development platform. Giving consumers choices has clearly paid off for android [7].

The idea of this thesis was to build an easy to use Android Mobile Application that would help students to access School of Art and Design related information quickly.

1.3 Functional Requirements

This thesis was developed under the guidance of Leigh Cotnoir (San Diego State University School of Art and Design). Below are the essential features of this application. Requirements were gathered with valued inputs from Leigh Cotnoir.

The features include:

- The application must have provision to display the School Contact information like office, working hours, driving direction etc.
- The application should be able to list all the Faculty and Staff information
- The application should be able to retrieve all the events happening at the School of Art and Design.
- Students should be able to view the event details and also be able to set reminder.
- Application should be able to list all the student work based on program.
- Students should be able to browse through all the pictures by swiping through them.
- Students should be able to get all the course related description.
- Students should be able to get details about all the student organizations.
- Students should be able to get scholarship related details.
- Students should be able to get Advising hours.
- Students should be able to get all the art supply store information.
- Application should provide a way to get to SDSU giving page.
- Application should provide a way to get to SDSU Art Council page.
CHAPTER 2
AN OVERVIEW OF ANDROID SYSTEM

2.1 ARCHITECTURE OF ANDROID

Android Operating System (OS) is nothing but a software stack consisting of many layers, where each layer consists of several different sub-components. The layers are:

- Applications
- Application Framework Services and Libraries
- Libraries, Daemons and Services
- Linux Kernel that has drivers for hardware, file system access and inter-process-communication.

Every layer in the architecture is responsible for providing different services to the layer on top of it. Figure 2.1 [8] gives an overview of Android Architecture. We will discuss each briefly about each layer in coming sections.

2.1.1 Layer One: Applications

In Android Architecture the topmost layer is called Applications layer. Android houses a set of core applications like: (1) SMS Program, (2) Web Browser, (4) Email Client, (6) Maps etc.

All the applications are written in Java. Tools provided by Android SDK will compile data and resources into one single file called as Android package file with .apk extension. This file can be thought of as one application and this is used to install the application on an device running android OS.

Once this .apk file has been installed, Android handles each application as a different user; this is what makes the Android operating system a multi-user system. This system assigns an ID to each of these applications and sets permissions to all files that an application might need; thus only ID allotted application can access them. Every application runs inside its own process space, every process has its own virtual machine, thus an application’s code runs independently from other applications.
Two applications exchange information between them by using the same ID. An application can access devices like contacts, internet etc. by requesting permission, which will be given while installing the application. Android applications consist many application components [9].

Four types of components supported by Android applications are:

- **Activities**: Each screen that user sees and interacts with is an Activity. Every activity runs independently and one application can kick start an activity of another application.

- **Services**: Are used to perform heavy lifting such as playing music in background. Typically services don’t have user interface [9].

- **Content Providers**: Data can be shared between applications using content providers.
- **Broadcast Receivers**: Can be used to set reminders or send push notifications etc. Usually broadcast are triggered by the system. We can programmatically trigger broadcast through applications as well.

### 2.1.2 Layer Two: Application Framework

The application layer directly interacts with this layer. This layer enables the reuse of different components: Applications can expose their capabilities and any other application can take advantage of these capabilities. Layer however enforces security constraints on the capabilities [10].

Application Framework Components are:

- **Activity Manager**: Lifecycle of applications is handled by this component. Also provides a common navigation backstack.
- **Location Manager**: Global Positioning Systems (GPS) or a cell phone tower is used to manage location service.
- **Content Providers**: Used by applications to share data between other applications.
- **Telephony Manager**: Used to manage voice calls.
- **View System**: Assists in building of an application, like text boxes, buttons etc.
- **Notification Manager**: Manages alerts. Used to display alert in notification bar on top.
- **Resource Manager**: Manages resources like Videos, images, music files etc.

### 2.1.3 Layer Three: Libraries

This layer is nothing but a set of instructions that permits the device to handle different kinds of data. Playback, recording of audio is all done in this layer. The media framework library supports video, different picture formats. This layer also exposes many capabilities through the Application Framework. These libraries are written in C or C++.

Some of the important libraries are:

- **Surface Manager**: This is responsible for off-screen buffering. Drawings are not directly drawn on screen but are put on this off screen buffer. Here it is joined with other drawings to give us the final image on screen.
- **OpenGL**: Graphic Library shortly known as GL is responsible for rendering 2D and 3D contents on the screen.
- **FreeType**: Used to render bitmap and vector font.
- **Media Framework**: This library can be used to record using different media codecs and thus allowing playback of various media formats.
SSL: Secure Sockets Layer (SSL). Is a cryptographic protocol used to communicate securely over Internet.

SQLite: SQLite is an ANSI-C-based, open source database software. For the advantages of small size and fast running speed, the software is mainly used within application software where systematic storage is required to process and manage less complicated yet large amounts of data effectively or suited in onboard software made for embedded devices. With constant updates made into the software and for its widespread popularity among software developers, further and broader adoption of SQLite is anticipated [11]. SQLite is an open source database software, a light weight system, and reliable; it is extensively used in application software that needs to save native data in a methodical manner or adopted into embedded device software. Unlike other types of database systems, SQLite is typically used as “local-only” database that saves all the results of database usage in a one file. SQLite has a small library sized about 250 KB with full structure and about 180 KB when running with necessary options only. It also runs on the minimum stack space of 16 KB and uses heap size of roughly 100 KB, which is very tiny. This enables the users to setup a database engine in an operation environment with limited active memory, such as mobile phones, PDAs, and MP3 players [12].

WebKit: Houses tools that render HTML, so can be used to browse web pages. Also has Cookie manager and JavaScript Engine.

Libc: This is a standard C Library modified for embedded Linux devices.

2.1.4 Layer Four: Android Runtime

It houses Dalvik Virtual Machine shortly known as DVM and the Core Java Libraries.

2.1.4.1 Dalvik Virtual Machine

The Dalvik Virtual machine is the virtual machine used in android. This is an interpreter, which was developed by Dan Bornstein of Google. DVM and JVM are not compatible with each other as DVM is built for small handheld systems.

Figure 2.2 [13] gives an overview of this process. Using DVM we can run multiple virtual machine processes per device, this provides high efficiency even in low resource environment.

2.1.4.2 Core Java Libraries

Android applications are written using Java Programming Language. Core java libraries provide most of the core functionalities like file access, network access, data structure.
2.1.5 Layer Five: Linux Kernel

It is this layer that cooperates with the hardware and also includes all of the necessary hardware drivers. Also this forms the base layer on which other layers are built. Linux 2.6 Operating System is in the core of this and this is the one responsible for performing memory management, security settings etc.

Drivers are software programs that regulate and communicate with the hardware devices. Example, any device that has Bluetooth hardware in it, the kernel must have a Bluetooth driver that helps in exchanging information with the Bluetooth devices. Linux kernel also acts as an abstraction layer for the hardware and the software layers.

2.2 ANDROID APPLICATIONS

Top layer in the android architecture consists of Android Applications. These are written using java programming language. An android application is made of application components like Services, Activities, Broadcast Receivers and Content Providers. An asynchronous messaging known as Intents activates services, Activities and Broadcast Receivers.

All android applications contain a manifest file, which declares all the components that are used in the application along with all the requirements like hardware, version of
android, etc for an application to function properly. Android application also contains resources like images, layouts etc.

2.2.1 Android Application Components

Application components are the building blocks of an android application.
The four types of application components are covered in the Sections 2.2.1.1 through 2.2.1.4.

2.2.1.1 CONTENT PROVIDERS

This is the interface for exchanging data between two processes. Content Providers are primarily used to get access to data. The data can be present in a file system or SQLite database. Content Providers provide mechanisms for securing the data by encapsulating them. With the help of Content Providers, other applications can query or modify data. For example, for example Android has a content provider for getting contact information of the user. Thus, any application with proper permissions can get access contacts.

2.2.1.2 ACTIVITIES

Activity is the one that provides a view with which users interact. Any screen you see in an android application is an activity. We can say that activity is the building block of the user interface. Usually an application is made of many activities that work with each other in tandem to send and receive data [9].

2.2.1.2.1 Activity Stack

Activity Stack sometimes referred to as “back stack” is nothing but a data structure with basic “Last in first out” mechanism. Application uses this stack to maintain history of all activities. Every time an activity is started it’s pushed on to the stack and the previous activity is preserved in the stack. Whenever back button is pressed the current activity is destroyed and previous activity is popped from the stack it resumes its execution.

2.2.1.2.2 Tasks

A user goes through a sequence of activities to accomplish an objective, this sequence of activities is known as a task. These tasks can be interrupted and a new task can be started.
For example when user presses home button, previously executing task will be interrupted and when user opens another application a new task will be started.

2.2.1.2.3 Activity Life Cycle

All activities have a life cycle that they follow, in this life cycle they transition between many states. Activity lifecycle is shown in Figure 2.3 [14].

The state transition can happen when the activity is being created, resumed, stopped or destroyed. Activities will respond to state change and there are many callback methods that handle the state change. The user cannot not call the callback methods also known as the activity lifecycle methods, instead they are called by the system. It’s the responsibility of the user to implement these callback methods to do the specific task [14].

An activity can be in one of the three states:

- **Active (Resumed):** Here the activity is in the foreground of the screen and the user has focus.

- **Paused:** Here another activity is brought to the front, so the previous activity has lost focus, but the activity is still visible i.e. the activity does not occupy the entire screen. All the state information is preserved. An activity that is paused can get destroyed in very low memory situations.

- **Stopped:** When in this state, activity goes into background and is not visible to the user. All the state information is preserved. An activity that is stopped is still alive, but is often destroyed when memory needs to be given for some other task.

Lifecycle methods are shown in Figure 2.3 [14]. They are:

- **onCreate():** This method is called when activity is created. This method is really important because all the necessary components are initialized here. Within this method the setContentView() method is called to define the UI layout for the activity.

- **onStart():** Activity is made visible to the user right here. Any resources that are required by the activity must be started here. Example, Broadcast Receiver if needed by the user, then it can be recorded in this method.

- **onRestart():** This method is called after the activity is stopped (but not destroyed), just prior to calling the onStart() method.

- **onResume():** Here the activity is brought to foreground and is running. The user also has focus and also can interact with the activity.

- **onPause():** This method does not destroy activity but the activity rests on the activity stack and is no longer in focus. When the user leaves the current activity this method is called. This method must preserve any changes the user has.
• onStop(): User won't be able to see the activity as it is hidden. Clean up should be done here, like releasing any resources held by the activity. Example, if a Broadcast Receiver was used by the user, then it can be unregistered in this method.
• onDestroy(): All the resources connected with the activity are freed here.

2.2.1.2.4 Saving State
The state of activity must be preserved whenever you stop or pause. All details about its data members must be saved so that whenever the activity is brought back we can get its information easily and bring the activity to foreground.

However, when system runs into low memory it will destroy paused activities. This is where saving state becomes very important.

Two types of saving states are:
• Dynamic Instance State: This state hoards information like the instance variables of the activity etc. The dynamic instance state is usually saved using method onSaveInstanceState() which is called after the onPause() method and reinstated in the onRestoreInstanceState() or onCreate() method.
• Persistent State: The state that hoards information that should be accessible next time the application is run. Example, the contact information. This state is usually saved using method onPause() and restored in the onResume() method.

2.2.1.3 SERVICES
A service is an application component, which runs in background and does not interact with UI. It is usually used for doing heavy lifting, which involve long operations, like fetching data over the network.

Two forms of services:
• Started: In this form an application component kicks off the service; the service once started runs indeterminately in the background. When done, it stops by itself but does not return any result to the caller. A good example for this can be file download from the internet.
• Bound: In this form the application component attaches itself to the service. Such a service is called a “bound” service. The components send requests, get back results, and communicate with processes using interprocess communication (IPC). This kind of service does not run forever. Many components can bind to the same service and is destroyed after all the components unbind with the service.
2.2.1.4 **Broadcast Receivers**

Can be used to set reminders or send push notifications etc. Many broadcast are made by the system, but can also be triggered by the application. Broadcast receivers respond to system wide announcements. This announcement can be made by the system or an application can trigger it. Some examples for such announcements are event reminder, when battery is low etc.

Broadcast Receivers will not interact with user interface but can create status bar notification to notify the user.

Two types of broadcast receivers are:

- Ordered Broadcasts: These broadcasts are sent to one receiver at a time. Intents are used for transporting and accepting broadcasts, which run in background and are not aware to the user.

- Normal Broadcasts: All receivers of this type of broadcast run in no specific order. These broadcasts are very well organized, but receivers cannot use the results.

### 2.2.2 Activating Components

Android system has the ability to start one application from another application component. In this case the component will run in the application’s process that houses it and the one that started it. Thus android applications wont have a single entry point.

Android system will run each application in an independent process which has file permissions for preventing other applications. Therefore, applications cannot straight away activate a component from a different application. Hence the components that have been activated signals telling which intent to start. Then the system activates that intent.

Intents can be thought of as glue that holds different components together at runtime. Intent can activate Activities, Services and Broadcast Receivers application components. Intents can also be used to return data if an activity needs it.

### 2.2.3 Application Resources

Application Resources form an integral part of any android application. They are very important for the visual presentation of the application. Some of the resources include images, styles,, layout of an activity, etc. Some of these are defined in XML files.
They are used mainly:

- To update the Characteristics of an Application: Can renew various characteristics of the application without modifying code.

### 2.2.4 The Manifest File

Every Android application has a manifest file `AndroidManifest.xml`. All the components that are used in this application must be defined here. When the application is run, the system reads this file and checks if the application component exists before starting the app. This manifest file exists in the root of the application’s project directory.

The android manifest file also has information about:

- **Application Components**: Here we define all the components used by the application like activities, services, broadcast receivers, and content providers.
- **Permissions**: In an Android application if you are accessing any private API or user data, you will need to obtain permission. These permissions must be defined in the manifest file. We will discuss more about the permissions in the next chapter, when we get to the implementation specifics of the app.
- **Minimum API level**: We can specify the minimum API required for this app to run.
- **Application Features**: Defines any hardware and software features that the application might need, like Bluetooth services, multi-touch screen, etc.
CHAPTER 3

DESIGN AND IMPLEMENTATION OF SDSU SCHOOL OF ART AND DESIGN APP

3.1 ARCHITECTURE

The Basic Architecture of the applications involves 3 important things. Figure 3.1 gives an overview of this process:

1. Smartphone running Android OS Froyo(2.2) and above.
2. PHP Server to serve the data.
3. MySql Where the Actual data is stored.

Figure 3.1. Showing the overall architecture.

3.1.1 MySQL

Let’s first start with the MySql database. MySQL offers an implementation of a SQL database suitable for small to medium websites. For our application we used MySql database that was already in place to serve webpages. Many applications that use MySQL are focused towards the LAMP stack (Linux, Apache, MySQL, PHP). MySQL is usually used with two distinctive storage engines, one is called MyISAM, which does not support transactions and
stores each table in a set of three files. MySQL has one major advantage: it is free. MySQL has some issues with permanency and clustering; it is very hard to install a steady database cluster with MySQL with the regular version. Depending on the database storage, MySQL will support transactions or not, so the requirements of the application have to be taken into account when creating the database tables. For large, heavy loaded databases, it is a major operations problem that changing the database structure is only possible when locking the complete tables. This will mean that the database cannot be accessed during that operation so that this can only be done during low traffic times.

3.1.2 PHP

PHP is an server side programming language available free of cost that can be easily downloaded from php.net. Its very easy to learn and understand and also very efficient on cross-platforms like Windows, Linux, and UNIX, and varieties of the latter two. An advantages of running PHP is that it does not put strain on servers. It uses its own built-in memory space that reduces the workload and enhances the processing speed automatically. PHP is optimized to make the server’s job easier; thus nowadays the use of PHP is popular among programmers [15].

Advantages of PHP [15]:

- PHP is accessible and available for free.
- There is a wealth of documentation available online, in many languages.
- PHP is a loosely typed language, which makes basic scripts work much faster and allows developers to concentrate less on design.
- PHP is flexible, we can write php programs in either object oriented way or procedural programming way.
- It is system independent, can run on Unix, Windows.
- Programmer of Java, Perl, C can find many parallels to ease transition to PHP.

For our application we used PHP intensively to access raw data from MySql and convert the raw data in JSON format before returning it to Android.

3.1.3 JSON

JavaScript Object Notation popularly known as JSON is a light-weight data-interchange format.
The advantages of JSON are:

- Easy for machines to generate and parse.
- Readability and write.
- Completely language independent. Conventions used are familiar to:
  - Programmers who are familiar with languages like C, C#, C++, Java, Perl,
  - Python, JavaScript, etc.

JSON has two structures [16]:

1. Object which has unordered set of key/value pairs: The key is of type string and the value can be string, integer, object, array, boolean, or null. The object begins with a left brace (\{) and ends with a right brace (\}). The colon shadows each key (:). The object structure is show in Figure 3.2.

2. Array has ordered list of values: Each element in the array has a value. The array starts with a left bracket “[” and terminates with a right bracket “]” as shown in Figure 3.2 [16].

An example of JSON object looks something like this.

```json
{
    "firstName" : "Sush",
    "lastName" : "Chandrashekar",
}
```
Dealing with JSON in PHP is easy. You just need to create a PHP object and populate that object with key value pairs and at the end use method `json_encode` provided by PHP to convert this object to JSON.

Example:

```php
if(!$result = mysqli_query($db,"select * from abc"))
    die('SQL ERROR while getting parent: '.mysqli_error($db));

// check for empty result
if (1) {
    // looping through all results
    $response["abc"] = array();

    while ($row = $result->fetch_array()) {
        $temp = array();
        $temp["id"] = $row["id"];
        $temp["name"] = $row["name"];
        // push into response array
        array_push($response["abc"], $temp);
    }

    json_encode($response);
}
```

Dealing with JSON in java is very simple. There are two classes for this purpose. `JSONObject` class, which deals with top level JSON object and `JSONArray` as the name suggests, is a class that deals with JSON array.

Full flow of accessing the data and Rendering it on the screen goes like this.

1. We call PHP script that resides on the server through HTTP.
2. PHP script queries the MySql database and gets the raw data.
3. PHP script converts the raw data to JSON object with help of `json_encode`
4. The JSON object created is served to android application.
5. The application parses the JSON object with help of `JSONObject` and `JSONArray` classes and extracts the data
6. This extracted data is tied to UI for display on the screen.
3.2 DESIGN

While designing any app whether it is a web app or mobile you need to keep a few things in mind. The most crucial and important things to keep in mind are accessibility, reuse and usability.

3.2.1 Accessibility

As mobile technology becomes more and more popular among many aspects of life: education, employment, health care, government, commerce, recreation, and more, it is vital that the web is accessible in order to provide equal access and equal opportunity to people with disabilities. The U.S Government embraces accessibility at the workplace in both the Americans With Disabilities Act (ADA) and the Federal Rehabilitation Act. In brief, both legal documents can be taken to apply towards the use of technology.

Here is the brief background about how ADA compliance begun. Section 504 of the U.S. Rehabilitation Act of 1973 states that:

- Commands that institutions accepting federal funds provide equal access to their programs.
- Declares that disability rights are a form of civil rights and thus should be covered by the 14th Amendment of the U.S. Constitution.
- Uses all the institutional budget (not just the computing area’s budget) in gauging the “Reasonableness” of required accommodations for accessibility [17].

This Act was the first act, which executed the thought of providing accessibility aid to the people with disabilities.

Next the U.S. Rehabilitation Act of 1973 was extended with the Americans with Disabilities Act (ADA) in 1990. The ADA of 1990 states:

- Requires that every institution getting federal funds establish and keep a plan of compliance [18].

Finally, in 1998 an amendment was passed to Section 508 of the U.S. Rehabilitation Act of 1973 [18, 19]:

- Provides accessibility standards, developed by the Architectural and Transportation Barriers Compliance Board (Access Board), for the web and many other areas of electronic and information technology [20].
- States that federal agencies must ensure web sites are accessible to employees and the public to the extent it does not pose an “undue burden” to the site owner.
Standards for webpages were published Dec. 21, 2000.

Enforcement date was set at June 21, 2001.

Android provides accessibility features and services for helping such users to navigate their devices easily. Making an application accessible is about promise to usability and getting the details right, so that it delights the user [21].

The following steps must be completed in order to ensure a minimum level of application accessibility [21]:

- **Describe user interface controls:** Components that do not have visible text should be given content description, particularly ImageButton, ImageView and CheckBox components. The “android:contentDescription” XML layout attribute must be used or the “setContentDescription(Charectersequence)” method must be used to deliver this information for accessibility services.

  ```xml
  <ImageView
      android:id="@+id/contact_entry_image"
      android:src="@drawable/ic_launcher"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:contentDescription="@string/desc" />
  
  <ImageButton
      android:id="@+id/contact_entry_image"
      android:src="@drawable/ic_launcher"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:contentDescription="@string/desc" />
  
  <LinearLayout
      android:focusable="true"
      android:focusableInTouchMode="true"
      android:layout_width="0px"
      android:layout_height="0px" />
  
  <LinearLayout
      android:focusable="true"
      android:focusableInTouchMode="true"
      android:layout_width="0px"
      android:layout_height="0px" />
  ```

- **Enable focus-based navigation:** Make sure that users easily navigate through your screen by adding focusable components.

- **No audio-only feedback:** We should make sure that we have audio feedback as secondary to support users who are deaf or hard of hearing.

- **Custom view controls:** All custom interface controls built for your application must implement accessibility interfaces for all your custom views and make sure to provide content descriptions. Custom controls that are proposed to be compatible with
versions of Android back to 1.6, use the Support Library to implement all the latest accessibility features.

### 3.2.2 Usability

Designing the user interface is a very important feature for developing the application for mobile devices. There are many restrictions on mobile devices such as limited memory and processing power. In addition, understanding the behavior of the users is more important and essential. We need to make sure we consider each architectural layer and we should carefully prioritize them, so that we maximize physical capacity of the device. Designing user interface for mobile applications is not easy. Therefore, developing applications for mobile devices is challenging and rewarding in its outcome [22].

The most important challenge for designing UI for mobile applications is the small screen:

1. Small screens restrict the amount of data visible on the screen.
2. Limits our freedom of size and placements of graphical elements on the screen.

Some factors that impact performance of UI design are Memory, CPU and Handling server side data.

Some guidelines for UI design are:

1. Facilitate Frequency Uses: Time is very important to mobile device users. Thus, they desire to reduce interactions times and increase pace of interaction [23].
2. Dialogs: actions sequence should be organized into groups with beginning, middle, and end [23].
3. Reversal of Actions: it should allow easy reversal of actions [23].
4. Feedback: It should be some system feedback for each action such as —HTTP404 ERROR— and —THE PAGE CANNOT BE FOUND— [23].
5. Error Prevention and Simple Error Handling: have to take the physical devices into account [23].
6. Design for Small Device: mobile platforms will be smaller in size and include items such as buttons, and key chains, new techniques may be necessary to overcome the physical limitations [23].
7. Reduce short-term memory load: UI should use any little memorization during tasks performance [23].
8. Design for Top-Down Interaction [23].
9. Flexibility: any item of menu can easy move during other item is chosen. It is easy to return to that item again [23].
Many factors impact functionality and performance of mobile application. Such as available memory, CPU, and bandwidth capacity. We will discuss these factors in more details.

Memory considered as drawback in mobile application design. Today, there are some mobile devices that have memory capacity with respect to their size. For example, MIDP applications, which were developed, to work on a array of phones take into account the wide-ranging memory capacity of these devices.

CPU shows up analogous issues. Many developers adapted to work on a desktop environment to developed applications for mobile devices based on the processing capacity of CPU of today's desktop computers. Sometime this method becomes a careless programming style. Applications, which were developed based on desktop environment, would run very slowly on a typical mobile device. The solution here is to carefully analyze programming style such as analyzing loops in programs [24].

3.2.3 Reuse

Reuse is nothing but using one module of a product to facilitate and develop a different product with different functionality. On average, only 15% of new code serves an original purpose in principle, while 85% could be standardized and reused in practice [25]. Object Oriented reuse of code is mainly done through Inheritance and class reuse [26].

3.3 IMPLEMENTATION

SDSU School of art and design mobile application provides many functionalities for students to access information on the go. In this section we will discuss each of these functionalities in details.

When the app is launched for the first time after downloading, a screen is displayed which basically explains which permissions needed for this app to function properly. Permissions are a way of specifying what an application is allowed to do and what are all the resources are that application has access to. This can prevent users from installing some malicious applications on their phone. These permissions are specified separately from code in an XML file; this is a convenient way for the user to design. This way of declaring permissions in a XML file is called “provisioning” [27].
The Home screen as seen in Figure 3.3 is made as a TabHost layout, this is a way of implementing tab views in android. In our implementation of TabHost we have three views:

1. Home
2. Contact
3. SDSU

![Home screen](image)

Figure 3.3. Home screen.

Home screen is made of GridLayout, which consists of 6 grids now with 6 icons each occupying one grid. From the home screen the user may choose to access one of the following:

- Directory
- Events
- Gallery
- Students
Giving
Art Council

3.3.1 Directory

By choosing this, a user can get all the information about Faculty and Staff working at SDSU School of Art And Design. When clicked on the “Directory” icon is clicked, a call is made to PHP server asking for the data. In android all calls to an external server to get data should be made using AsyncTask. This is the way we can make sure the UI thread does not do heavy lifting. Using AsyncTask this will not block the UI thread and the data from the server is fetched using a background task. So the PHP script is called using HTTP GET. The script is run on the server and goes through the raw data and coverts into JSON format. The JSON data got back from server is in the form of

```json
{
    facultyDetails : [
        {
            Name: “Faculty 1”,
            Email: “axxx@abc.com”
        },
        {
            Name: “Faculty 2”,
            Email: “axxx@abc.com”
        },
        ………
    ]
}
```

This JSON object, once received by Android application, should be parsed to retrieve all the individual objects, which represent each faculty or staff record.

A generic class JASONParser.java does this job and returns java JSONObject and now we extract the array inside this object and store it in JSONArray.

To display the faculty list we use a new Activity, which consists of ListView Layout. The list view consists of Faculty/Staff picture, Last name and email id.

Figure 3.4 shows how the screen looks. The list is sorted alphabetically according to last name. Since there were a lot of records needed to be displayed. I had to come up with a design to load the images only when visible in the view
port. This mechanism of loading images only when needed is known as “LazyLoading”. This way initial loading of the directory list is fast.

Also in order to easily navigate in the list view, FastScroll is enabled with Alphabetical Section indexing, to easily jump to required alphabet.

The generic JSONParser.java is the most important class, which is used by all the activities making http request.

```java
public JSONObject makeHttpRequest(String url, String method, List<NameValuePair> params) {
    // Making HTTP request
    try {
        // check for request method
        if(method == "POST") {
            // request method is POST
```
// defaultHttpClient
DefaultHttpClient httpClient = new DefaultHttpClient();
HttpPost httpPost = new HttpPost(url);
httpPost.setEntity(new UrlEncodedFormEntity(params));

HttpResponse httpResponse = httpClient.execute(httpPost);
HttpEntity httpEntity = httpResponse.getEntity();
is = httpEntity.getContent();

} else if (method == "GET") {
    // request method is GET
    DefaultHttpClient httpClient = new DefaultHttpClient();
    String paramString = URLEncodedUtils.format(params, "utf-8");
    HttpGet httpGet = new HttpGet(url);

    HttpResponse httpResponse = httpClient.execute(httpGet);
    HttpEntity httpEntity = httpResponse.getEntity();
is = httpEntity.getContent();
}

try {
    BufferedReader reader = new BufferedReader(new InputStreamReader(is, "iso-8859-1"), 8);
    StringBuilder sb = new StringBuilder();
    String line = null;
    while ((line = reader.readLine()) != null) {
        sb.append(line + "\n");
    }

} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
} catch (ClientProtocolException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}

try {
    BufferedReader reader = new BufferedReader(new InputStreamReader(is, "iso-8859-1"), 8);
    StringBuilder sb = new StringBuilder();
    String line = null;
    while ((line = reader.readLine()) != null) {
        sb.append(line + "\n");
    }

} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
} catch (ClientProtocolException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
is.close();
json = sb.toString();
} catch (Exception e) {
    Log.e("Buffer Error", "Error converting result " + e.toString());
}

// try parse the string to a JSON object
try {
    jObj = new JSONObject(json);
} catch (JSONException e) {
    Log.e("JSON Parser", "Error parsing data " + e.toString());
}

// return JSON String
return jObj;

Figure 3.5 shows how the screen looks. From the directory list view, users can view details of the faculty by clicking on it.

Users can email the faculty members directly by clicking on the email id, and also view the faculty’s complete portfolio.

All the images in the application are loaded using URI. This portfolio needs all the images related to that faculty loaded into grid view. So there was a need for to load all these images asynchronously and cache them. Also there was a need to load all the images using multiple threads. The following code is an example for creating multiple threads [28].

// the run() method can simply be overridden...
Thread thread1 = new Thread("example1") {
    @Override
    public void run() {
        Log.i("thread1", "I like android");
    }
};

// ...or a Runnable object can be passed to the Thread constructor
Thread thread2 = new Thread(new Runnable() {
    public void run() {
        Log.i("thread2", "I like android 2");
    }
}, "example 2");
Figure 3.5. Faculty details and email activities.

```java
// remember to call start() or else the threads won't be spawned and nothing will happen
thread1.start();
thread2.start();
```

Also there was a need to cache images. This way we can avoid Out of memory exception and fetching images always from network is not only a waste of time but also resources. So implementation of a local cache was necessary. There are many local caching mechanisms. Like UnLimitedDiscCache, FileCountLimitedDiscCache and CacheSizeLimitedDiscCache [29]. Our android application makes use of CacheSizeLimitedDiscCaching to cache images. In this type of caching we will limit the cache directory size and if cache size is about to be exceeded the most recently used image file will be removed. Figure 3.6 shows multi threaded loading of all images into grid which are cached. These activities make use of `ImageView` layout for displaying images.
3.3.2 Events

From the home screen users can also retrieve all the events happening at SDSU School of Art and Design. When you click on the “Events” from home screen, a call is made to the PHP server to get back JSON data. This data is rendered by ExpandableListView. Each list view header corresponds to a type of event. The different kinds of events visible are: Student Exhibition, Visiting Artists, Student Events, University Exhibition and Student Deadlines. We get all the events happening in next 5 months starting from the current date. Figure 3.7 shows how the screen looks.

Users can get event details like Title, Presenter, Room, Date and time.

One can also subscribe for notification by clicking on the date. When you click on a date, a dialog pops up confirming users action. When user selects yes, a PendingIntent will be created and it will be populated with event details for execution some time in future.
When the time is up, broadcast is sent which will be received by `BroadcastReceiver` which will then kick off the `PendingIntent`. Notification shown in Figure 3.8.

### 3.3.3 Gallery

Users can view all the work done by Students in different programs by clicking on Gallery, which will display a list of Programs offered by the school and you can view the gallery related to that program.

Here too we needed to cache images. This way we can avoid Out of memory exception and fetching images always from network is not only a waste of time but also resources. So we made use of `CacheSizeLimitedDiscCaching` to cache images. Figure 3.9 shows multi-threaded loading of all images into grid, which are cached. These activities make use of `ImageView` layout for displaying images.
3.3.4 Students

Users can get all the information related to Students here. These are again displayed in ExpandableListView. This list is obtained from server with help of PHP.

- Student Advising
- Course Description
- Organization
- Scholarships
- Art Supplies

3.3.4.1 Student Advising

Here students can get the real time information about the advising hours for “Undergraduate Advisor”, “Art History Advisor”, “Graphic Design Advisor” and “Graduate
Figure 3.9. Gallery.
Advisor”. Students can also get details like Advisor, Office, Phone number, email id and advising hours. Figure 3.10 shows how the screen looks.

![Figure 3.10. Student advising.](image)

### 3.3.4.2 Course Description

Users can get all the course description by clicking on the level of course they want to explore. Whenever a level is selected a call is made to server asking for all the courses that fall under this level. Example:

http://abc.com/getCourses.php?level=100

List courses come back as JSON object, which are then rendered to **ListView**. Users can get details about each course by selecting individual courses. Details like Course Title, Units, Session in which this course is offered etc. Figure 3.11 shows how the screen looks.

Detail screen is displayed in **LinearLayout**. Figure 3.12 shows how the screen looks.
3.3.4.3 ORGANIZATIONS

Different disciplines within the School of Art and Design maintain active student organizations or active memberships in well-known professional organizations. Students are encouraged to participate in these organizations to promote professional development, growth, and networking within their respective fields of artistic practice. In many cases, there are important leadership opportunities in these groups as well. Users can obtain details about these organizations by selecting the organization. All of this data related to the organization is got from the database as JSON objects. When an organization is selected, the list view expands and shows all the available organizations. All the details about the organizations are stored in database as Wiki Markup. Figure 3.13 shows how the screen looks.
3.3.4.4 Scholarships

Some of the scholarships are administered through the SDSU Office of Financial Aid and Scholarships, and some of the scholarships are administered through the School of Art and Design. All the details about these scholarships can be found in this section. Details about all the scholarships are stored in database as Wiki Markup. To display it correctly in Android had to convert Wiki Markup to HTML. The details are displayed in a TextView with HTML rendering enabled. Figure 3.14 shows how the screen looks.

3.3.4.5 Art Supplies

This section is dedicated to endorsing local vendors for a couple of reasons: most students need materials fairly immediately, and because there are too many online resources to list them all. For all supplies, like large sheet metals and plastics, welding and woodworking materials/tools, roll fabrics, and photo equipment, users can select a category for a list of local vendors.
ExpandableListView renders this data with each list view header corresponding to the category of art supply the user is looking for. When selected, the data is retrieved from the server and displayed in ListView. A user can obtain details of each supply store by clicking on them.

Details will tell the address, phone number and types of materials they have and link to their website, and tells if this store offers student discount.

Again the data in database was stored as WikiMarkup, which had to be converted to HTML. Figure 3.15 shows how the screen looks.

The converted HTML is displayed in TextView, which renders HTML.

### 3.3.5 Giving

This section is dedicated for users who want to help support the School of Art and Design by making an online contribution. When clicked on, the user is taken to SDSU donation website which is displayed inside the app itself. The webpage is rendered within
3.3.6 Art Council

Students can participate in activities bridging the School of Art and Design with the larger Southern California community through the Art Council. They can know more about this by viewing this section. When clicked on the Art Council button, user is taken to SDSU Art Council website which is displayed inside the app itself. The webpage is rendered within WebView embedded in app, thus making it look like part of the app. Figure 3.17 shows how the screen looks.

3.3.7 Contact

User can access School Of Art and Design contact information by clicking on the contact tab. Information like Phone number, Email ID, Address, Hours, etc. All these information is got from the database in the form of JSON object and is rendered inside TextView.
Figure 3.15. Art supplies.
Figure 3.16. Giving.

Figure 3.17. Art council.
User can also get driving directions to School of Art and Design by clicking on “Google Maps Directions” link on this screen. Figure 3.18 shows a screen shot of Contact Info screen.

![Contact Info Screen](image)

**Figure 3.18. Contact info.**

### 3.3.8 SDSU

When a user clicks on the SDSU tab, SDSU webpage is opened inside the TabView. The webpage is rendered within WebView embedded in the app, thus making it look like part of the app. Figure 3.19 shows the SDSU webpage displayed inside the app.
Figure 3.19. SDSU.
CHAPTER 4

TESTING

One of the crucial steps in software development life cycle is testing. With the help of testing, we can make sure our application works correctly [30]. There are different types of testing mechanisms that can be used for software testing. Below, I discuss few testing types, which were adapted for my applications.

- **Unit Testing**: This test involves testing individual components of the software. The developer himself usually does this.

  For my application I tested all the individual components of the application, such as Gallery, Contact Info and Directory.

  Result: These individual units worked as intended.

- **Black Box Testing**: In this testing, the tester does not have any knowledge of the implementation details. He or she just tests the software based on the functionality and requirement.

  The end user who had no knowledge of implementation details was given an application to navigate through it to find details about course ART-512.

  Result: The end user was able to successfully navigate to course ART-512.

- **White Box Testing**: In this testing, the tester should have good knowledge of the implementation details.

  This process of checking if server returns the data in the format needed for the application.

  Result: Server returned the data in JSON format as expected by the application.

- **Integration Testing**: In this we check to see if the application is working correctly after individual modules are integrated.

  After combining all the individual modules as Directory, Events and Gallery. We tested the whole application to see if it works as expected.

  Result: When combined all the individual units, app functioned as intended.
CHAPTER 5

OBSTACLES AND RESULTS

5.1 CONVERSION OF WIKI MARKUP TO HTML

For building our Android application, we made use of database that was already existing for SDSU School Of Art and Design website. Many pages for this website were stored as Wiki markup and android applications do not have a way to render Wiki markup tags.

I had to come up with a generic parser that converted all the Wiki markup tags to HTML.

For example:

<table>
<thead>
<tr>
<th>Wiki Markup</th>
<th>HTML</th>
</tr>
</thead>
<tbody>
<tr>
<td>[br]</td>
<td>&lt;br/&gt;</td>
</tr>
<tr>
<td>== level 1 ==</td>
<td>&lt;h1&gt;level 1&lt;/h1&gt;</td>
</tr>
<tr>
<td>=== level 2 ===</td>
<td>&lt;h2&gt;level 2&lt;/h2&gt;</td>
</tr>
</tbody>
</table>

5.2 MULTITHREADED IMAGE LOADING AND CACHING

Since our application has many modules, which involves loading images using Internet. I had to come up with a design to start loading multiple images at the same time using multithreading. Once these images were loaded, we needed a mechanism to cache images. After exploring several caching mechanisms, I ended up implementing a local cache for this purpose.

5.3 LAZY LOADING IMAGES

In our application, for modules like Directory, Portfolio and Gallery, not all images need to be displayed at same time. We can defer image loading until the point at which it is needed; this way we can load all the images that are visible in view port first and load the rest of the images when they come into view port. This will make the application run faster.
CHAPTER 6

FUTURE ENHANCEMENTS

Any software application will always have room for enhancements; this can be considered as an ongoing process that has a never-ending cycle.

With all the requirements and time limitations, the application is functioning as we expected.

Currently, our application has all features one might use when accessing School of Art and Design website.

Some of the future enhancements that can be done include integrating our app with learning management system we currently have in place, Blackboard application. This way students will be able to login and access their grades and download assignments, etc.

Making use of all the server scripts and design documents, we can port our app to other platforms like iOS and Microsoft mobile as well.
REFERENCES

February 2014, n.d.

2014, n.d.

Croatian National Corpus, http://hnk.ffzg.hr/bibl/iti2008/PDF/(102)/102-03-156.pdf,
accessed February 2014, n.d.

(Special Interest Group) global development 4th annual workshop, 2011. Academia,
http://www.academia.edu/1553463/What_is_the_Potential_Impact_of_Using_Mobile
_Devices_in_Education by Penn State’s World Campus, accessed February 2014, n.d.

Encore Wiki, https://www.encorewiki.org/pages/viewpage.action?pageId=7445,

[6] J. Chen and J. Kinshuk, Mobile technology in educational services. ProQuest,
http://search.proquest.com.libproxy.sdsu.edu/docview/205847796, accessed March
2014, n.d.

id=1936334, accessed February 2014, n.d.

[8] Embedded Linux Wiki, Android architecture. Embedded Linux Wiki,


2014, n.d.


