DESIGN AND IMPLEMENTATION OF A NATIVE MOBILE MULTIMEDIA LEARNING APPLICATION FRAMEWORK ON ANDROID PLATFORM

By

KIEMUTE OYIBO (B.Eng. Electrical/Electronics)

Supervised by: Prof. Mohamed Hamada

Masters Thesis

Presented to the Department of Computer Science at the
African University of Science and Technology
In Partial Fulfillment of the
Requirements for the Award of

MASTERS OF SCIENCE DEGREE IN COMPUTER SCIENCE

African University of Science and Technology, Abuja – Nigeria

May 2013

Certification

AFRICAN UNIVERSITY OF SCIENCE AND TECHNOLOGY, ABUJA – NIGERIA
DEPARTMENT OF COMPUTER SCIENCE

This is to certify that the work described in this thesis entitled **Design and Implementation of a Native Mobile Multimedia Learning Application Framework on Android Platform**

By

KIEMUTE OYIBO (40233)

Was carried out in the Department of Computer Science under the supervision of:

_____
Prof. Mohamed Hamada – Supervisor

Date_____

_____
Prof. Mamodou Traore – Head of Department

Date_____

Dedication

I dedicate this work to the Almighty God,
Who gave me the wisdom and strength to start and
Complete this thesis, and to my late step grandmother
Whose tender hands first rocked the cradle, toiling night
And day to see her grandson among the shining stars of night!

Declaration

I, Kiemute Oyibo, the undersigned, an M.Sc. graduate student of the African University of Science and Technology (AUST) and the author of this thesis entitled "Design and Implementation of a Native Mobile Multimedia Learning Application Framework on Android Platform," hereby solemnly declare that this thesis is an original work done and prepared by me under the supervision of Prof. Mohamed Hamada in the Department of Computer Science, AUST. This work has not been previously presented as the basis for the award of any degree, diploma or similar title at this or any other university. The materials borrowed from other sources and included in my thesis have been properly acknowledged.

_____ _____

Student's Signature Date

Acknowledgements

I would like to express my deepest gratitude and appreciation to my supervisor Prof. Mohamed Hamada of the University of Aizu, Japan and African University of Science and Technology, Abuja, who gave me the motivation and impetus to kick-start this work, and guided me all the way. I would also like to thank the Head of Department of Computer Science Prof. Mamodou Traore of Blaise Pascal University, France and African University of Science and Technology, Abuja, and Dr. Kola Babalola of Computer Science, African University of Science and Technology, Abuja, for their academic and moral support.

My gratitude also goes to Prof. Wole Soboyejo, President and Provost of African University of Science and Technology, Abuja, as well as the entire staff of the University for their administrative support and encouragement. I would also like to thank specially the Petroleum Technology Development Fund (PTDF) for the postgraduate scholarship award, without which this thesis would not have been possible.

In the same vein, I would like to say a big Thank You to my fellow Computer Science students, Ph.D. and M.Sc. alike, who gave me support in one way or the other, namely, Arreytambe Tabot, Bright Ighoroje, Doyin Adegoke, Obaro Odiete, Hajara Abdulwahab, Yusuf Sahabi, Patrick Kalamula, Eustace Ebhothemen, Nila, Chongwe, Clement David, Emmanuel Onu, Emamurho Ugherughe, Ali Gombe and others whom I am unable to mention here.

Finally, I would like to thank my mother and father Mr. Joseph Esivwename Oyibo and Mrs. Victoria Oyibo, and my siblings—Mamede, Akpovi, Ese, Tobore, Rugba, Ufuoma and Eguolo—for the unflinching support, ranging from moral to financial.

Thank you, One. Thank you, All!

Abstract

Internet connectivity is one of the fundamental requirements for a successful mobile learning environment. However, within the context of Africa, availability and access, let alone cost, still pose a great challenge in higher education, especially in distance learning. Consequently, there arises a dire need for a native mobile learning application framework that would serve as an alternative to web-based learning environments in localized contexts such as Africa, where the problems of internet connectivity and bandwidth remain untackled. However, little body of knowledge exists on how such native application frameworks, which leverage the mobile device's underlying hardware resources and rich user interfaces (UIs) in offering a heightened learning user experience (UX), can be designed and implemented. As a result, this thesis sets out to bridge this gap by proposing a Native Mobile Multimedia Learning Application (NMMLA) Framework, implemented on the Android platform by using a systematic approach we called "Content Flow Algorithm Tree," which can be leveraged by mobile learning application developers in developing native applications for various higher education courses, especially in science and engineering. The framework is a one-page-setup and do-it-yourself toolkit and library that will facilitate the development of NMMLAs by reducing deployment time or time to market. Basically, the framework supports five (5) types of multimedia learning content—images, Hypertext Markup Language (HTML), audio, video and simulation—aimed at meeting the different needs of learners with different learning preferences. The framework provides a number of key features, which include theme, course, quiz and simulation menus; listview and tabview render modes; and Search and Help utilities. This work will benefit researchers and stakeholders in the m-learning field, especially Higher Education Institutions (HEIs), training and learning organizations. It will also benefit multimedia learning content developers and providers in general and on the Android platform in particular by preventing them from reinventing the wheel. Above all, it will benefit teachers, students and workers, especially distance learners, in the pursuit of life-long formal and informal learning, as they will be able to learn anywhere and anytime without internet connectivity and limited bandwidth being a barrier.

Keywords: mobile learning; multimedia content; learning component; learning module; reusable learning object, native mobile application; mobile learning framework; Android platform

Table of Contents

Page

List of Figures

List of Tables

Acronyms

AB – Actionbar
AD – Activity Diagram
ADT – Android Development Tool
API – Application Programming Interface
CP – Content Provider
DV – Detail View
DPI – Density-Independent Pixel Per Inch
FDM – Framework Data Model
GNU – GNU Not Unix
GPL – General Public License
GV – Gridview
HP – Homepage
HPFA – HomePageFragmentActivity
HTML – Hypertext Markup Language
HUI – Homepage User Interface
HEI – Higher Education Institution
IDE – Integrated Development Environment
JDK – Java Development Kit
JRE – Java Runtime Environment
LM – Listview Mode
LMFA – ListModuleFragmentActivity
LV – Listview
MLS – Mobile Learning System
NL – Navigation Level
NMMLA – Native Mobile Multimedia Learning Application
TV – Tabview
TCD – Technical Class Diagram
TFFA – TabFileFragmentActivity
TM – Tabview Mode
TMFA – TabModuleFragmentActivity
UCD – Use Case Diagram
UI – User Interface
UX – User Experience
XML–Extended Markup Language

Chapter 1
Introduction

The advent of Personal Digital Assistants (PDAs) and much later smartphones brought about a paradigm shift in the way, how, when and where we learn—from e-learning to m-learning [1], which fosters a much more personalized and self-directed learning. Upside Learning [2], while referring to mobile technology "as any device that is designed to provide access to information in any location, or while on the move," defines mobile learning as "the acquisition or modification of any knowledge or skill through the use of mobile technology, anywhere, anytime, resulting in the modification of behaviour."

Mobile learning has made great inroads worldwide into our way of life and every facet of our humanity, be it personal or professional, in a way some few years ago no one expected or ever imagined. For example, on the educational and organisational fronts, it has made such impact that you would hardly find a Higher Education Institution (HEI) teacher or student, a corporate employer or employee, both in developed and developing countries, without a smartphone—be it in the sitting room, bedroom, office, classroom, on the road, in the air or at sea. According to Heiphetz [3], it has impacted our lives to such a great extent that some of us are unable to leverage all of its benefits, which include but not limited to the following:

1. Makes content universally accessible anytime, anywhere
2. Adapts to student and employee needs (personalization)
3. Enables reflection
4. Is continuous, ongoing and flexible
5. Enables formal and informal learning
6. Increases knowledge retention and saves time
7. Encourages knowledge sharing and gathering
8. Readily available
9. Adapts to the needs of the organisation (academia and business)
10. Creates best practices

M-learning is made possible by mobile devices, mobile technology, mobile platforms and mobile applications, which come basically in three different forms: 1) a dedicated standalone application that can run on individual mobile devices; 2) a client-server model with the client application running on mobile device and a server application on remote server; and 3) a mobile web browser that requires back-end application-server connection in the course of sending requests from the mobile device [4, 5].

The introduction of the open-source Android platform [6] in 2007 by the Open Handset Alliance (OHA) pushed the frontiers of mobile learning further, owing to its openness, flexibility, and relatively low cost of developing and owning its applications, as opposed to the iOS, Windows

Phone 7 and other mobile development platforms [7]. Similarly, according to [8], "Android applications have none of the costly and time-intensive testing and certification programs required by other platforms such as BREW and Symbian." As a result, a large number of learning content developers and providers (e.g. Moodle, Blackboard etc) started taking advantage of it as a medium for delivering rich, interactive multimedia content to a wide range of learners with different learning preferences across different geographical locations and time zones. Consequently, mobile multimedia learning applications (native and web-based) abound in the marketplace today, as evident in Google Play. However, while there is a substantial body of knowledge on the design and implementation of mobile multimedia learning application frameworks for web-based applications, there is little or none on native applications, which can take advantage of the mobile device's underlying hardware resources and rich user interfaces in delivering rich multimedia learning user experience (UX) [9]. This research sets out to bridge this gap by providing a conceptual design of a NMMLA framework and implementing it as a library on the Android platform using an Object-Oriented Programming (OOP), Universal Modelling Language (UML) and Model View Controller (MVC) approach, Java programming language and the Eclipse Integrated Development Environment (IDE) with Android Development Tool (ADT) and other required development tools plugged in. The framework will help guide the process of mobile multimedia learning application development and facilitate future development on the Android platform.

The framework is made up of a number of components. A component, within the context of the framework, is represented by an icon (with certain functionality) on the Android device's screen. It is either hosted in the main body, called gridview (GV) or at the top of the screen, called actionbar (AB). Thus, the framework has two types of components, namely, gridview and actionbar. The former are the main components, while the latter are the support components.

The main (GV) components are grouped into five (5) major abstracted categories, which include `AtomicItem`, `TabFile`, `ListItem`, `TabModule` and `ListModule`, which content developers and providers can leverage in delivering a complete functional NMMLA, which include components such as Introduction, Learn, Simulate, Evaluate, Resources and Help in line with industry guidelines such as the Advanced Distributed Learning M-learning Guide [10].

The support (AB) components are further grouped into two: menu and action. The action components include `About`, `Search` and `Help`. They derive from the main components, with the last two offering utility services across an instance application. `About` is a HTML file which holds information about the instance application, which utilizes the framework. `Search` enables the learner to look up words in the dictionary included in the application. `Help` offers a list of HTML files, which provide information on the usage of various components of the application. `Search` and `Help` (represented by an icon and text) are pinned to the actionbar throughout the application UIs. On the other hand, the menu components include `Course`, `Theme` and `Render Mode` menus, which are pinned to the actionbar as well. They enable the framework to support multiple courses, themes and render modes respectively. `Theme` is

customizable by the content provider (CP) if he chooses not to use the default provided by the framework. For small–sized screen, the actionbar components may overflow to the menu at the bottom of the screen, while for large screens they may be rendered as a list of submenus at the right-hand corner of the actionbar. And for much larger screens, such as tablets, all the menu items may be rendered across the actionbar.

The NMMLA framework was designed by adopting an Object-oriented Programming (OOP) approach and the Universal Modelling Language (UML). For the modelling, Use Case Diagram (UCD), Activity Diagram (AD) and Technical Class Diagrams (TCDs) were used, while for the implementation, Java programming language, Eclipse IDE with plugged-in ADT and other required development and desktop/online asset-generating tools, were used. Moreover, the framework is backward compatible. By virtue of Android's `FragmentActivity` class [6] and the Actionbar Sherlock Library [11], the framework is capable of supporting both phones and tablets alike, ranging from API 8 (Android 2.2) to API 17 (Android 4.2) platforms.

This academic work will benefit stakeholders in the mobile learning field, be it in education, government or organization, in four major ways. First, it will help content providers on the Android platform in these sectors with little or no application development know-how to concentrate on the development of quality multimedia on any subject or course for learners, while relying on it as a reliable medium to deploy their content, thus reducing time to market. Second, it will provide NMMLA developers on any platform, especially computer scientists and aspiring techno-entrepreneurs, with the fundamental knowledge, skills, systematic design techniques and programming approaches needed for successful software application development. Third, in the context of forward engineering, the UML diagrams, provided by this thesis, offer a veritable design base, required for the successful development of future native multimedia learning applications. Fourth, it will help students (especially those living far away from the classroom), workers and individuals with different learning preferences have access to rich multimedia content of their choice and learn on the go without internet access being a barrier, especially on the African continent where poor or lack of internet connectivity and limited bandwidth continue to militate against the full adoption of e-learning and m-learning.

The rest of this chapter presents the background to this thesis as well as the aims and objectives. It goes further to state the research question(s), which this thesis aims to answer. Finally, it explains the thesis structure and organization by providing an insight into what the next chapters cover. However, before proceeding further, it may be useful to state some of the text-formatting styles adopted and assumptions made throughout this work. Standard programming constructs such as Java class and object names; framework class and object names; Android widget names and related keywords such as "Activity", "Search", "listview" etc are written in `Courier New` font. This is done so as to facilitate easy reading and understanding of the concepts presented in this material. Also, the CP in the middle of the m-learning value chain, for the most part, is referred to as "user", while the consumer at the tail end of the chain is referred to as "learner".

1.1 Background

Mobile learning, being an offshoot of mobile technology and the internet revolution, is one of the greatest wonders that have happened to humanity since the industrial revolution in the early 19th century [7]. It offers a veritable tool in man's endless quest for knowledge to provide answers to many of the yet unanswered philosophical and metaphysical questions, and find scientific solutions to the plurality of its problems, which range from social to medical and from physical to economic, just to mention a few. Similarly, today, like never before, the media for the acquisition of the required knowledge and skills to solve these problems have increased phenomenally and become raceless, spaceless and timeless. As a result, anyone, anywhere, anytime, any age (AAAA), can learn through whichever means and media he prefers, and come up with lasting solutions to any of the world's problems, as the world has become flat, and keeps flattening each day, due to the power of Information and Communications Technology [12], multimedia, and lately mobile technology and devices. According to [13], "mobile phones are misnamed." Rather, "they should be called 'gateways to all human knowledge'." Similarly, mobile technology was described by [2] as the most popular and widespread technology on the planet, which has become the most rapidly adopted technology in history, with the global mobile industry expected to grow to $1.9 trillion by 2015 from the current $1.5 trillion level. Furthermore, the global subscriber base and the number of mobile connections are projected to grow to 4.6 billion and 9.1 billion respectively by 2015. Finally, according to the multimedia principle [9], "People learn more deeply from words and pictures than from words alone."

On the mobile front, Africa is widely acknowledged as the world's fastest growing mobile market [14]. According to George Ferreira [15], Africa is the second largest and fastest mobile phone market in the world after China, with Nigeria, South Africa, Kenya and Ghana taking the lead in smart phone sales, adding that mobile device penetration grew from a base of 90 million in 2005 to a current estimate of 450 million handsets in 2012. In a survey carried out by a global market research firm, TNS, it was found that 25% of Nigeria's over 105 million mobile telephone subscribers use smartphones. Similarly, Tony Liangwei [16], projected 30 million smartphones were expected to be sold in Nigeria between now and 2015.

The statistics in the foregoing hold a huge potential for the African continent and Nigeria in particular on the mobile learning front: more and more people will eventually own a smartphone in the next few years to come, which will not only be used for basic communication or just-in-time learning but can support rich multimedia learning content as well. This behooves the African research community to leverage this great opportunity for the educational development of its citizenry as developed countries have done over the years [7].

1.1.1 Contextualized M-Learning

While internet access and high-speed connectivity can be taken for granted elsewhere in the world, here in Africa and Nigeria in particular, these twin necessities of an information age are still luxuries and continue to pose a great challenge for learners and teachers on the continent and in the country [7]. In cases and places where they are available, the cost of access or

download is still on the high side to the extent that most students and teachers cannot afford it [17]. Thus, as espoused by [17, 18, 19], there is need for the adoption and development of a contextualized learning framework or model that takes the infrastructural and technological limitations of these environments into consideration. Consequently, they called on learning content developers and providers on the continent and in the country respectively to endeavour to develop contextualized e-learning models and systems, which support and facilitate teaching and learning in these environments. This is necessary, more especially when, as [17] noted, full internet connectivity, in places where there is, is not uniform across all locations, and most distance learners neither attend residential sessions nor have the opportunity to have synchronous assistance in hard-to-understand portions of study modules because they are not living close to their teachers or classmates.

1.1.2 Mobile Devices and Application Development

A mobile device basically is a portable device that can be carried from place to place and used in enhancing day-to-day living such as communication, searching for information on the Internet, job opportunities, and above all, teaching and learning. According to [10], a mobile device is any device that:

1. Turns on instantly (don't require boot-up)
2. Is carried in a pocket or purse most all the time and are gaining ubiquity
3. Has sufficient power to last one day
4. Has input and output capabilities and a processor

It added that mobile devices are more than just a phone, as they come in different categories and sizes. Establishing that basic mobile phones are not suitable for m-learning, it identified and classified those that offer huge potential for m-learning in a logical way as shown in Table 1.1.

Table .1. ADL Classifications of Mobile Devices

S/N	Mobile Device	Capabilities
1.	Smartphones	System-On-Chip (SoC), full browser / HTML5 support, Wi-Fi, 3G/4G, music player, GPS, video-capable, Bluetooth, touch support, camera, accelerometer, 3D video acceleration, etc.
2.	Tablets	Same core features as smartphones, but larger screen sizes (e.g., 5', 7', 9') and optional keyboard, and no true phone.
3.	Non-phone Devices	Wi-Fi support, browser, other features, etc. iPod, PDAs, handheld game consoles, wearable devices, or portable media players.

1.1.2.1 Mobile Development Concerns

Usually, when developing for mobile devices, more issues than for desktops need to be addressed by the developer, as the mobile device has a lot of limitations compared to the desktop environment. According to [10], these limitations include battery life, connectivity, cost, data charges, device ownership, screen size, security and technology. Other concerns which need to be addressed as well by the developer include network, carrier, device and platform.

1.1.2.2 Screen Sizes, Resolutions and Densities

Unlike desktops and laptops, mobile devices have some restrictions such as limited battery life, small screen size and resolution. Besides, they come in different display sizes, densities and resolutions as evident in Android phones and tablets. Generally, Android devices come in four different display screen sizes, which include small, normal, large and xlarge. Consequently, the programmer must be prepared to tackle these issues if he wants his application to support (run seamlessly in) a wide range of mobile devices with different sizes and resolutions. Resolution is the number of physical pixels contained in a screen. However, it is more preferable for developers to work with screen density, which is the quantity of pixels within a physical area of screen, referred to as dpi (dot per inch). But then, during the definition of layouts and layout of widgets on the UI, a density-independent metric known as density-independent pixel (dp), which is a virtual pixel unit, is required. This metric allows layout dimensions or positions of widgets on the screen to be expressed in a density-independent way. As a standard, one (1) dp is equivalent to one (1) physical pixel on a 160 dpi screen, which is the baseline density assumed by the system for a medium density screen [20, 21]. Generally, Android devices come in four basic different screen densities—low density-independent pixel per inch (ldpi), simply referred to as low; medium density-independent pixel per inch (mdpi), simply referred to as medium; high density-independent pixel per inch (hdpi), simply referred to as high; extra high density-independent pixel per inch (xdpi), simply referred to as extra high.

1.1.3 Theory of Multimedia Learning

According to Mayer [9, 22], one of the fundamental hypotheses underlying research on multimedia learning is that multimedia instructional materials that are designed taking into consideration how the human mind works are more likely to result in meaningful learning than those that are not. He summarizes it in the *multimedia principle*: "People learn more deeply from words and pictures than from words alone." His cognitive theory of multimedia learning (CTML) is based on three cognitive science principles of learning, which include the following:

1. The human information processing system includes dual channels for visual/pictorial and auditory/verbal processing (i.e., dual-channels assumption); and
2. Each channel has limited capacity for processing (i.e., limited capacity assumption) and
3. Active learning entails carrying out a coordinated set of cognitive processes during learning (i.e., active processing assumption).

He advocates that multimedia instructional materials should be designed to meet the specific five cognitive processes in multimedia learning, which are depicted in Fig. 1.1 and as follows:

1. Selecting relevant words from the presented text or narration,
2. Selecting relevant image from the presented illustrations,
3. Organizing the selected words into a coherent verbal representation,
4. Organizing selected images into a coherent pictorial representation and
5. Integrating the pictorial and verbal representations and prior knowledge.

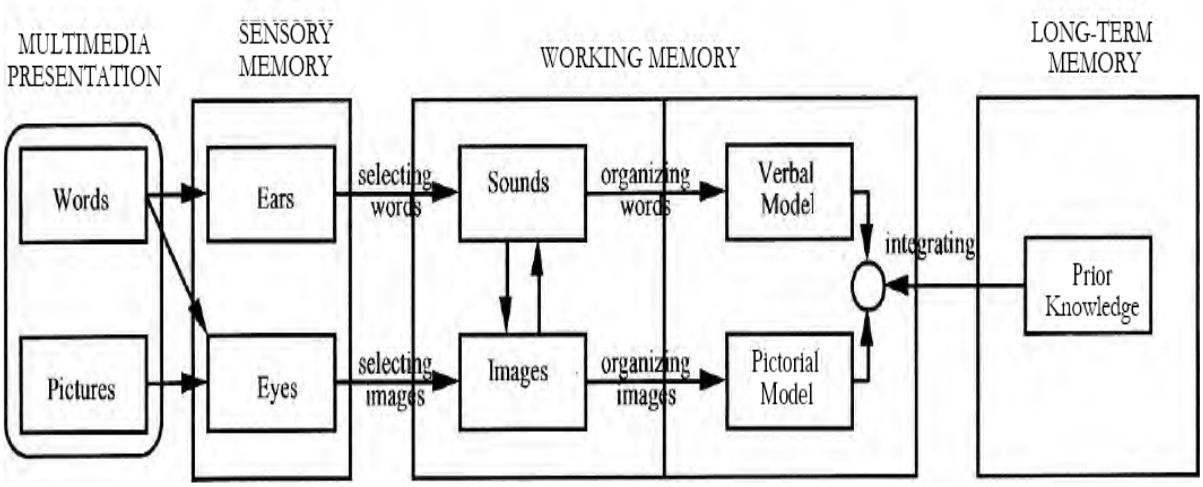


Figure .1. Cognitive Theory of Multimedia Learning

According to Pocatilu & Pocovnicu [23], an entertaining multimedia content has the potential of "transforming young students' learning from something that they *need* to do into something that they *like* to do." They gave a typical example as entertaining multimedia content which combines gaming and learning. See Table 1.2 for the other cognitive principles of mobile learning [24]. For the most part, the one in the bottom row builds on the other in the top row.

Table 1.2. Cognitive Principles of Mobile Learning

S/N	Cognitive Principle	Statement
1.	Modality Principle	People learn, retain, and transfer information better when the instructional environment involves auditory narration and animation, rather than on-screen text and animation.
2	Redundancy Principle	People learn, retain, and transfer information better when the instructional environment involves narration and animation, rather than on-screen text, narration and animation.
3.	Coherence Principle	People learn, retain, and transfer information better when the instructional environment is free of extraneous words, pictures or sounds.
4.	Signalling Principle	People learn and transfer information better when the instructional environment involves cues that guide an individual's attention and processing during a multimedia presentation.
5.	Contiguity Principle	People learn, retain, and transfer information better in an instructional environment where words or narration and pictures or animation are presented simultaneously in time and space.
6.	Segmentation Principle	People learn and transfer information better in an instructional environment where they experience concurrent narration and animation in short, user-controlled segments, rather than as a longer continuous presentation.

1.1.4 Learning Style Models

Just like humans are different, so are their learning styles and preferences. In this section, we will briefly look at two (2) of the most common learning style models: Neil Fleming's VAK/VARK Model and David Kolb's Learning Styles Model. This is necessary so as to put the objectives, which this thesis seeks to realize, into proper context.

1.1.4.1 Neil Fleming's VAK/VARK Model

The Neil Fleming's VAK/VARK model is used by learners to identify their preferred learning style in order to maximize their educational experience by focusing on what benefits them the most. It is one of the most common and widely-used categorizations of the various types of learning styles [25, 26, 27] based on neuro-linguistic programming (VARK) models. According to the model, as cited in [28], learners can be categorized into three (3) major groups: visual learners, auditory learners and kinesthetic learners or tactile learners.

1.1.4.1.1 Visual Learners: Visual learners, according to Fleming's model, have preference for seeing and visualizing what they learn. He noted that they think in pictures and prefer visual aids such as overhead slides, diagrams, flipcharts, handouts, videos, etc. This group of learners prefers sitting in front in the classroom so no one obstructs their view of the teacher and blackboard. For this group of learners, video, slides and HTML files, replete with diagrams and images, which the NMMLA framework supports, will be useful and beneficial [27].

1.1.4.1.2 Auditory Learners: This set of learners, according to Fleming, best learns through listening to audio content such as lectures, discussions, tapes, etc. The accommodation of audio content in the framework is targeted at meeting the need of this set of learners [27].

1.1.4.1.3 Tactile/Kinesthetic Learners: This group of learners prefers to learn by experience, i.e. through a hands-on approach, e.g. feeling, touching, moving and doing things. They prefer being actively involved in exploring the world around them and engaging in scientific experiments [27]. The NMMLA framework attempts to accommodate this group of learners by supporting swiping and sequencing through content and the delivery of interactive content such as simulations, which learners can interact with very closely so as to realise a heightened UX.

1.1.4.2 David Kolb's Learning Styles Model

David Kolb's [29] learning styles model gave rise to the Learning Style Inventory (LSI): an assessment method used to determine an individual's learning style. Based on the Experiential Learning Theory (ELT), it is one of the most widely accepted models with substantial empirical support. According to this model, which identified two pairs of related approaches towards *grasping* and *transforming* experience, namely, Concrete Experience/Abstract Conceptualization and Reflective Observation/Active Experimentation respectively, the ideal learning process engages all four of these modes in response to situational demands [28]. Thus, in order for learning to be effective, all four of these approaches must be actively involved and integrated. However, as individuals attempt to use all four approaches, the model postulated, they tend to

develop strengths in one experience-grasping approach and one experience-transforming approach. Consequently, the resulting learning styles are combinations of the individual's preferred approaches. These learning styles include converger, diverger, assimilator and accommodator [30].

1.1.4.2.1 Convergers: Convergers are characterized by abstract conceptualization and active experimentation. They are good at making practical applications of ideas and using deductive reasoning to solve problems. The support of a simulation component in the NMMLA framework will help this group of learners to actively engage in simulation activities on a mobile phone prior to having the opportunity for live experiments.

1.1.4.2.2 Divergers: Divergers tend towards concrete experience and reflective observation. They are imaginative and are good at coming up with ideas and seeing things from different perspectives. The NMMLA framework provides for this group of learners likewise by allowing them to interact with the multimedia learning application on the Android device through clicking, swiping, zooming and sequencing through their learning content, which includes HTML files, images, audio, video, simulations and quizzes.

1.1.4.2.3 Assimilators: Assimilators are characterized by abstract conceptualization and reflective observation. They are capable of creating theoretical models by means of inductive reasoning. Textual learning content, delivered in HTML format, supported by the framework, will be suitable for this group of learners.

1.1.4.2.4 Accommodators: Accommodators prefer to have concrete experience and engage in active experimentation. They are good at actively engaging with the world around them and actually doing things instead of merely reading about and studying them. The simulation component accommodated by the framework will be beneficial to this group of active learners.

1.1.5 Framework Overview

A review of existing literature revealed that not much research has been done on the process of developing a NMMLA framework on the mobile platform. As such, as earlier stated, this thesis attempts to come up with a conceptual design and implement it as a library on the Android platform. The framework leverages the underlying hardware resources, such as audio and video players, accelerometers etc, in delivering interactive multimedia-rich content for various educational and training courses. Fig. 1.2 shows a schematic of how the framework can be instantiated to realize a functional application. The CP loads the content from the IDE filesystem into the framework, compiles it into a ".apk" format and pushes it to the Android device.

1.1.5.1 Multimedia Support

Due to the different learning preferences and styles of learners, the framework is designed to support various types of interactive multimedia learning content including quizzes, which enable learners to evaluate themselves upon finishing taking a modular content. Fig. 1.3, from content standpoint, portrays the types of multimedia, which are supported by the NMMLA framework.

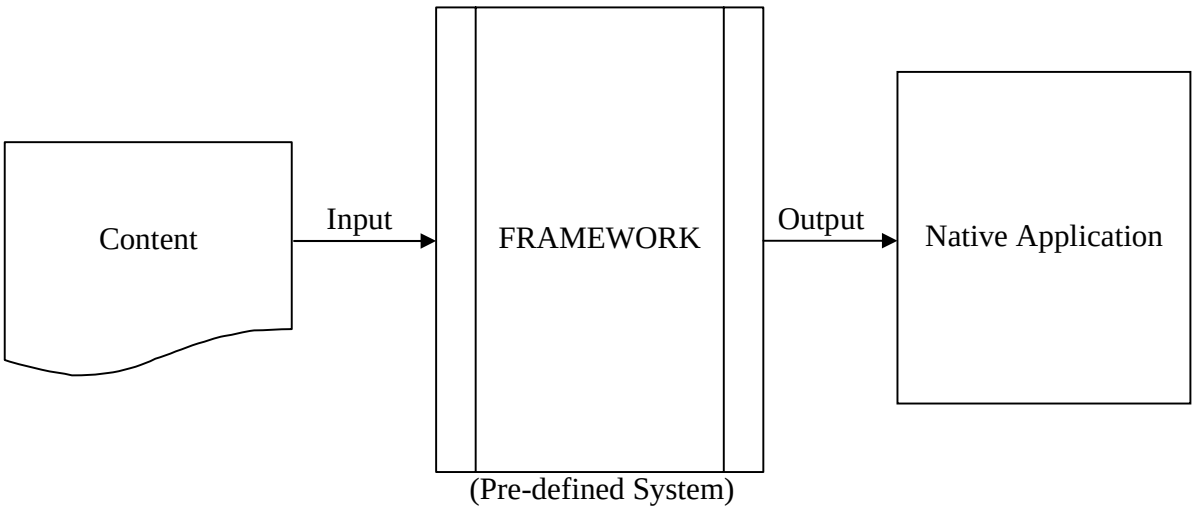


Figure 1.2. Instantiating the Framework to Realize a Native Mobile Application

They include text, HTML, images, audio, video and simulations (graphics and animations). These contents, for the most part, except for audio which plays on the background, are embedded or rendered in a view widget on the screen, e.g., `webview`, `imageview`, `videoview` etc. Table 1.3 depicts the various types of audio, video and image multimedia formats supported by the framework and the Android platform [6]. Consequently, the framework can be said to support the needs of verbal, visual and tactile learners, which can be regarded as the major kinds of learners as outlined by Fleming model. HTML, video and simulation are targeted at meeting the needs of verbal, visual and tactile learners respectively. In general, learners can interact with the content they are studying/learning by clicking, swiping, paging etc.

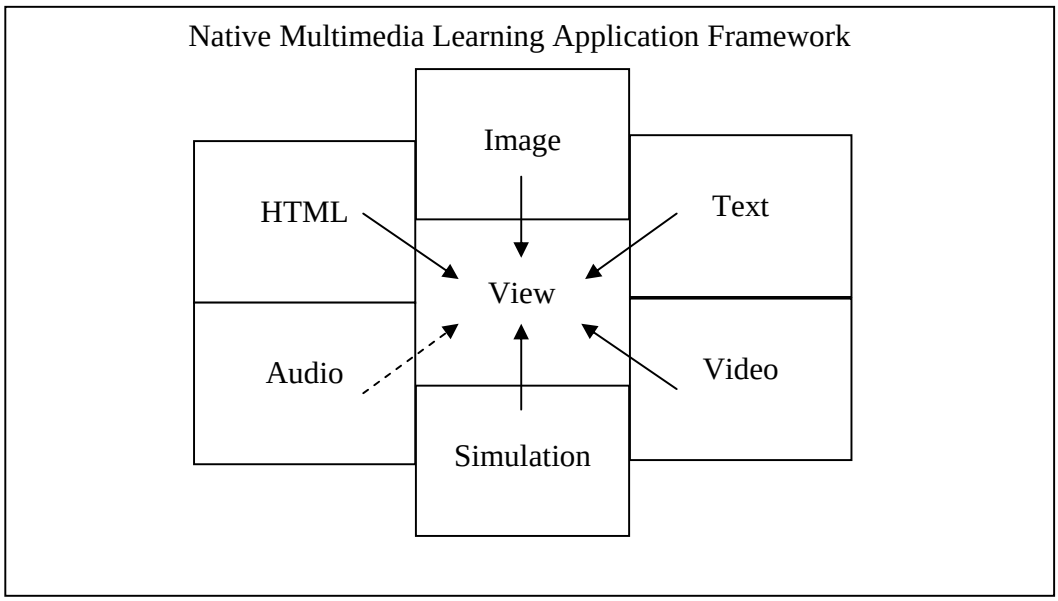


Figure 1.3. Types of Content Supported by the NMMLA Framework

They can also carry their multimedia content anywhere they go. This is a fundamental deviation from the traditional learning model, instructor-led training, where the learner was passive, confined in space and time, or in situations whereby he is able to carry his content (reading materials and textbooks) wherever he goes, weight and non-portability pose a great challenge.

Table 1.3. File Formats Supported by Framework

Types	**Audio**	**Video**	**Image**
Supported File Type/ Container Formats	3GPP (.3gp) MPEG-4 (.mp4, .m4a) ADTS raw AAC (.aac, decode in Android 3.1+, encode in Android 4.0+, ADIF not supported) MPEG-TS (.ts, not seekable, Android 3.0+)	3GPP (.3gp) MPEG-4 (.mp4) MPEG-TS (.ts, AAC audio only, not seekable, Android 3.0+)	JPEG (.jpg) GIF (.gif) PNG (.png) WebP (.webp)

1.1.5.2 Framework Use Case Diagram

Fig. 1.4 shows the UCD for the NMMLA framework. It captures the main actors (content provider on the right and learner on the left) and the overall components in the framework, which are abstracted. The abstract nomenclature is indicative of their render modes (RMs), which include listview mode (LM) and tabview mode (TM) for modular components, and detail view (DV) for atomic components. These components, represented by intuitive icons, are laid out either in the homepage (HP) gridview (GV) and at the top of the screen (actionbar) in an instance application. For small–sized screens, the actionbar components may overflow to the menu at the bottom of the screen, while for large-sized screens they may be rendered as a menu at the right-hand corner of the actionbar. Better still, for larger screens like tablets, the components are spread out along the actionbar. Table 1.4 shows the Framework Component Grid (FCG). It outlines the features of the various components supported by the framework, including their navigation levels (NLs). The dark orange color shows that the modular component can render in TM when clicked. The same explanation holds for the modular component with mid-orange color. The light orange color shows that the modular component is composed of items in a list. Similarly, the blue color indicates that a file is an atomic item (either in the HP GV or on the AB) or a tabbed detail view or an item in a list. The same applies to the light blue color for sim (simulation). NL indicates the number of clicks (see Fig. 1.5 [6]) required to get to the detail view of each item contained in an atomic or modular component [7].

The CP loads the framework with the required content from the right, while the learner consumes it from the left. As shown, the entry names for all items of content are stored in a database, such as SQLite database, while their corresponding files (image, HTML, video etc) are stored in the CP's project filesystem in the IDE [7].

Now, let us take an example of how the learner can interact with the framework components, and how this interaction can be mapped to the component grid in Table 1.4.

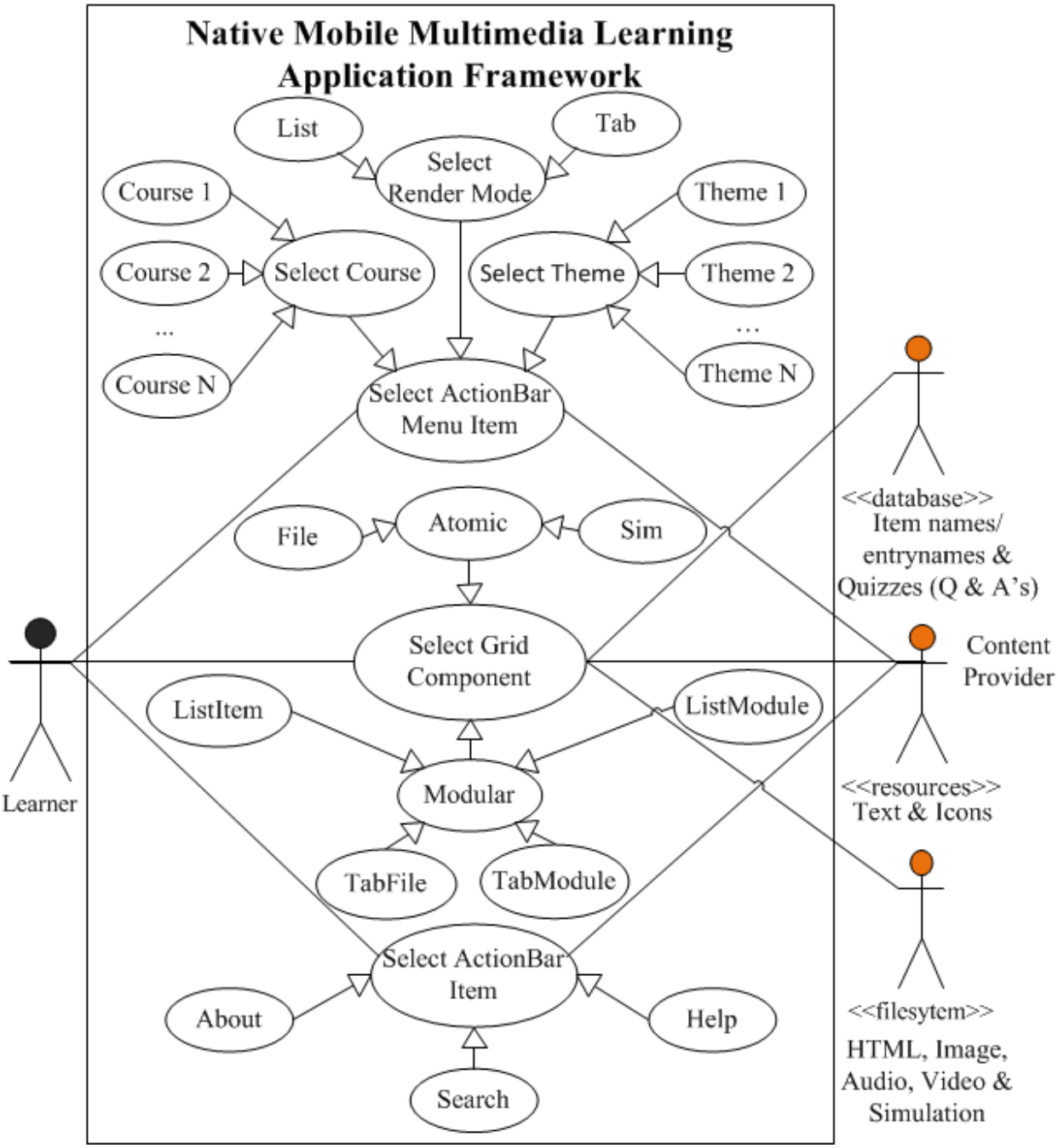


Fig. 1.4. NMMLA Framework Use Case Diagram from Component Standpoint

To illustrate this, let us say, as an example, he wants to study a module in Course 2. As a result, he selects Course 2 from the Course drop-down menu. Course 1 (default) pops off and Course 2 comes up on the screen, displaying its components in the HP GV and on the AB. The learner prefers to study a textual module, contained in a Learn component comprising a number of modules of HTML items (just like the chapters of a book having different topics) [7].

Table 1.4. Framework Component Grid

Component	**Features**					**Other**
Name	**Tab**	**List**	**Item**	**File**	**Sim**	**NL**
AtomicItem						1
TabFile						1
ListItem						2
TabModule						2
ListModule						3

However, he does not prefer the default render mode (LM) preset by the CP. As a result, he proceeds to select his choice (TM) from the RM options menu. And then, he goes on to click Learn. At this, a navigational tabview (dark orange in the FCG) opens, displaying the items of the first module in the component in a listview (mid-orange in the FCG). From this list, he can select an item (light orange in the FCG), which opens up the detail view of the corresponding file (dark blue in the FCG). All of these took only just two clicks or NLs (mid-purple in the FCG) to arrive at the detail view, as against three if he were in LM. Basically, the RM allows the learner to decide whether to swipe or click through his content, e.g. HTML files [7].

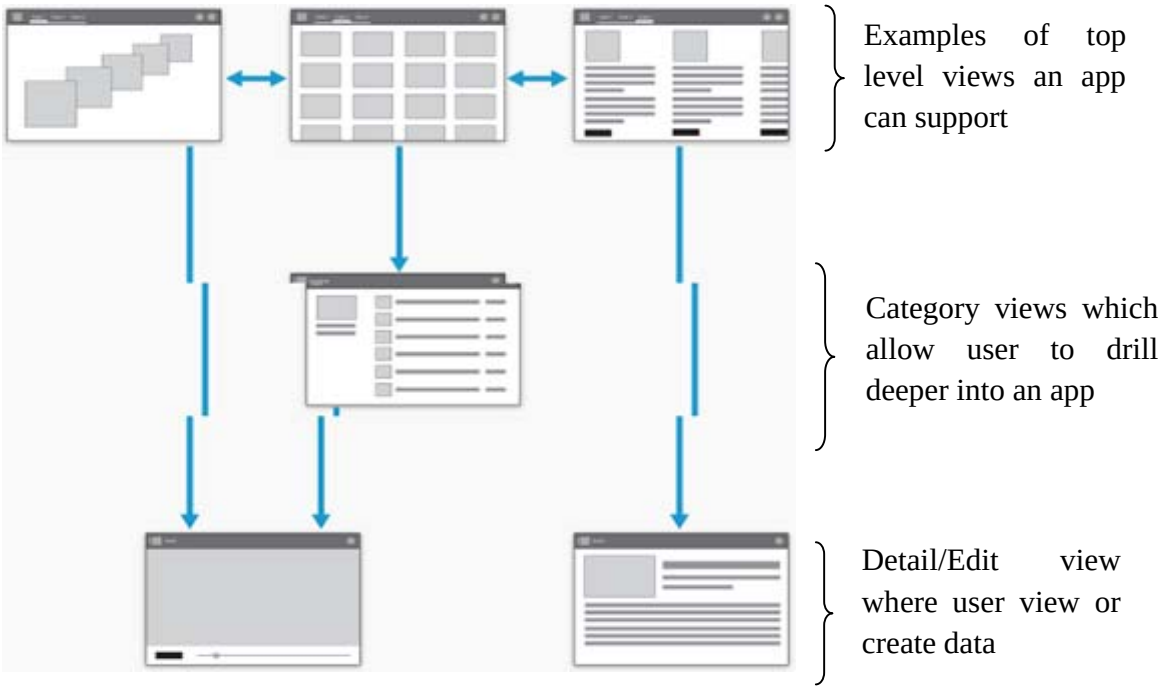


Figure 1.5. Framework Navigation Hierarchy

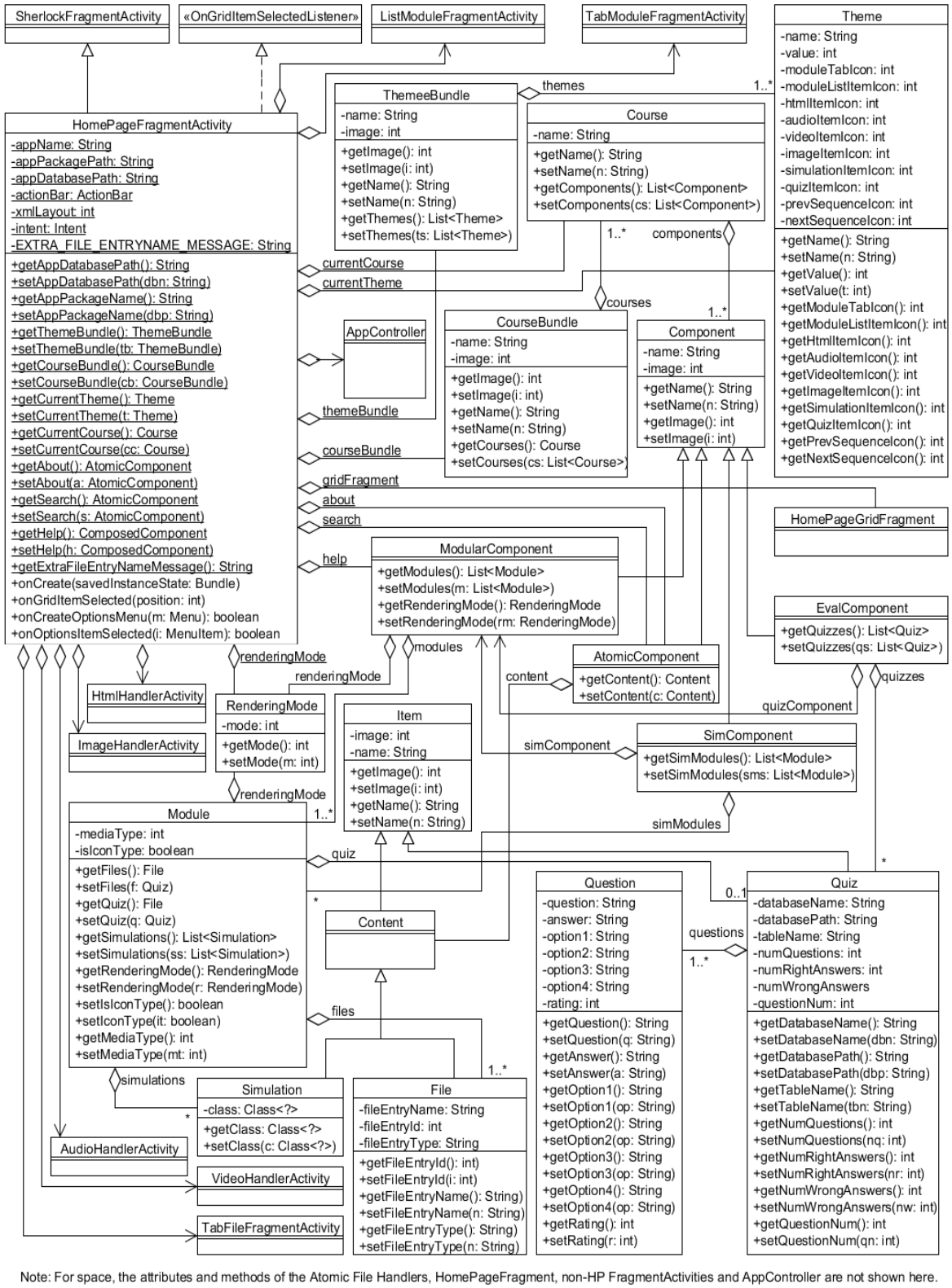
Moreover, the Course menu can also be used by content providers to organize courses according to levels of difficulty such as "Basic," "Intermediate," and "Advance [7]."

1.1.5.3 Framework Model View Controller

The Model View Controller (MVC) is a programming paradigm or practice leveraged by programmers to organize projects (especially large projects) in a standard way. Fortunately, in Android programming, a large part of it is already implemented by the Framework [31, 32]. We will briefly present the MVC, starting with the model (see Fig. 1.6), which our framework utilizes in realizing a complete functional NMMLA. Moreover, while discussing the controller, we present our Content Flow Algorithm Tree, which provides a visual flow of content through the framework and between the framework controllers in an instance application [7].

1.1.5.3.1 Model: Model is the domain-specific representation of the data (content) on which the application operates. In our framework, a model is a data structure that is used to retrieve data from a database or other data sources such as the Android project filesystem in an IDE. Fig. 1.6 shows the UML TCD for the design and composition of the data models (atomic and modular) which the framework supports. It depicts the static relationships (mainly inheritance and aggregation) between the various data models, such as `Course`, `Component`, `Module`, `Item`, `File`, `Simulation` etc, which make up a NMMLA. Examples of `File` objects include images, HTML, audio and video, which are specified by the "fileEntryType" attribute in the `File` class. Similarly, examples of `Simulation` include dynamic classes, which are characterized by graphics and animations. Also depicted in the TCD is the Homepage class, implemented as `HomePageFragmentActivity` (`HPFA`). This class inherits from the Android's `FragmentActivity` class and implements the interface `onGridItemSelectedListener`. The implementation of this interface defines what happens when a component in the HP GV is clicked. Moreover, the inclusion of the HP class alongside the data models helps in portraying how the HP of the NMMLA relates with other components, such as `CourseBundle`, `ThemeBundle`, `Help`, `Search` etc. It also portrays how HP relates with `intent`-receiving classes such as `ListModuleFragmentActivity` (`LMFA`), `HtmlHandlerActivity` (`HHA`) etc. These classes, referenced by a diamond-ended arrow, come up on screen when invoked, i.e. sent an `intent` by `HPFA`.

Another very useful piece of information captured in the diagram is how the `Module` class relates with `Quiz` class. We see in the diagram that for every module composed of items there can be one or no corresponding quiz as desired by the CP. Finally, we see that a module can have a render mode, LM or TM, which determines how its composed items will be rendered on the screen. This allows the learner to have and make choices according to his learning style and preferences. Similarly, a modular component can have a render mode, LM or TM, which determines how its modules will be rendered on the screen. Finally, the HP class also has a render mode static attribute, which determines in the overall how the modular components laid out on the HP screen will be rendered when selected (clicked) by the learner [7].



Note: For space, the attributes and methods of the Atomic File Handlers, HomePageFragment, non-HP FragmentActivities and AppController are not shown here.

Figure. 1.6. Framework Data Model Class Diagrams

1.1.5.3.2 Controller: In MVC pattern, the controller which, in most implementations, sits between the model and the view, defines the behavior of the application, maps user actions to model update, and responds to user gestures such as click, swipe etc [30, 31]. Basically, the framework comprises eleven (11) controllers, excluding simulation classes (shown in a broken-line box), which are simply RLOs, within the context of this framework, to be provided by the CP. The controllers are categorized according to functions as follows:

1. Course Loader (Root)
2. Component Router (Trunk)
3. Module/Item Dispatchers (Branch)
4. AtomicFile Handlers (Leaves)

This hierarchy clearly realizes our conceptual Content Flow Algorithm Tree (see Fig. 1.7), a communication network, which originates from the CP's application setup controller class (root), where content is uploaded using the APIs provided by the Homepage controller, and culminates in the framework's detail-view handlers (leaves), where the RLOs are rendered for the learner to study and interact with. The controllers were implemented in Android using both regular `activity` and a special `activity` called `FragmentActivity`, which by virtue of the usage of the Actionbar Sherlock Library have the word "Sherlock" pre-appended to them.

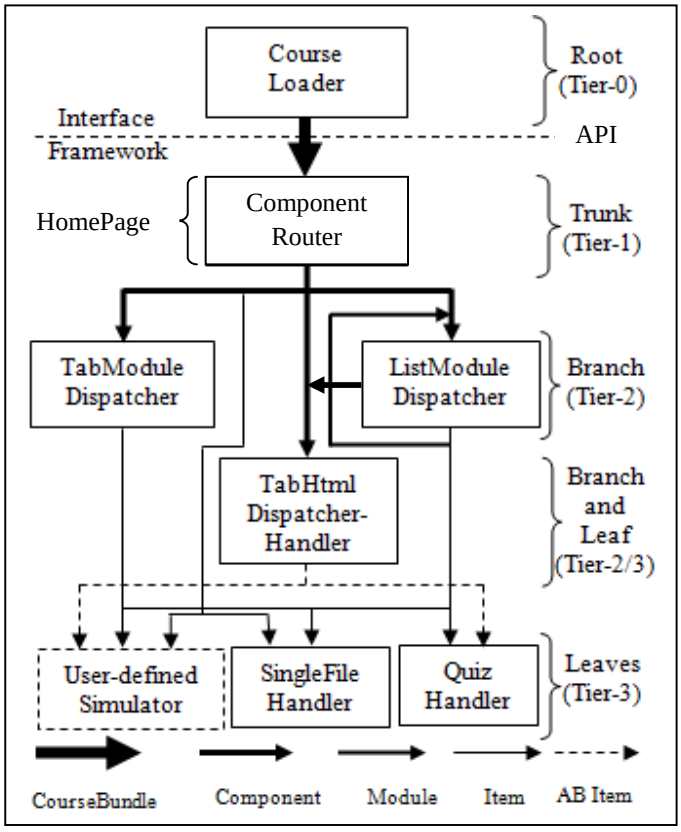


Figure 1.7. Framework Content Flow Algorithm Tree

1.1.5.3.2.1 Course Loader: This is where the instance application content is fed into the framework. It is the root of the CFAT, responsible for fetching learning content (RLOs) from the application's databases and filesystem (like nutrients from the roots of a tree), which are finally rendered as detail views (at the leaves) after traversing the component router (trunk) and module/item dispatchers (branches). This was implemented in Android as a user-named (setup) activity, which also presents a splash screen before opening the HP.

1.1.5.3.2.2 Component Router: It is the main UI which offers APIs to receive the application content (mainly a course and a theme bundle). On receipt of the course bundle, it unbundles it and renders the components of the first course on the screen as icons in a GV/AB. Then, it routes their flow across the framework (to Tier-2 or directly to Tier-3 controllers) upon learner's events and depending on the component's RM. We implemented this controller as `HomePageFragmentActivity` (`HPFA`) by extending Android's FragmentActivity class.

1.1.5.3.2.3 Module/Item Dispatchers: They are so-called because they do module or item, or both, dispatching. They include ListModule, TabModule and TabFile controllers. Generally, they are Tier-2 controllers, which receive a component with one or more modules and dispatch the composed items to atomic file handlers, except for TabFile, which is also a file handler (i.e. Tier-3) by virtue of receiving a one-module component from both component router and ListModule dispatcher and rendering the composed files in tabbed DVs. Specifically, ListModule performs both roles: dispatches modules (to self) and items to atomic file handlers. We used FragmentActivity in Android to implement them.

1. *TabFileFragmentActivity (TFFA):* It can be referred to as a self-contained dispatcher as it both receives a component and renders files in DV (reason being Tier2/3). It operates on a single-module TM component comprising a list of files, sent from the router (trunk) or ListModule dispatcher (branch) via an `intent` [6] and renders the files in navigational tabbed detail views (leaves). It also responds to and handles learner's gestures, e.g. swiping. The broken lines indicate navigating from AB item, e.g. quiz, to a detail view.
2. *TabModuleFragmentActivity (TMFA):* It operates on a component, comprising modules, with TM, sent from the router or trunk, and renders their composed items in navigational tabs of listviews. It is an item dispatcher. It does this by responding to and handling learner's gestures, e.g. sending a file item in any of its tabs of listviews to an atomic file handler to render it in DV when the learner clicks on it.
3. *ListModuleFragmentActivity (LMFA):* It is both module and item dispatchers. It operates on a component, which is composed of modules or a single module composed of items, with LM, sent from the component router or self respectively, and renders the modules or items in a listview. It also responds to learner's gestures, e.g. a click on a module or an item in a cell, by sending the clicked module to self or item to the appropriate atomic file handler at the leaves.

1.1.5.3.2.4 AtomicFile Handlers: They handle atomic files, such as HTML, image (or slide), audio and video, received from the dispatchers and render them on the screen as DV at the leaves. We used regular activities in Android to implement them.

1. *HtmlHandlerActivity (HHA):* It treats and renders HTML files in webviews. It also handles sequencing back and forth through the files belonging together in a module.
2. *ImageHandlerActivity (IHA):* It treats and renders images in imageviews and handles sequencing through them.
3. *AudioHandlerActivity (AHA):* It renders and plays audio files of different formats supported by Android in media player in the background, and handles sequencing through the files belonging together in a module.
4. *VideoHandlerActivity (VHA):* It plays video files of different formats supported by Android using a media controller. It renders them in a videoview, and handles sequencing through the files belonging together in a module.

1.1.5.3.2.5 Quiz Handlers: They were implemented in Android as three regular activities. They handle and render questions, score and answers respectively during quiz sessions.

a) *QuestionActivity (QA):* It is responsible for rendering questions and options on the screen. It also handles sequencing through questions back and forth during a quiz.
b) *ScoreActivity (SA):* It renders the learner's quiz's score on the screen, after which the learner can proceed to AnswersActivity to check answers to the attempted questions.
c) *AnswersActivity (AA):* It renders quiz's answers on the screen after the learner has finished taking a quiz. It also provides him with the option to retake another quiz or return to the activity where he left off, examples of which are those displaying a list of quizzes (or modules) and item's detail view during pre-and-post evaluation respectively.

1.1.5.3.3 View: The view is what the learner sees. It is responsible for rendering the model on the screen in a visual form known as widgets suitable to interact with. In a native Android application, the view includes `listview`, `imageview`, `textview`, `buttons` etc. Usually, the `activity` holds these views, which are laid out in and loaded from an Extended Markup Language (XML) file stored in the layout folder. These views can be populated with content pulled from within the code, IDE project's filesystem (such as `res` folder) and SQLite database (using data models) [6, 7, 31, 32].

1.2 Aims and Objectives

This thesis aims to design and implement a NMMLA on the Android platform that will facilitate the development and deployment of multimedia learning content, reduce time to market, and help students and workers learn on the go without the lack or cost of internet connectivity and limited bandwidth being a setback. As a result, this thesis sets out to achieve the following:

1. Propose and design a mobile multimedia learning application framework that supports listviews, tabviews, HTML, images, audio, video, simulations, multiple themes, and the

delivery of multiple as well as a wide range of educational courses in different fields of knowledge using a systematic approach that can guide the process of mobile multimedia learning application development in the future.

2. Implement the framework on the Android platform using an Object Oriented Programming (OOP) and Model View Controller (MVC) approach, Java programming language and ADT plugged into Eclipse IDE.
3. Realise a complete working Native Mobile Multimedia Learning Application (NMMLA) that can support both Android phones and tablets.

1.3 Motivation for Choosing Native Application and Android Platform

The review of existing literature on mobile learning reveals that a greater part focuses on web-based multimedia learning applications, frameworks and environments, paying little attention to their native counterparts and the role they play in mobile learning, especially, given the fact that not all students or employees or individuals—for example, on a continent like Africa where most people still live below the poverty line—having being able to procure a smartphone, say for example, a Samsung Galaxy Gio S5660, which on the average goes for nothing less than $130 [33], could afford internet connectivity in order to leverage the benefits of mobile learning in their academic pursuit, personal or professional life. Similarly, for those who are able to afford connectivity, speed and bandwidth become an issue. Many factors have been identified as responsible for the slow growth of broadband internet services in Africa. According to Dowuona [34], the 2012 West and Central AfricaCom Conference in Dakar, Senegal, "attributed the sluggish growth of broadband Internet in Africa to high cost of bandwidth and unfriendly regulatory environment." Thus, in an environment like Africa, there is need to encourage the development of native multimedia learning applications and frameworks that facilitate the development and delivery of rich, interactive and highly engaging multimedia learning applications, as such applications require one-off download or installation from a local repository or marketplace such as Google Play, thereby reducing or removing the need for continuous internet connectivity, which, for the most, is unaffordable. This will allow a large number of learners without internet access or continuous internet access to take advantage of mobile learning and the benefits it brings in the development of the human mind, without connectivity or bandwidth posing a challenge. In this light, the mobile phones, equipped with the right amount of storage and multimedia capabilities, can be said to truly replace the traditional textbooks and study materials. However, we encourage the inclusion of internet resources (e.g. through hyperlinks) NMMLAs as well. This will allow for a more frequently updatable content and enable learners with internet connectivity to access these resources anywhere and anytime.

On the other hand, Android was chosen as the platform of choice for the implementation of the framework because of its openness, flexibility and the ubiquity of its mobile devices (phones and tablets alike) around the world. According to Hanafi and Samsudin [35], mobile applications developed on the Android platform are more efficient and effective compared to other platforms such as Windows, Symbian etc. They argued that such applications produce faster, more user

friendly and appealing applications. Moreover, Android is becoming more and more ubiquitous and patronized than any other mobile operating system (OS) or platform in the world. According to Gartner [36], in its quarterly report on the global smartphone sales based on OS for the last quarter of 2012 (see Table 1.5), Android increased its market share in the same period from 51.3% to a commanding 69.7%.

Table 1.5 Gartner's 2012 4Q Report on Global Smartphone Sales by Operating System

Worldwide Smartphone Sales to End Users by Operating System in 4Q12 (Thousands of Units)

Operating System	4Q12 Units	4Q12 Market Share (%)	4Q11 Units	4Q11 Market Share (%)
Android	144,720.3	69.7	77,054.2	51.3
iOS	43,457.4	20.9	35,456.0	23.6
Research In Motion	7,333.0	3.5	13,184.5	8.8
Microsoft	6,185.5	3.0	2,759.0	1.8
Bada	2,684.0	1.3	3,111.3	2.1
Symbian	2,569.1	1.2	17,458.4	11.6
Others	713.1	0.3	1,166.5	0.8
Total	**207,662.4**	**100.0**	**150,189.9**	**100.0**

Source: Gartner (February 2013)

1.4 Research Questions

The main thrust of this thesis is to address the fundamental research question posed as follows:

> *"How can African students with different learning preferences learn anywhere and anytime without the cost or lack of internet connectivity being a barrier?"*

By extension or elaboration, the following research questions readily derive from the above:

1. How can teachers in HEIs and training organizations deliver their teaching and training materials to those students far away from the classroom and studying workers who are always on the move without cost or lack of internet connectivity and limited bandwidth being a barrier?
2. How can learning content providers deliver rich and interactive multimedia learning content on a ubiquitous mobile platform with little or no technical know-how of the required Software Development Kits (SDKs), programming knowledge or skills?
3. What are the steps, procedures, tools and software development methodology and techniques required for the successful design and implementation of a native mobile multimedia learning application framework, from which applications can be instantiated in order to reduce development time and time to market?

1.5 Thesis Structure

The rest of this thesis is organized into five (5) chapters: Chapter 2 to Chapter 6. Chapter 2 focuses on software frameworks and reviews of a number of them as well as existing mobile learning frameworks and environments. Chapter 3 zeroes in on the methodology. It discusses the framework requirements, design and implementation approaches used. Chapter 4 dwells on the implementation of the framework by focusing on the tools used as well as the framework packages and their constituent classes. It goes further to explain how the framework functions, focusing on the data and control flow and how content is rendered in different views. Chapter 5 presents the outcomes (results) from the design and implementation of the framework. It portrays how the framework can be instantiated and leveraged in realizing mobile multimedia learning applications. Chapter 6 focuses on the conclusion, contributions and challenges encountered in the course of realizing the framework. In concluding, it wraps up by giving a perspective on possible future research.

1.6 Expected Contributions

The NMMLA Framework makes a significant contribution to the field of m-learning, especially within the African context where internet connectivity and bandwidth continue to pose a great challenge. As a growing field, proponent and scholars such as Traxler [1], Kurubacak [37], Pettit and Kukulska-Hulme [38], Motiwalla [39], Sharples et al. [40], Keegan [41] and others, as cited by Muyinda et al. [17], have called for the development of theories, models, frameworks and tools that can advance the m-learning field. Thus, this research attempts to respond and act in line to this clarion call by making the following contributions:

1. Providing a framework for the design and development of native mobile multimedia learning applications, which support the delivery of rich interactive and modular multimedia learning content such as texts, images, audio, video and animations as well as multiple and customizable courses and themes.
2. Showing how to design and implement a framework as a library on Android platform using software engineering development techniques, Eclipse IDE and relevant tools.
3. Providing a broader understanding of the area of multimedia learning application development (mobile and desktop alike) and a basis for future improved frameworks and systems that integrate native and web-based technologies and platforms.

Chapter 2
Literature Review

Computing power and network bandwidth have increased dramatically over the past decade. However, the design and implementation of complex software remains expensive and error-prone. Much of the cost and effort stems from the continuous re-discovery and re-invention of core concepts and components across the software industry. In particular, the growing heterogeneity of hardware architectures and diversity of operating system and communication platforms makes it hard to build correct, portable, efficient, and inexpensive applications from scratch.

– Fayad & Schmidt (1997)

2.1 Overview of Software Framework

In this chapter, we present the review of existing relevant literature, which we carried out in order to know what has been done before on the subject matter of this thesis, what standards and guidelines that are already existing, which we could leverage in the design of our proposed framework. We started by defining and establishing what a framework is, especially in the field of software engineering. We also looked at their advantages and disadvantages. We then proceeded to review a cross-section of open-source and commercial mobile frameworks, multimedia learning frameworks and learning management systems (LMSs) in general.

Framework means different things in different fields. However, generally, it is provided as a guide for project implementers or system developers to follow in order to avoid duplication of efforts or reinvention of the wheel, which Fayad and Schmidt [42] rightly pointed out above as being one of the major costs in software development. Generally, a framework can be defined as a blueprint—a guideline or set of steps—that guides the execution of a project, carrying out of a process or the realization of a system. However, in computer programming, a framework, technically known as software framework, is an abstraction in which software providing generic functionality can be selectively altered or manipulated by additional user written code, resulting in the realization of application-specific software [28]. According to [43], a software framework is a set of source codes or libraries which provide functionality common to a whole class of applications. In other words, a software framework can defined as a set of cooperating and customizable classes, which offers a reusable design for a specific class of software [44].

Furthermore, it can be established from the foregoing that a software framework is a universal, reusable software platform used to develop or instantiate software applications, products and services in a given domain. It is leveraged by programmers to facilitate software development by creating application-specific subclasses of the framework's abstract classes. For the most part, the framework dictates the application architecture (overall structure, partitioning into classes and objects, how the classes and objects collaborate etc) and the overall application's flow of

execution. These design parameters are predefined so that the application designer can concentrate on the specifics of the application, as the design decisions that are common to its application domain are already taken care of by the framework. For this reason, frameworks more often than not emphasize design reuse over code reuse. However, in some cases, frameworks include concrete subclasses, which developer can put to work immediately. (This is where the NMMLA framework fits in.)

Examples of frameworks include support programs, compilers, code libraries, application programming interfaces (APIs) and tool sets (toolkits) that integrate all the different components to enable development of a project or solution [28]. Specific examples include web application framework which provides user session management, data storage, and a templating system; desktop application framework which provides user interface functionality and widgets, which are commonly used GUI elements [43]; and the Android Framework utilized in developing the NMMLA framework.

2.1.1 Features of a Framework

There are some key characteristics which differentiate a framework from normal libraries. They are discussed in the following section [28].

1. *Inversion of Control:* In a framework, unlike in libraries or normal user applications, the overall program's flow of control is not dictated by the caller, but by the framework. The "inversion of control" in the run-time architecture of a framework is often referred to as *The Hollywood Principle*, *i.e.*, "Don't call us, we'll call you." This is characteristic, for example, of the Android application framework which is built extensively around callbacks.
2. *Default Behavior:* A framework has a default behavior. This default behavior must actually be some useful behavior and not a series of no-ops.
3. *Extensibility:* A framework can be extended by the user usually by selective overriding or specialized by user code to provide specific functionality. A good example is the t
4. *Non-modifiable framework Code:* The framework code, in general, is not allowed to be modified. However, users can extend the framework, but not modify its code.
5. *Multi-functional:* A framework offers a broader range of functionalities, which are all often used by one type of application while a library usually provide one specific piece of functionality.

2.1.2 Purpose for a Framework

Generally, the purpose of software frameworks is to facilitate software development by allowing designers and developers to concentrate on meeting software requirements rather than dealing with the more standard low-level details of providing a working system, thereby reducing the possibility of introducing new bugs and the overall development and deployment time.

2.1.2.1 Advantages of Using a Framework

Leveraging frameworks in the development of applications offer a number of advantages. The advantages include the following [43]:

1. *Reliability and Cost-Effectiveness:* Using code built and tested by other programmers increases software reliability, and saves time and money.
2. *Specialization:* It allows for specialization in collaborative projects, which are made of framework and application developers. This separation of tasks, which uses divide-and-conquer mechanism, allows each team to focus on more specific goals and leverage their individual strengths. For example, the programmers who are experts at user interface design might work on the client application while the security experts test and strengthen the framework upon which the application is built.
3. *Security:* Frameworks can offer security features which are often required for a common class of applications. This provides every application written with the framework to benefit from the added security without the extra time and cost of developing it. Examples include secure session management and escaping database input.
4. *Modularity:* By handling "lower level" tasks, frameworks can assist with code modularity. Business logic, for example, can remain in the application while the mundane tasks of database connectivity and handling user logins can be handled separately in the framework.
5. *Best Practices:* Frameworks often help enforce platform-specific best practices and rules. A desktop GUI framework, for example, may automatically build toolbars and buttons common to the local operating system. A web application framework may assist with encrypting user passwords or payment processing.
6. *Design Patterns Support:* Frameworks can assist in programming to design patterns and general best practices. For example, many frameworks are built according to the Model-View-Controller (MVC) design pattern [45].
7. *Upgrade Benefits:* Upgrades to a framework can enhance application functionality without extra programming by the final application developer. If, for example, an e-commerce framework offers a new payment method, that option can automatically become available to the end user with no extra programming by the application developer.

2.1.2.2 Disadvantages of Using a Framework

There are disadvantages as well in using a framework. They include but not limited to the following [43]:

1. *Degraded Performance:* Performance can sometimes degrade when common code or classes are used. This sometimes occurs when a framework must check for the various scenarios in which it is used to determine a path of action. It can also occur with generalized code that is not optimized for a specific situation. Performance degradation,

though, is often offset by the enhanced speed of development and quality of the final application.

2. *Steep Learning Curve:* Frameworks often require a high and very often steep learning curve so as to use them efficiently and correctly. Usually, specific frameworks become more valuable to individual programmers as they use them repeatedly. As developers continuously use the same framework for each new project, the learning curve flattens and productivity increases.
3. *Inflexibility:* Functionality which needs to circumvent or work around deficiencies in a framework can cause more programming issues than developing the full functionality in the first place. Good frameworks provide utility and structure while still leaving enough flexibility to not get in the way of the programmer. Some frameworks are so rigid and highly structured that choosing them for an inappropriate project can be disastrous. This is not the fault of the framework, but some are more generally suited and flexible than others. This must be carefully considered by those choosing a framework.
4. *Bugs and Insecurity:* Bugs and security issues in a framework can affect every application, which utilizes or instantiates that framework. For this reason, a framework must be tested, proven and patched separately or in addition to the final software product.

2.2 Mobile Application Frameworks

A cross-section of existing mobile application frameworks will be reviewed in the following subsections. We will focus on the major and most popular existing learning environments, which we could find. In addition, we will endeavour to highlight their strengths and application areas. Finally, we will look at the reason why and situations where the NMMLA framework will be of greater benefit to content developers.

2.2.1 Android Application Framework

Android is a framework developed and promoted by a consortium of hardware and software companies, ranging from device and chip manufactures to telecommunication and software providers. It was initiated by Google in 2007 under the umbrella of OHA. Its ultimate goal was to promote ubiquity and open source development. The Android framework basically is a collection of APIs that allow developers to quickly and easily build Android applications for phones and tablets alike. The framework, as shown in the Android platform architecture in Fig. 2.1, sits between the system/developers' applications and the Android Runtime which include the Core Libraries and the Dalvik Virtual Machine (VM). The Core Libraries exposes most of the functionalities in the Java programming language. The Dalvik VM is written in such a way that Android devices are able to run multiple VMs efficiently. It executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. Also, it relies on the Linux kernel for underlying functionality such as threading and low-level memory management. Every application runs in its own process, with its own instance of the VM. This prevents each running application from interfering with others. Underlying the Android Runtime are the native libraries (C/C++) which are sitting directly on the Linux OS (a light-weight and secure kernel).

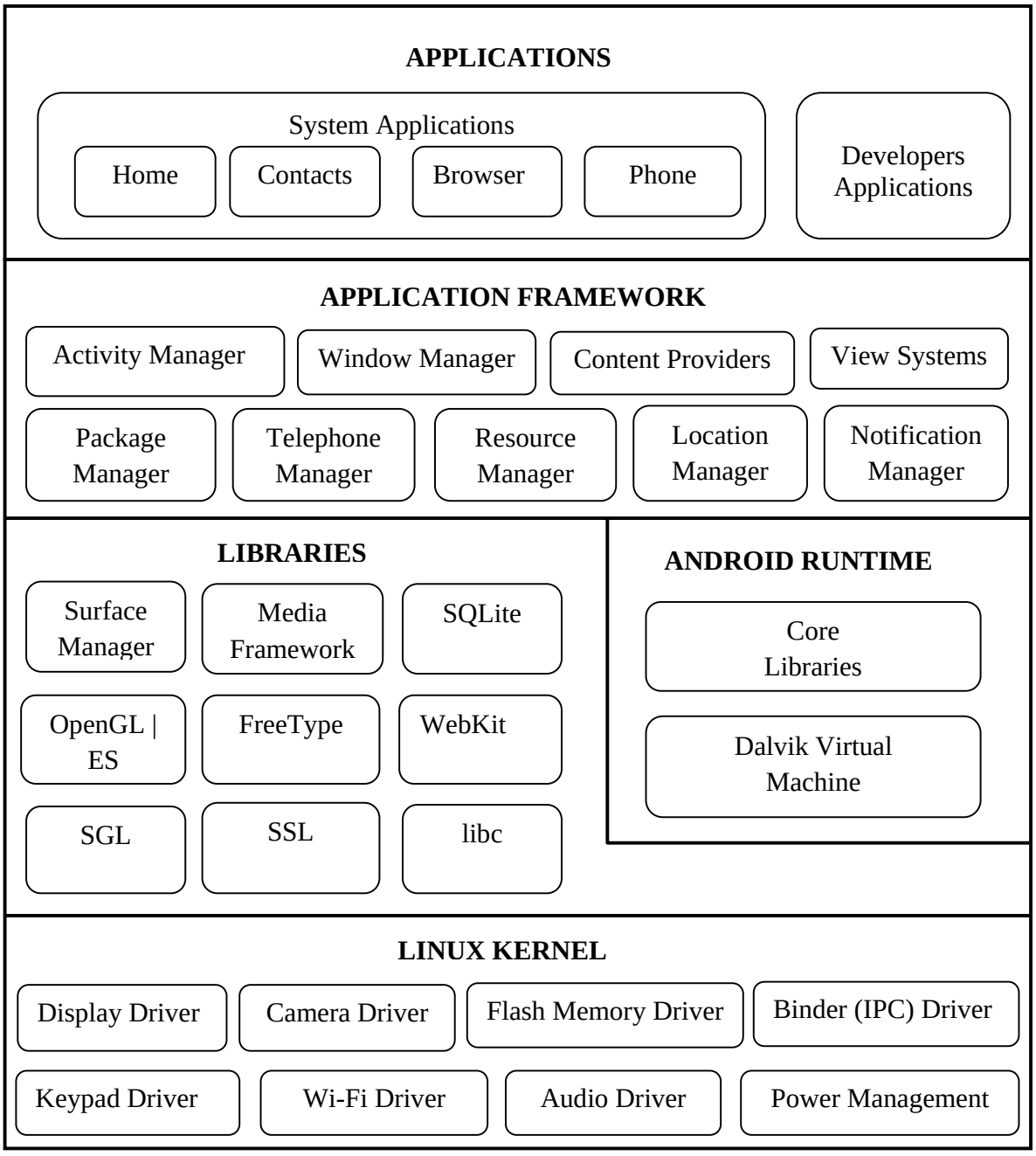


Figure 2.1. Android Platform Architecture

These libraries make it possible for native and third-party applications to communicate with the underlying OS. The set of native C/C++ libraries (see Table 2.1), included in the Native Development Kit (NDK), is utilized by various components of the Android system. Through the Android Framework, these capabilities are made accessible to application developers [6].

Table 2.1. Android Native C/C++ Libraries

S/N	Library	Function
1.	System C Library	This a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices.
2.	Media Libraries	It is based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG.
3.	Surface Manager	It manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
4.	LibWebCore	It is a modern web browser engine which powers both the Android browser and an embeddable web view.
5.	SGL	The SGL is the underlying 2D graphics engine.
6.	3D Libraries	This is an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer.
7.	FreeType	FreeType is used to render text images (of vector and bitmap font formats) on to the screen.
8.	SQLite	It is a lightweight relational database engine, leveraged by Android applications.
9.	Linux Kernel	The Linux kernel acts as an abstraction layer between the hardware and the rest of the software stack. Android leverages Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model.

Though the Android Framework is open source such that just anyone could develop a native application in whatever domain, developing a functional application on this platform may require some amount of time and basic knowledge of software development. For the most part, the learning curve could be very steep for most developers, especially for beginners or new comers on the Android platform. One of the ultimate goals of this thesis is to reduce this challenge in the m-learning domain by providing: 1) a blueprint for application/framework developers; and 2) a one-page-setup and Do-It-Yourself (DIY) toolkit for content developers, which will facilitate the building of complete functional native mobile multimedia learning application on the Android platform.

2.2.2 Rhodes Framework

Rhodes Framework, now called RhoMobile Suite, is an open-source MIT-licensed Ruby-based platform, which allows developers to quickly write locally executing, device-optimized native mobile applications. Developed by Motorola, it is the first smartphone framework for mobile application development. It supports mobile platforms such as Android, iPhone, iPad, RIM Blackberry, Windows Mobile, Windows Phone 7 and Symbian. It also supports the use of synchronized data offline. It is based on MVC, with views written in HTML and controllers in Ruby. It is able to leverage device capabilities such as GPS, PIM contacts and calendar, camera, native mapping, push, barcode, signature capture, Bluetooth and Near Field Communications

(NFC). One of its advantages is that it allows for writing codes or interfaces (in HTML) once and building for the majority of smartphones [46]. However, just like the Android Framework, the learning curve could be very steep. It is not completely free as the word "open-source" suggests, as you are required to pay to have access to higher features as an enterprise. Besides, it is not tailored specifically for multimedia learning application development like the NMMLA framework.

2.2.3 Open Mobile IS

Open Mobile IS is an open source framework for mobile application development. It is licensed under GNU LGPL. Its goal is to provide all the necessary tools, APIs and documents that will ease the development of effective nomad applications. It utilizes the Java framework, which is divided into components providing all the needed functionalities [47].

2.2.4 PhoneGap

PhoneGap is a free and open-source framework that allows developers to easily create mobile applications using standardized web APIs for a wide range of platforms which include Android, Symbian, Blackberry, iPhone, Bada, Windows Phone 7 + 8 and WebOS. The Web technologies include HTML, CSS, and JavaScript. However, to easily communicate with backend services written in any other languages, developers can make use of network protocols such as XmlHTTPRequest, Web Sockets, etc. This capability allows PhoneGap apps to remotely access existing business processes while the device is connected to the Internet. Also, there are different numbers of APIs which are available for different platforms. For Android in particular, APIs are available for File, Media, Storage, Contacts, Camera, Compass, GPS etc. Fig. 2.2 shows the process of creating a PhoneGap application and deploying it to various mobile platforms. Developers can build their own apps, called third-party apps, and deploy them to the marketplace such as App Store and Google Play for end-users to download [48].

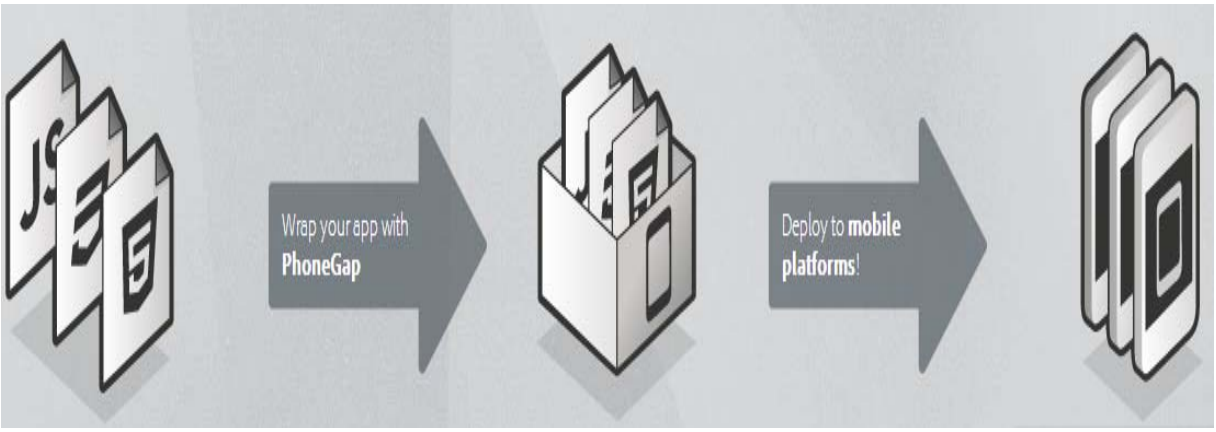


Figure 2.2. Schematic for Creating PhoneGap Mobile App

2.3 Multimedia Learning Frameworks and Environments

A significant amount of work has been done relating to multimedia learning environments and frameworks. However, most of them are theoretical, web-based and on multimedia content. Parsons et al. [49] worked on Design Requirements Framework for Mobile Learning Environments in which they proposed a conceptual framework for mobile learning applications that is able to provide systematic support for mobile learning experience design. They combined a game metaphor and several studies (narratives) of mobile learning environments to realize it. The m-learning design requirements include Learning Objectives, Learning Experience, m-Learning Context and Generic Mobile Environments. To test the framework, they took both reverse and forward engineering approaches. They applied it as "a post-hoc analytical tool to three successful m-learning systems and as an ad-hoc analysis framework for the requirements phase of an m-learning project for the purpose of validation." As a post-hoc analytical tool, they mapped the characteristics of the systems onto the framework. However, this framework is yet to be applied within a design and implementation context of a complete m-learning application.

Shanmugapriya and Tamilarasi [4] worked on Android-based learning environment using web services. In this work, they proposed an m-learning application for ubiquitous learning environment. They showed a framework where Android phones and tablets are able to access the web-based m-learning system from the classroom environment using Wi-Fi and outside the campus using 3G Network, and a laptop or computer from the home environment using wireless Internet. They also outlined the roles of the various users of the system such as content developer, application (framework) developer and learner. However, they did not show how such a mobile application framework can be developed by the framework developer, for example, how the application data can be modelled.

Mostakhdemin-Hosseini and Tuimala [50] also did some work on a mobile learning framework. In their work, they gave an overview of the development issues and steps the mobile learning system developer must consider and follow respectively. Concerning the former, they identified analysed three factors which include Mobile Usability, Wireless Technology and E-learning System. Regarding the latter, they captured the steps in a block diagram, which include Education Components, Device/Network Capabilities, Concept Development and Prototyping. They went further to show the Prototyping in a Use Case Diagram, where the major functionalities of the framework were captured. They include Register a Course, Download Exercises, Listen via Mobile Phone, Access Course Website, Follow Lecture Session, Publish Task, Provide Feedback, Receive Task Answers and Upload Lecture Session. After the assessment of the mobile learning systems built based on the framework, they came up with the following conclusion:

1. Designing m-learning system requires that the implementation platform be carefully studied.
2. The main challenges in the development of the prototype were network capabilities and the deficiency of deploying self-developed application on mobile devices.

3. Video streaming plays an important role in a mobile learning system but available limited bandwidth posed a great challenge for PDAs and mobile devices at the time. However, 3G network has the potential of offering real time, streaming access to video archives.
4. Students and staff are willing to embrace m-learning system in their education process.

From the foregoing, we find that the mobile learning framework presented by Mostakhdemin-Hosseini and Tuimala [50] is web-based and highly abstracted, while the NMMLA framework which we are proposing in this thesis is native and goes down to show, for example, how the modular contents are designed and organized, and how simulations can be integrated into the mobile learning environment, as portrayed in the application data model class diagram.

Muyinda et al. [17] carried out research on how pedagogical models of reusable learning objects (RLOs) could be designed and developed for mobile learning in different contexts, especially in developing countries. They used a "Design Research approach to develop a UML-based model for instantiating applications for deploying and utilizing learning objects on multi-generation order mobile phones in developing countries of Africa." They called it M-learning Object Deployment and Utilization Model (MoLODUM). However, this framework was web-based, whereas in this thesis we propose a framework for native mobile applications.

Furthermore, Leacock et al. [51] worked on a framework for evaluating the quality of multimedia learning resources. In the work, they presented the structure and theoretical foundations of a Learning Object Review Instrument (LORI), which ultimate goal was to "balance assessment validity with efficiency of the evaluation process". The instrument enables learning object users to create reviews based on ratings and comments on nine metrics of quality: content quality, learning goal alignment, feedback and adaptation, motivation, presentation design, interaction usability, accessibility, reusability, and standards compliance. They concluded as follows:

1. With a few minutes of effort, an evaluator can provide a meaningful learning object review that will be informative on its own and can also be aggregated with the reviews of others who have evaluated the same object. Therefore, LORI strikes a pragmatic balance between depth of assessment and time.
2. LORI, as empirically proven by a number of researchers, is useful within the context of collaborative evaluation and, when used in an educational environment, will go a long way in helping participants to acquire instructional design and development skills.
3. A heuristic approach, which LORI is based on, is better than the traditional approach (e.g. pre-publication expert peer review), which is often labor- and time-intensive.

Diezmann et al. [52] presented a theoretical framework for multimedia resources, which was used to guide the development of a set of multimedia resources in science education. It was designed to support contemporary approaches to learning and teaching where learners are viewed as active constructors of knowledge and teachers as facilitators of that learning process

[53]. They outlined a number of features expected of such multimedia and their associated design principles (see Table 2.2), which multimedia learning content and framework developers must take into consideration. They concluded that educational multimedia resources have much to offer teacher education programs, especially in the light of increasing enrolments, diminishing budgets and flexible delivery.

Table 2.2. Features of Multimedia and Associated Design Principles

S/N	Features	Principles
1.	Screen Design	• Focus the learner's attention • Develop and maintain interest • Promote processing • Promote engagement between the learner and lesson content • Help learners find and organize information • Facilitate lesson navigation
2.	Interaction	• Provide opportunities for interaction • Chunk the content and build in questions and summaries • Ask questions but avoid interrupting the instructional flow • Use rhetorical questions to get students' to think about content and to stimulate curiosity • Provide for active exploration in the program rather than a linear Sequence
3.	Feedback	• Keep feedback on the same screen as the response • Provide feedback immediately following a response • Provide feedback to verify correctness • Tailor feedback to the individual • Provide encouraging feedback • Allow students' to print feedback
4.	Navigation	• Clearly defined procedures for navigation and support • Consistency in screen structure and location of keys • Use of familiar icons on control panels • Progress map or chart to show location within a program • Help segments with additional information to allow a learner to follow interests and construct his or her own learning experiences
5.	Learner Control	• Provide selectable areas for users to access information • Allow users to access information in a user-determined order • Provide maps so students can find their locations and allow students to jump to locations • Provide feedback if there are to be time delays on accessing information • Arrange information so users are not overwhelmed by the quantity of information • Provide visual effects and give visual feedback
6.		•Use colour sparingly and consistently with a maximum of 3 to 6 colours per screen • Use brightest colours for most important information

		• Use neutral colours for backgrounds and dark colours on a light background for text • Avoid combining complementary colours (e.g. red/green) • Use commonly accepted colours for particular actions (red for stop) • Avoid hot colours on the screen as they appear to pulsate
7.	Graphics	• Graphics include photos and scanned pictures • Icons and photos enhance menu screens • Information is better understood and retained when supplemented with graphics • Avoid graphics for decoration or for effect • Use graphics to indicate choices (e.g. left/right arrows)
8.	Animation	• Can be motivational and attention getting • Useful for the explanation of dynamic processes • Subtle benefits by highlighting key information, heightening interesting, and facilitating recall
9.	Audio elements	• Use audio when the message is short and audio rather than text for long passages • Do not let audio compete with text or video presentation • Provide headphones for in-class use • Tell students what is relevant and chunk the message with other instructional activities
10.	Video elements	• Use video as an advance organizer or a summation • Synchronize video with content, and reinforce/ repeat the concepts being presented

2.4 Learning Management Systems

In this section, we review two widely used mobile learning environments. They include Moodle and Blackboard, which are open source and commercial respectively. Both are LMSs, which are leveraged by HEIs, organizations and individuals to meet their educational and learning needs.

2.4.1 Moodle

Modular Object-Oriented Dynamic Learning Environment (Moodle) is an open-source software package under GPL for producing Internet-based courses and websites. It is a global development project designed to support a social constructionist framework of education. It can be installed and run on any Windows, Mac and Linux OS computer that can run PHP and support an SQL type database such as MySQL. Basically, a Moodle web-based application comprises a front page through which users can log in using their browser. Usually, the way users access a Moodle site varies. Depending on the organization, they might be given logins; they might be permitted to create accounts themselves, or they might be signed in automatically from another system. The following outlines the basic features of Moodle [54]:

1. Moodle's basic structure is organized around courses. These are basically pages or areas within Moodle where teachers can present their learning resources and activities to students.

2. Courses can contain content for a year's studies, a single session or any other variants depending on the teacher or establishment. They can be used by one teacher or shared by a group of teachers.
3. How students enroll on courses depends on the establishment; for example they can self-enroll, be enrolled manually by their teacher or automatically by the admin.
4. Courses are organized into categories according to fields or disciplines. For example, Physics, Chemistry and Biology courses might come under the Science category.
5. Users log into Moodle without no special privileges—"teacher" or "student" role. However, according to their needs, individual courses and context, they are assigned roles by the administrator.

In summary, the major educational features and services offered by Moodle include assignment submission, discussion forum, files download, grading, instant messages, online calendar, online news and announcement, online quiz and wiki.

However, the Moodle project has evolved to include Moodle Mobile on different mobile platforms such as Android, iOS, Blackberry etc. Applications available on Android platform include UMM, mTouch, Mobile Learning Environment (MLE), mPage, mBook, mBot, Moodle Joule etc. UMM, for instance, is an unofficial clone of the Moodle Mobile app for iPhone that works on Android and Blackberry devices. It was built using the high-level cross-platform JavaScript framework PhoneGap. As a result, it uses web technologies such as HTML5, CSS3 and JavaScript. UMM is intended and has been designed to be easily customizable by HEIs, organizations, educational website owners etc [55].

2.4.2 Blackboard Mobile Learn

Blackboard Mobile Learn is a commercial web-based mobile learning environment that allows subscribers to learn anywhere and anytime. It is basically a LMS, which provides teachers and students, for example, with access to their courses, assignments grades, announcements, online discussion forums etc through their mobile devices while they are on the move. It also provides them with the freedom to organize their content the way they choose to on a wide range of mobile devices which include iOS, webOS, Android and Blackberry [56].

2.5 NMMLA Framework

From the foregoing review, we found out that the majority of existing work on mobile (multimedia) learning systems and frameworks had been centred round the web [7]. However, we deviate from this trend to present a NMMLA framework, which supports not just conventional multimedia learning content such as HTML, images, audio, video but simulation as well. Content providers with no in-depth programming or application development knowledge and without bothering about understanding and knowing how to use the Android SDK and Framework, can leverage our framework with little or no efforts to realize complete functional NMMLAs on the Android platform.

Chapter 3
Research Methodology

This research adopted two basic approaches in gathering requirements data and arriving at the final design for the framework. First, as part of a broader collaborative research project comprising two teams—framework developer and content developers—and given its responsibility to provide the required framework for instantiating a mobile multimedia learning application, comprising four (4) computer science courses, namely, Automata, Compiler, Data Structures and Algorithm, and Information Theory, which were being developed by four other M.Sc. students at the time, whose task was to develop rich multimedia learning contents, including simulations and animations, for the framework, which will serve as a pilot test, a series of interviews, discussions and interactive sessions were conducted with these researchers, various professors and other fellow researchers as well to capture the key requirements and specifications for the framework. Second, a survey of existing literature on the subject was carried out through desktop/internet research in order to put the requirements information gathered from the interviews and discussions in the right context, refine and reword them before moving on to design and implementation phases.

3.1 Framework Requirements

Based on the foregoing, the following requirements, subject to extension in the future, were arrived at (see Use Case Diagram in Chapter 1):

1. The framework should support various types of multimedia such as HTML, images, audio, video and simulations (graphics and animations).
2. The framework should support a `Course` menu.
3. The framework should support a `Theme` menu with default and/or user-defined themes.
4. The framework should support a dual view of modules and items, namely, `listview` and `tabview`, so as to provide the learner with a choice to decide how he wants his modular components and item-composed modules to be rendered on the screen.
5. The framework should support an `About` HTML file, which gives a summary of what an instance application utilizing the framework is all about and does.
6. The framework should support a `Help` file, which is a modular component composed of HTML or audio or video files, pinned to the actionbar, and the learner can consult when he needs usage help on certain components or parts of the application.
7. The framework should support a `Search` tool to look up words in an instance application's dictionary provided by the content provider.
8. The framework should support various types of components, namely, `AtomicItem`, `TabFile`, `ListItem`, `ListModule` and `TabModule`, targeted at delivering different multimedia contents at different levels in various display modes or views.

9. The framework should support pre-and-post self-evaluation sessions. Thus, it should support an `Evaluation` component on the homepage and a quiz action item on the actionbar when the learner is on a listview of modules/items or on a detail view of a file, which will enable him to pre- and post-evaluate himself respectively.
10. The framework should support the delivery of the application modules and quizzes from both SQLite database and the IDE's filesystem.
11. The framework should provide a significant number of icon sets, scaled for devices with ldpi, mdpi and xdpi screen densities, which the content provider can leverage as resources for application components, modules, items, such as `About`, `Help` etc.
12. The framework should be able to integrate seamlessly with other user-defined classes, for example, dynamic classes which deal with simulation.

3.2 Spiral Modelling Approach

The design and implementation of the framework adopted the Boehm Spiral Modelling Approach [57], with the final functional framework arrived at progressively and incrementally. Throughout the development, a continuous stream of versions were churned out and made available to the content developers, supervising professor and fellow researchers and students for their reviews and inputs. Fig. 3.1 shows an abstracted version of the Spiral Model [58], which combines the concept of iterative development (prototyping) with the systematic, controlled aspects of the Waterfall Model [28].

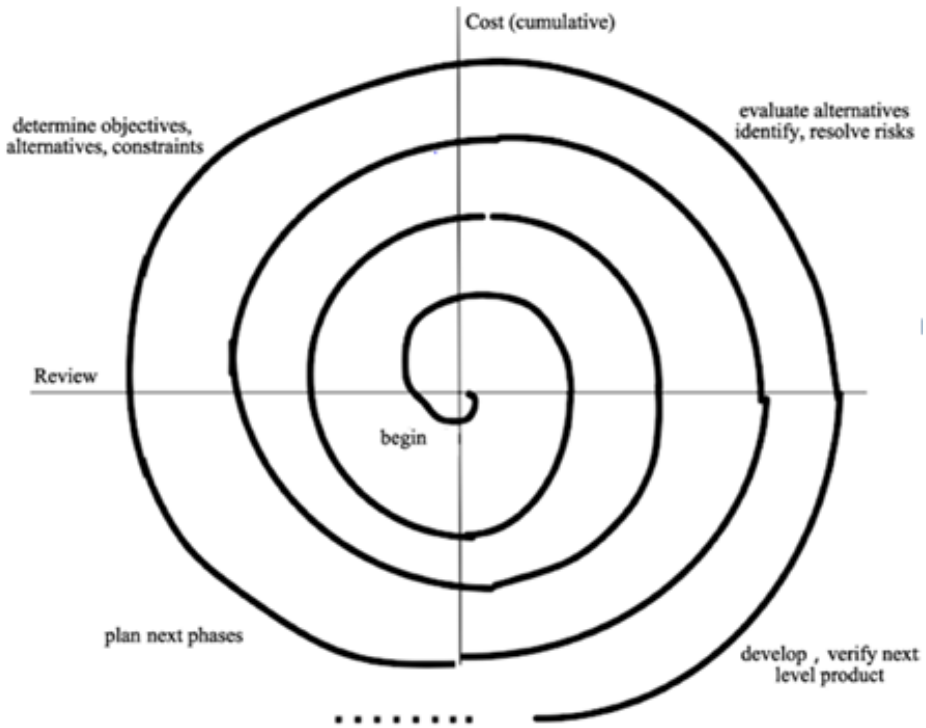


Figure 3.1. Abstraction of Key Underlying Ideas behind Spiral Model

The model is such that incorporates risk analysis and management all the way, which grows as the mass of the software development project, cost and thus understanding of the problem grow with each iteration. Starting at the centre, with the final design, implementation, integration, and test occurring in the final iteration, each turn around the spiral goes through a series of tasks, which are outlined as follows:

1. Determine the objectives, alternatives, and constraints on the new iteration.
2. Evaluate alternatives and identify and resolve risk issues.
3. Develop and verify the product for this iteration.
4. Plan the next iteration.

The reason for choosing the Spiral Model over other methods such as Waterfall is that the software development process can be repeated multiple times for multiple builds [28]. Also, some level of functionality can be delivered to the customer faster than the Waterfall method throughout the development cycle. Moreover, the Spiral Method helps in managing risk and uncertainty by giving room for multiple decision points and explicitly acknowledging that everything cannot be known before the next activity starts [59].

3.3 UML Diagrams

In designing and implementation the framework, Objected Oriented Programming (OOP) and Universal Modelling Language (UML) approaches were used. In particular, the following UML diagrams were leveraged in building the framework:

1. Use Case Diagram (UCD)
2. Technical Class Diagram (TCD)
3. Activity Diagram (AD)

3.3.1 Use Case Diagram

The UCD is a UML diagram which captures the main functionalities and actors together with their relationships in the instantiation of the framework. The design of the framework started with a Use Case Diagram (see Fig. 1.4 in Chapter 1) which gives a functional overview of the framework based on the requirement analysis. It depicts the main and supporting components, actors, data storage systems and how they interact with one another to provide a complete functional multimedia learning application.

3.3.2 Technical Class Diagrams

Generally, a Technical Class Diagram (TCD) gives a static overview of all the classes in an application and how they relate with one another. Fig. 1.6 shows the Framework Data Models (FDMs) together with the Homepage class. Basically, the TCD depicts the framework architecture, i.e. its classes and their static relationships such as aggregation, inheritance etc. In addition, the framework's TCDs indicate the communication of one class to another by using references (diamond-ended arrows). Appendix A shows other (detached) class diagrams, which make up the framework. (See Table 4.3 for most FDM classes and their constructors.)

3.3.3 Activity Diagram

An Activity Diagram is a UML diagram, similar to a flowchart, which captures the business processes in a system at a high level and the details of complex operations at a low level. As part of the design and implementation of the framework, an Activity Diagram was drawn to depict how data (content) and control flow between the major `activities` when an application is instantiated from the framework. See Fig. 4.2 and Section 4.4 in Chapter 4 for the Activity Diagram and detailed explanation respectively.

3.4 Modelling View Controller

The framework also utilized the MVC design pattern in realizing a functional application. The MVC concept was first defined by the programming language Smalltalk in the 1970's. Basically, an application can be thought of as having three main layers as follows:

1. Presentation (user interface)
2. Application logic (business process)
3. Resource management

From the MVC standpoint, the presentation layer is divided into controller and view. The most important separation is between presentation and application logic. The view/controller split is less so. MVC is concerned with more of the architecture of an application than is characteristic of a design pattern. Hence, it is sometimes regarded as an architectural pattern. This architecture, depicted in Fig. 3.2, is a widely adopted pattern across many languages and implementation frameworks, especially in web-based applications. Android is an example of a framework, which implemented MVC. The main purpose of MVC is to achieve code reusability and a clear separation between its three components, which include the following [45]:

1. Model: business logic and processing
2. View: user interface (UI)
3. Controller: handles navigation and input commands

3.4.1 Model

Model is the domain-specific representation of the data on which the application operates. The model, sometimes referred to as the domain layer, is another name for the application or business logic layer, which is responsible for manipulating and adding meaning to raw data. The application model includes the following:

1. Application state
2. Application rules
3. Persistence data

As shown in Fig. 3.2, the model is responsible for encapsulating the application state and exposing its functionality. In addition, the model responds to state queries made by the view and notifies the view of its change in state. However, though Fig. 3.2 shows an interaction between the model and view, in some implementations, there is none, as the controller sits in-between.

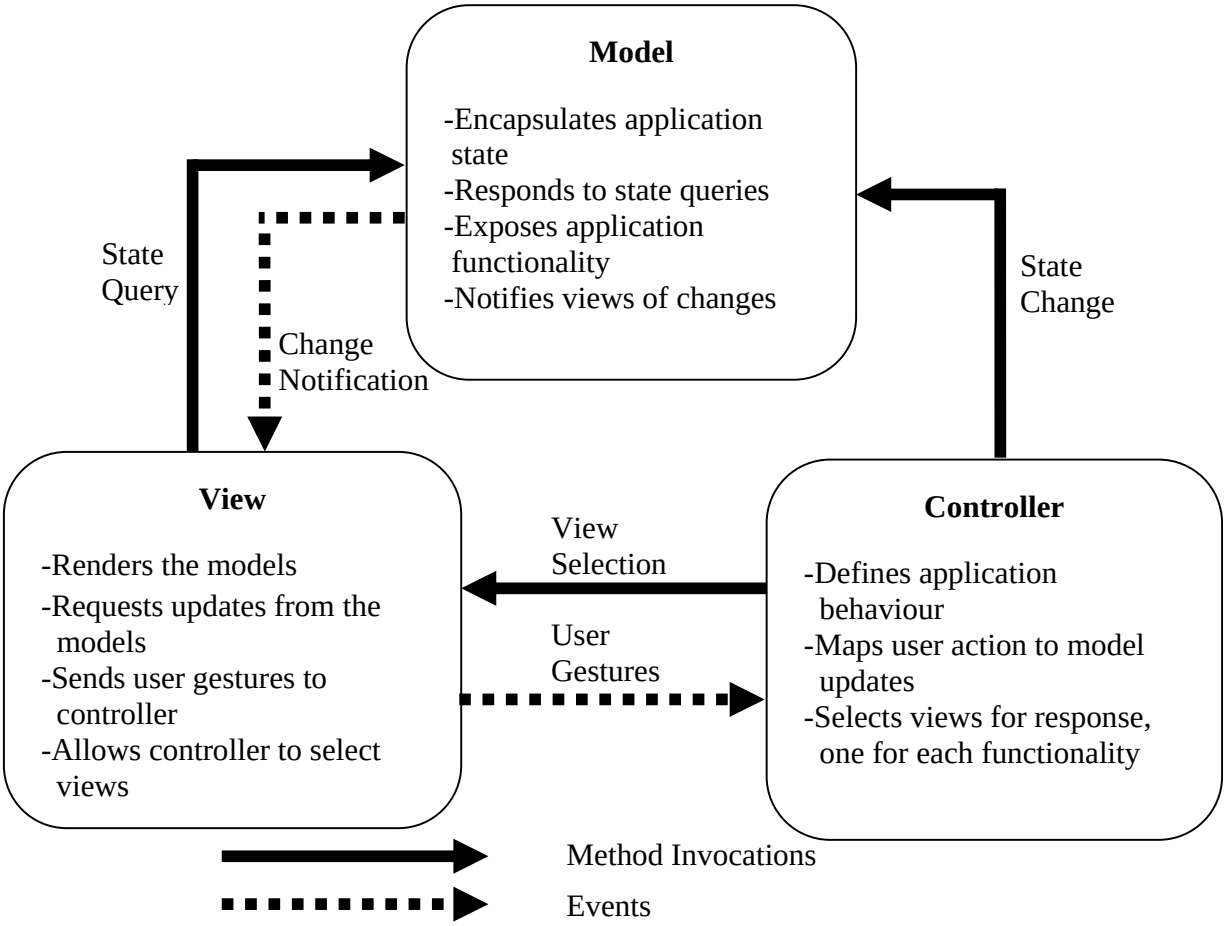


Figure 3.2. Model View Controller Architecture

The instances of the application data, defined by the data model, currently in play, and the value of those instances or objects, represent the application state. The application rules represent the business logic of the application, for example, "Who does what?" in an admin-user software system, usually pre-defined by privilege levels by the admin. Persistence refers to long-term storage of data beyond the scope and time-frame of the user's session in the application. Though many applications use a persistent storage mechanism (such as a database) to store data, MVC does not specifically mention the resource management layer because it is understood to be underneath or encapsulated by the model. The model also has the ability to read/write from different or multiple databases [45].

3.4.1.1 Framework Data Model

A model, in our framework, is a program class that is used to retrieve data from a database or other data sources such as the development environment's filesystem. Generally, the model has attributes and methods. The latter are used to alter or manipulate the former (e.g. RM of a module) by the controller before rendering them in a view. In the NMMLA framework, an SQLite database and the IDE filesystem are used to store the application's core data such as learning modules made up of file items and quizzes made up of questions, options and answers.

In the database, these data are basically tables of texts and numbers, which are to be created and provided by the content developers, while in the filesystem, they are files such as HTML, images, audio, video, and classes such as simulations. See FDM in Fig. 1.6, which depicts the framework data models, which include `File`, `Module`, `Component`, `Question`, `Quiz` etc.

3.4.1.2 Module and Quiz Data Models

Table 3.1a shows a module's database table (modelled as `Module` in the FDM). It outlines the fields and data types contained in the module table. Similarly, Table 3.1b shows the quiz table (modelled as `Quiz` in the FDM) together with the constituent fields and data types.

Table 3.1a. Module Table Fields and Data Types

S/N	Field	Data Type
1.	_id	NUMBER
2.	fileName	TEXT
3.	fileEntryName	TEXT

Table 3.1b. Quiz Table Fields and Data Types

S/N	Field	Data Type
1.	_id	NUMBER
2.	Question	TEXT
3.	correctAnswer	TEXT
4.	incorrectAnswer1	TEXT
5.	incorrectAnswer2	TEXT
6.	incorrectAnswer3	TEXT

The physical files are stored in the IDE's filesystem (see Table 3.2). They include HTML and image files, which are stored in the `asset` folder, audio and video files in the `raw` folder, and module and quiz database files in the `asset` folder.

Table 3.2. Native Application Files and Storage Folders

S/N	Files	Folder
1.	HTML	`Asset`
2.	Image	`Asset`
3.	Audio	`Raw`
4.	Video	`Raw`
5.	Module database	`Asset`
6.	Quiz database	`Asset`

The application databases, which comprise respective module and quiz tables are also stored in the `asset` folder of the filesystem as well. In the module table, the "fileEntryName" in the SQLite database is just a pointer to the files located in the `asset` folder. So, as shown in Fig. 3.3, while the names of the files are looked up in the database, the actual files are fetched from the appropriate folders in the IDE. However, as an alternative, the HTML and image files can be directly read from the filesystem without having to look them up in a database. See Section 5.2.

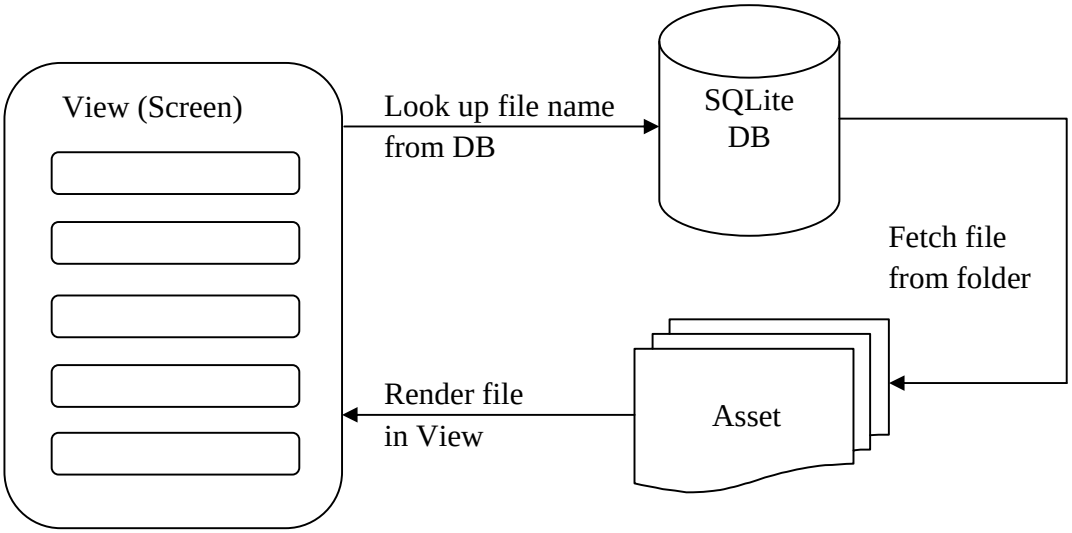


Figure 3.3. Data Flow from SQLite DB and Asset Folder to View

3.4.2 View

The view is what the learner sees. It is responsible for displaying the model on the screen using objects called widgets suitable for user interaction. In web applications, the view is the HTML page and the code which gathers dynamic data for the page, while in a native Android application, the view includes core data, business logic and navigational widgets (e.g. `listviews`, `textviews`, `imageviews`, `buttons` etc); and skin (general look of the screen). Fig. 3.4 shows the view's loading and population mechanism by the controller in a given application. A `listview` is loaded from an XML file stored in the layout folder into an `activity`. It can then be populated with data fetched from either within the application code or SQLite database or the application project's filesystem, or any combination.

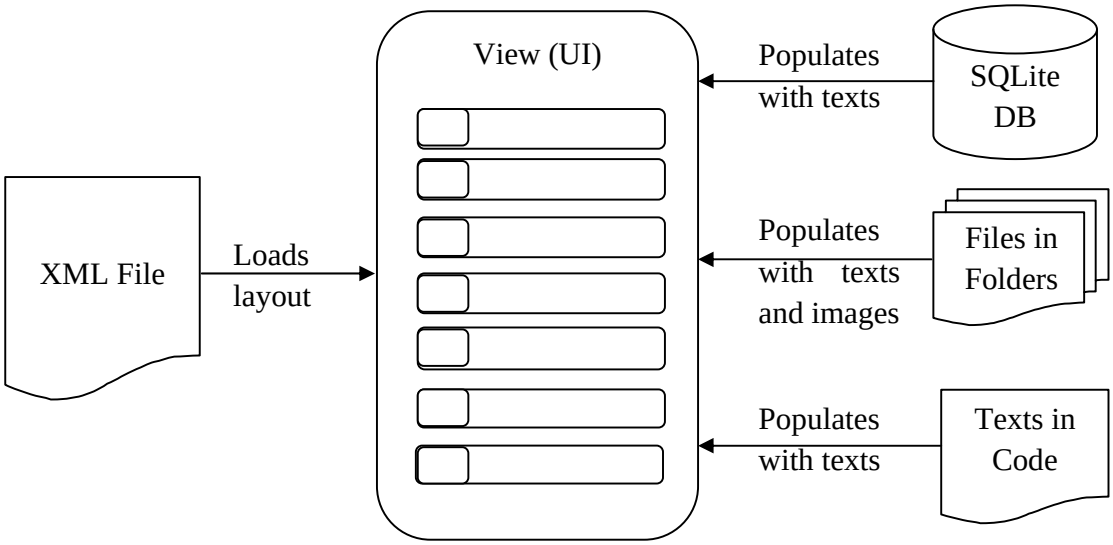


Fig. 3.4. Loading and Population of View Mechanism by Controller

3.4.3 Controller

The controller maps user requests to model actions, and the response of the Model to appropriate View. Within the context of our framework, Fig. 3.5 shows the data flow mechanism in an instance application presenting a list of items in a `FragmentActivity` (`FA`). The following describes the MVC communication as a result of learner's events:

1. Upon a previous event, the learner is presented with the current UI shown in Fig. 3.5.
2. The learner interacts with the UI in some way, e.g. by clicking an item, say a video item, in the `listview`.
3. The controller, in this case a `LMFA`, receives and responds to the input event from the learner via a registered handler known as a callback, which associated interface is implemented by `LMFA`, and sends an `intent` to a new controller (this time a regular `activity` not shown so as to keep it simple).
4. The new controller (represented by the same controller in the diagram) accesses the clicked `File` model, holding the filename from the SQLite database, and uses it to fetch the actual file from the `raw` folder into a `videoview` (not shown) on the screen.
5. The current controller, presenting a new UI (detail view of file), waits for further learner's interactions, to begin a new cycle of MVC interaction, depending on the event.

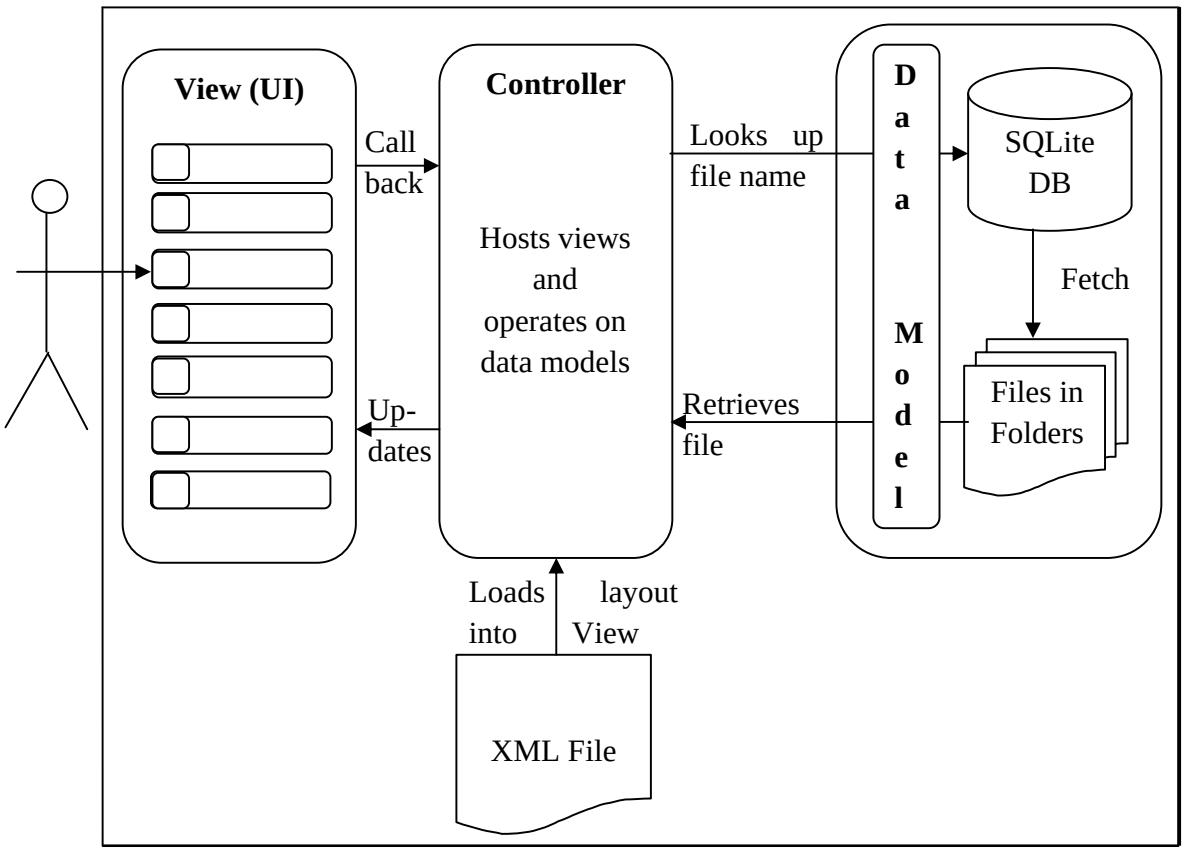


Figure 3.5. Implementation of MVC in NMMLA Framework

Chapter 4
Framework Implementation

After the design of the framework using UML and non-UML-based diagrams as seen in Chapter1 and Chapter 3, the next task was implementation (coding). This was carried out using the following Android development tools:

1. Java 1.6 (or higher) [6]
2. Eclipse Integrated Development Environment [61]
3. Android Development Tool (ADT 21.1.0) plugged into the Eclipse IDE [6]
4. Eclipse IDE Android Icon Set Wizard
5. Online Android Asset Studio [62]
6. Online Android Action Bar Style Generator [63]
7. SQLite Database Browser
8. ActionBarSherlock Library [11]
9. Android Phone Emulator (Android 3.2, API 13, QVGA 320X480 mdpi)
10. Android Tablet Emulator (Android 3.2, API 13, WSVGA 1024X600 mdpi)

Java 1.6 is last-but-one-version update to the Java platform from Sun Microsystems Inc, which comes with the Java Development Kit (JDK) and Java Runtime Environment (JRE) integrated into the Eclipse IDE, where the framework was developed. Eclipse IDE's Android Icon Set Wizard and Android Asset Studio are used to quickly generate and scale icon assets from existing source images, clipart, or text for different Android devices with different screen densities such as ldpi, mdpi, hdpi xdpi etc. Examples include actionbar and tab icons for Android 3.0 and above, tab and menu icons for pre-Android 3.0, launcher icons, notification icons, generic icons etc. Similarly, the Online Android Action Bar Style Generator allows developers to easily create a simple, attractive and seamless custom action bar style for Android applications [63]. Also, it is able to generate all the necessary nine patch assets plus associated XML drawables and styles which developers can copy directly into their project. For the framework, it is used to generate customized themes (`Theme_Green` and `Theme_Blue`) in addition to the pre-defined themes (`Theme.Sherlock`, `Theme_Sherlock_Light_DarkActionBar` and `Theme.Sherlock.Light`), which any framework `activity`, extending `SherlockActivity` must set its theme attribute to, either in the manifest or code as conditioned by the ActionBarSherlock Library [11]. The CP can leverage any of these themes provided by the framework without having to worry about creating another. Alternatively, he can create customised themes using the Online Android Action Bar Style Generator**.** Finally, the framework was test-run using real and virtual Android devices of different APIs, ranging from 8 to 17. However, for the purpose of presentation, we used emulators (API 13) for phone and tablet, as we will show later.

4.1 Framework Packages

The framework was built as an Android project named `NMMLAFramework`. It is made up of nine (9) packages. Table 4.1 shows the names of these packages and their content. The packages are organized such that most data-model classes are kept in the datamodel package, while classes that closely interact or cooperate to realize a task are kept together in the same package. The root package name is `com.austuniaizu.koyham.nmmla.framework`, to which each of the names in the package column, except the first row, in Table 4.1 is appended to form a complete package name.

Table 4.1. Framework Packages and Composition

S/N	Package	Class Composition
1.	[Root Package]	`AppController, Version`
2.	`Datamodel`	`File, Simulation, Content, Question, Quiz, Item, Course, CourseBundle, Theme, ThemeBundle, Module, Component, AtomicComponent, ModularComponent, SimComponent, EvalComponent, RenderingMode`
3.	`Homepage`	`FilesDBHelper, HomePageGridItemModelAdapter, HompageGridFragment, HomePageFragmentActivity`
4.	`TabFile`	`TabFileFragmentSPAdapter, TabFileFragment, TabFileFragmentActivity`
5.	`Listmodule`	`ListModuleFragment, ListItemFragment, ListModuleFragmentActivity`
6.	`Tabmodule`	`TabModuleFragmentSPAdapter, TabModuleFragment, TabModuleFragmentActivity`
7.	`Evaluate`	`QuestionsDBHelper, QuestionActivity, ScoreActivity, AnswersActivity`
8.	`Search`	`DictionaryDatabase, DictionaryProvider, SearchDictionaryActivty, WordActivity`
9.	`Util`	`FrameworkListFragment, ModuleOptionsMenu, CommonMenuOptionsItemSelectHelper, Utility, ModulesViewQuizSimOptionsMenu, DBHelper, DetailViewQuizOptionsMenu, HtmlHandlerActivity, ImageHandlerActivity, AudioHandlerActivity, VideoHandlerActivity`

4.1.1 Root Package

The root package is the overall package in which the other packages and special (`Application` and `Version`) classes are contained. The `Application` class is where we declared and initialized global variables (resources such as themes, component icons, module icons, item icons etc) visible across the application (as `final` and `static`), while the `Version` class contains the version number of the Android application being developed. For our purpose, the version number of the framework is 1.

4.1.2 DataModel

This package contains the data models or structures in the framework, which are manipulated or operated on by other controller classes in other packages. See Fig. 1.6 for the Framework Data Model (FDM).

4.1.3 HomePage

The `HomePage` package is the entry door into the framework during its instantiation, as it contains, among other non-`activity` classes, the Component Router, `HomePageFragmentActivity`, which is responsible for receiving content from the Course Loader, unbundling it and routing the constituent components to the other controllers in the application. See the Activity Diagram in Fig. 4.2 for the data and control flow.

4.1.4 TabFile

This package is responsible for rendering files in navigational tabs on the screen. It contains the self-contained module/item dispatcher, `TabFileFragmentActivity`, which cooperates with `TabFileFragmentSPAdapter` and `TabFileFragment` classes to render HTML and image files as fragments in respective `tabviews` (`TVs`). See Fig. 4.1b in Section 4.4.

4.1.5 ListModule

This package is responsible, through the module/item dispatcher, `ListModuleFragmentActivity`, for rendering a list of `Module`, `File`, `Simulation` and `Quiz` items in a `listview` (`LV`) container hosted by `LMFA`. `ListModuleFragment and ListItemFragment` cooperate with `LMFA` and are responsible for returning the list of items via an adapter as a fragment to the `listview` container.

4.1.6 TabModule

The `TabModule` package is responsible for handling tabbed `modules` containing a list of `items`. The module/item dispatcher, `TabModuleFragmentActivity`, renders instances of `TabModuleFragment` in a `viewpager` with the help of an object of `TabModuleFragmentSPAdapter`.

4.1.7 Evaluate

The `Evaluate` package is responsible for handling quiz sessions in an instance application. It contains `QuestionActivity` (which renders the quiz's questions), `ScoreActivity` (which renders the learner's score) and `AnswersActivity` (which displays all answers to the questions attempted by learner at the end of a quiz). The `Question` class contains static attributes such as `databaseName`, `tableName` and `numQuestions` (see Fig. 4.2), which must be set before a quiz can be taken, while the `Quiz` class is used to set the `currentQuiz` attribute of `QuestionActivity`. `QuestionsDBHelper`, which inherits from `DBHelper` in `Util` package, is used to fetch `questions` to be answered (the number of which is determined by `numQuestions`) from an SQLite database.

4.1.8 Search

The `Search` package is responsible for accepting search words and displaying search results. The `DictionaryProvider` provides access to an application's dictionary database and handles all the searches and suggestion queries from a `SearchManager` class invoked within. The `DictionaryDatabase` is responsible for handling the logic to return specific words from the dictionary, and loads the dictionary table when it needs to be created [6]. `SearchDictionaryActivty`, being the main `activity` in the package, is responsible for displaying search results triggered by the search dialog and handling actions from search suggestions. Finally, the `WordActivity` class is responsible for displaying the word (selected in `SearchDictionaryActivty`) searched for and its definition (meaning).

4.1.9 Util

This package contains classes, which serves as utilities to other classes in other packages. Examples include `HtmlHandlerActivity`, `AudioHandlerActivity` etc, to which `intent`s are sent from any of the `FA`s to display the detail views of `file`s on the screen. Also, the `Util` package contains base classes, e.g `DBHelper` and `FrameworkListFragment`, which other classes in other packages inherit from.

4.2 Framework Overview Schematic

Fig. 4.1 shows an overview of the framework when instantiated, from the perspective of content flow. It is the Android implementation of the Content Flow Algorithm Tree in Fig. 1.7, which helps in providing a better and clearer understanding of the Activity Diagram in Fig. 4.2. Thus, it shall be explained alongside Fig. 4.2 in Section 4.4.

4.3 Framework Implementation Activity Diagram

An Activity Diagram (AD) is a UML diagram that shows the business process or complex operations of a system. For easier understanding of the framework and how the NMMLA framework works, an AD was drawn to abstract the flow of data (content) and control in an instance application (see Fig. 4.2). This diagram was drawn using Microsoft Visio.

4.4 NMMLA Framework Implementation Algorithm

The framework comprises four top-level controller `activities`, which are `FA`s and outlined as follows:

1. `HomePageFragmentActivity` (`HPFA`)
2. `TabFileFragmentActivity` (`TFFA`)
3. `TabModuleFragmentActivity` (`TMFA`)
4. `ListModuleFragmentActivity` (`LMFA`)

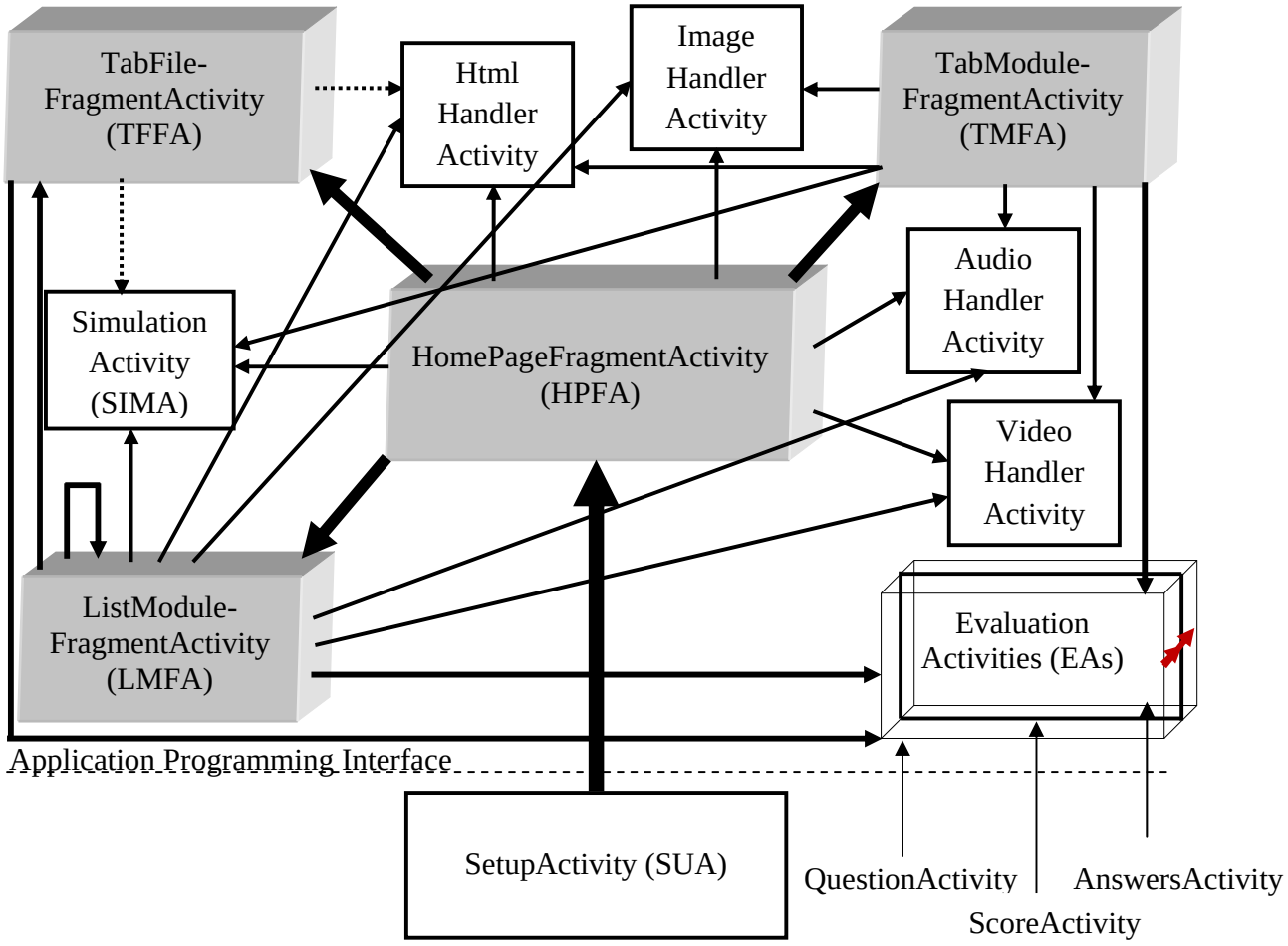
Fig. 4.1 shows the Android implementation of the CFAT. It provides an abstracted view of the functionality of the framework from `activity`'s standpoint. The four `FA`s together with the

Evaluation Activities (EAs) are correspondingly shown in a swim-lane AD in Fig 4.2. They are responsible for handling and controlling the content flow in an instance application. The HPFA provides the entry point into the framework from the CP's SetupActivity (SUA), where the application content is set up and uploaded from sources such as SQLite databases, resource and asset folders.

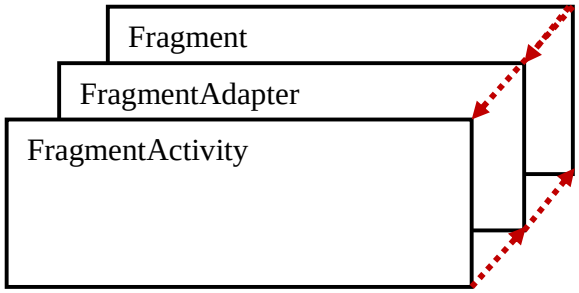
In Fig. 4.1, the boldest arrow (courseBundle) from SUA to HPFA denotes uploading content into the framework to realize a complete functional NMMLA. Similarly, the bolder arrows (components) originating from HPFA to the other FAs depict the routing (via an intent) of different components, which make up a course, to respective dispatchers.

The bold arrows (module) originating from LMFA and terminating in LMFA and TFFA symbolize sending a module from LMFA to self and TFFA, to render the module's items once again as items in a LV or detail-view files in TVs respectively, when the learner selects a module in a LV. Similarly, the bold arrows originating from LMFA/TMFA/TFFA and terminating in QuestionActivity in Evaluation Activities denote both sending a quiz model extracted from a module model (used to preset the static Quiz attribute of QuestionActivity), and sending an intent to open QuestionActivity for mid- and post- evaluation. See Lane 4 in Fig. 4.2. (Note: pre-valuation is also possible on the HP, but then, the sending of an intent to QuestionActivity still has to pass through LMFA, as all the modular quiz items in the application are deliberately collected by referencing to the application's modular component (where each constituent module contains a quiz) using EvalComponent object) and rendered in a LV in LMFA as the application's Evaluation component. This is the reason there are no direct arrows from HPFA to EAs.)

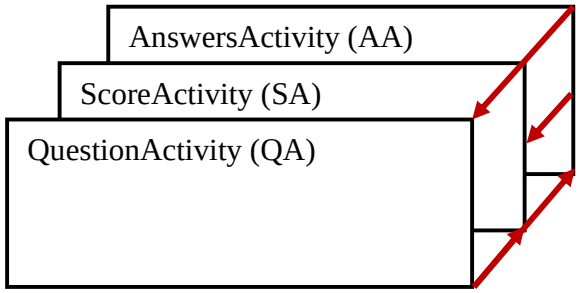
The least bold arrows (file item) originating from FAs and terminating in atomic handler activities symbolize the sending of a file name, selected by the learner in a LV, to their respective atomic handler activities for DV rendering after being loaded from the filesystem. Similarly, the least bold arrow (sim item) from FAs to SIMA symbolizes the opening of a simulation activity through the sending of an intent to the simulation activity to bring it up on screen by using its "class" attribute, i.e. its ".class" name. This occurs when the learner either selects an atomic simulation component in the HP GV or a simulation item from a LV or AB menu in LMFA/TMFA/TFFA. Note: the broken lines from TFFA to SIMA show that a simulation activity can only be opened from the AB when the learner is in TFFA. The same explanation holds for that from TFFA to Html handler activity.



(a) NMMLS Framework with Composed (Fragment) Activities



(b) FragmentActivity Decomposed



(c) Evaluate Activities Decomposed

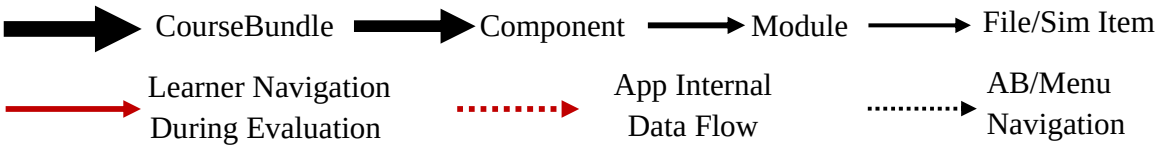
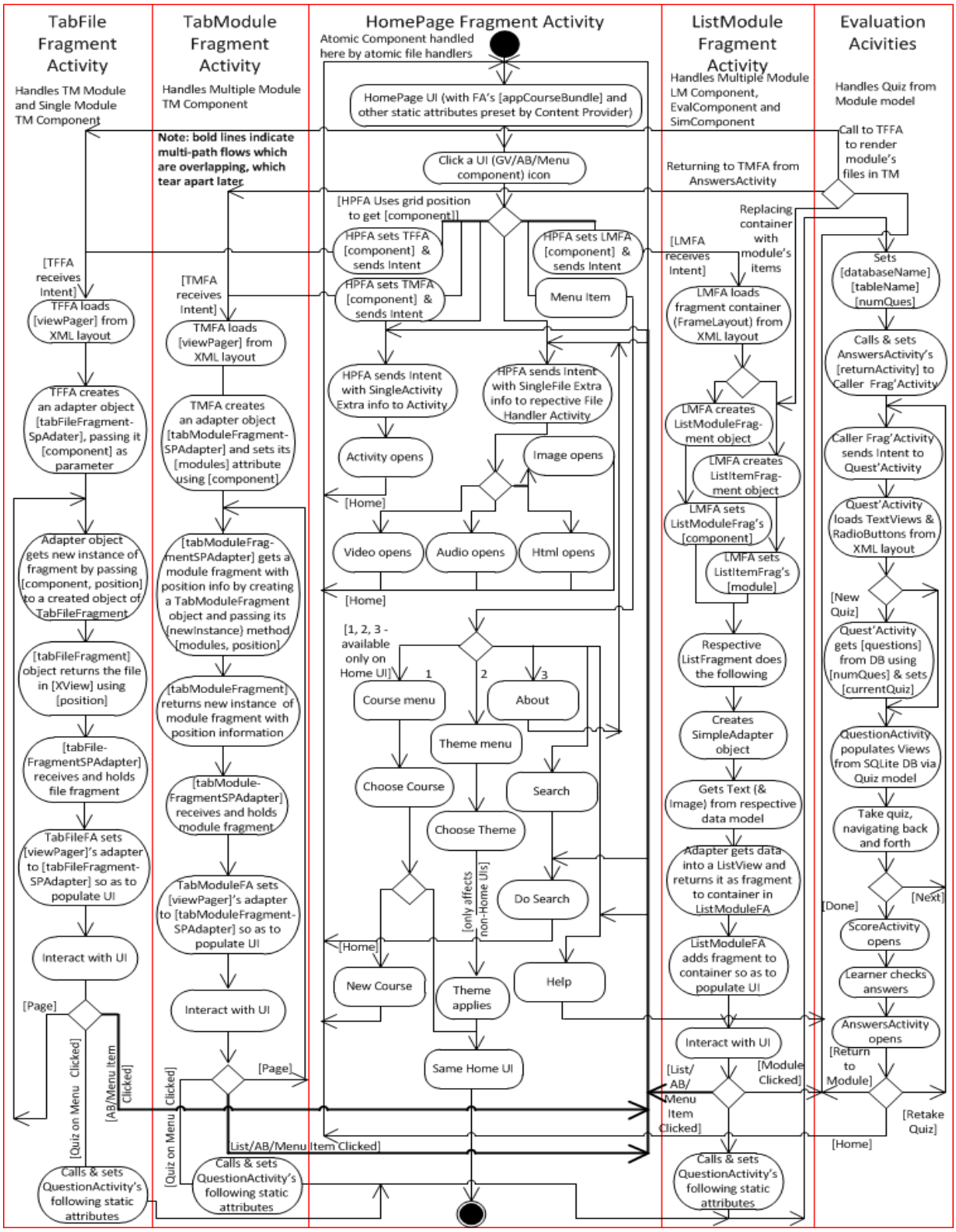


Figure 4.1. Implementation of the Content Flow Algorithm Tree in Android



Note: All cooperative classes (FA, Adapter & Fragment) are grouped within the FA swim-lane

Figure 4.2. Swim-Lane Activity Diagram for Content Flow Algorithm Tree

Fig. 4.1b shows a mechanism of how a module/item dispatcher, which is a FA, and other cooperative classes (FragmentAdapter and Fragment, not shown in Fig 4.1a), interact with one another internally to render data models in a section of the FA's view. The arrow, which points from FA to Fragment via FragmentAdapter and back to FA, shows, in a nutshell, how application content flows between the FA and the Fragment class via an adapter. Fig. 4.1c, on the hand, portrays a clearer picture of the individual activities, which make up Evaluation Activities and how the learner can navigate through them during a quiz session. We will now briefly discuss what goes on in the AD in Fig. 4.2 when an instance application is created.

4.4.1 HomePageFragmentActivity

From the SetupActivity (see Appendix C), after the application content has been setup using the relevant File, Sim, Quiz, Item, Module, Component and Course models, with a number of courses bundled in a courseBundle object, an intent is sent to the HPFA, in the middle and third lane in Fig. 4.2, to render the first course's components added to the bundle as functional icons in the HP GV. For simplicity and clarity, the inner processes that take place within the application in the course of rendering the components on the HP screen are not shown. However, Fig. 4.1b simply portrays how data flows internally between a FragmentActivity and its cooperative classes (FragmentAdapter and Fragment) to render a fragment in a sub-section of FA in question. Upon the rendering of the components on the HP, the HPFA takes over control of the application, depending on the learner's action or event in terms of the component clicked in the GV or on the AB.

If the learner clicks on an atomic component, for example, with a single simulation activity, HPFA uses the ".class" name of the activity to send an intent directly to it (the activity), which then comes up on the screen straightaway. On clicking the Back button, the learner returns to the HP as depicted in the AD. (Note, for the sake of simplicity and clarity, once again, the Back button operation is not captured in the Fig. 4.1a. For similar reason (space in particular), the AtomicFile Handlers and simulation activities shown alone in Fig. 4.1 are shown within the HPFA swim-lane in Fig. 4.2. Similar reason accounts for the AB/menu item operations such as Search, Theme, About and Help, shown within the HPFA swim-lane.)

4.4.2 TabFileFragmentActivity

If the user clicks on a single-module component having TM as render mode, HPFA calls TFFA to set its Component attribute to the clicked component model, and sends an intent to the TFFA to take over control. TFFA then loads a Viewpager widget defined in the XML layout into a corresponding viewPager variable declared in the code (and follows the flow described in Fig. 4.1b to render the file (usually HTML and image) on the screen). TFFA creates an instance of FragmentStatePageAdapter and uses it to fetch a file fragment into

XView (e.g. `webview` or `imageview`) from an object of `TabFileFragment`, which is then rendered on the `viewpager` on the screen. The learner can then interact with the UI. If he decides to page or tab through the files, the whole process is repeated to render the new file fragment in the view (see Fig 4.1b and Fig. 4.2). Otherwise, if he clicks an AB/menu item, the appropriate `activity` opens. (Note: the bold lines connote multiple paths are overlapping. This is deliberately done so as to reduce the number of lines criss-crossing. On getting to its destination, each line branches off.)

4.4.3 TabModuleFragmentActivity

If the user clicks on a TM component on the HP, `HPFA` calls `TMFA` to set its static `Component` attribute to the clicked component model, and then sends an `intent` to `TMFA` to take over control. The same process is followed (as explained in the previous subsection about `TFFA`) to render a module fragment on the screen, which is a tabbed list of items. However, if the learner chooses a quiz item on the AB menu, which is part of the `TMFA` UI, the control is passed on to `EAs` to begin a quiz session. This shall be explained in Subsection 4.4.5. Alternatively, if the learner clicks on an item in the presented list, the detail view of that item comes up on the screen, as indicated by the bold lines running to the top of `HPFA` and then branching to the left underneath the "Menu Item" state symbol.

4.4.4 ListModuleFragmentActivity

If the user clicks on a LM component, `HPFA` calls `LMFA` to preset its static `Component` attribute to the clicked component model, and sends an `intent` to `LMFA` to take over control. `LMFA` receives the `intent` and goes ahead to load a fragment container from the XML layout. Depending on the type of data model (single- or multiple-module component, or just a module) sent to it, the `LMFA` creates a `ListFragment` object and uses a setter method to set its `Component` or `Module` attribute. The `ListFragment` object (as carried out within the respective `ListModuleFragment` or `ListItemFragment` class, not shown here) creates a `SimpleAdapter` object which gets the data (text and image for each item in the list of `modules` or list of `items`). The adapter puts the data into a `listview` and returns it as a fragment to `LMFA`, which then adds it to its `LV` container in the current `FA`'s UI as shown in Fig. 4.1b. The user can then interact with the UI.

If the list rendered on the screen is a list of `modules` with each module having a LM render mode, and the learner selects one of the items by clicking, the whole process repeats as shown by the path running from the decision box in the fifth lane to the "`LMFA` creates ListItemFragment object" process within the `LMFA` lane. Similarly, the one running to `TFFA` denotes rendering the `items` in the `module` in TM if the clicked module has this as RM. Finally, the path running to `TMFA` originates from the fifth lane, which we shall discuss in Subsection 4.4.5.

However, if the list rendered on the screen is a list of `items` and the user selects any of it, or he clicks on an AB/menu item (such as `Search`, `Help`, `Home` etc), the appropriate `activity` opens. This is shown by the overlapping path running from the decision box near the bottom of `LMFA` in the fourth lane, veering to the left and then top, and finally tearing apart at various points to open the appropriate `activity`, which is either a user-defined simulation `activity` or HTML/image/audio/video handler `activity`, or `HPFA`.

Finally, if the user chooses a quiz option on the menu, control is passed to `Evaluation Activities` for the quiz session to begin after `QuestionActivity`'s static attributes (`databaseName`, `tableName`, `numQuestions` and `returnActivity`) relevant to loading the quiz from the SQLite database and returning to the calling `FA` respectively must have been preset using setter methods in the caller `FragmentActivity` class. However, for lack of space, this setting of `QuestionActivity`'s attributes is shown within the `Evaluation Activities` lane, as against in the respective caller `FA` class lane which actually does this and sends an `intent` to `QuestionActivity` to begin the quiz session.

4.4.5 Evaluation Activities

Finally, if the user clicks on a component of type `EvalComponent` on the HP, as explained in the foregoing section, `HPFA` calls `LMFA` to set its static `Component` attribute to the clicked `component` model, and then sends an `intent` to `LMFA` to take over control. `LMFA` follows the same path as in Lane 4 to the point where the application's quiz items are added to the `LV` fragment container as cellular pairs of texts and images, which the learner can select from.

When the learner clicks one of the quiz items, the caller `FA`, in this case `LMFA`, sets `QuestionActivity`'s static attributes (`databaseName`, `tableName`, `numQuestions` and `returnActivity`), which are relevant to loading the questions/options from the SQLite database and returning to the calling `FA` at the end of the session respectively. The caller `FA` then sends an `intent` to `QuestionActivity` for the quiz session to begin. This point can also be reached from the AB menus of `TMFA` and `TFFA` as shown in the first and second lanes.

From here, `QA` loads the XML layout (`textview` and `radiobuttons`), which it will use to render the question and option texts respectively on the screen. Then, if no quiz has been setup before, `QA` gets `questions` (a set of questions) from the SQLite DB and uses it and `numQuestions` value as parameters to create a `Quiz` object, which it assigns to its `currentQuiz` attribute. Then `QA` then populates the `views` using the `currentQuiz` data model, one question and a set of options at a time. Then the user can interact with the rendered UI by choosing an option and clicking `Next` to move to the next question. When the learner has finished taking his quiz, the `ScoreActivity` renders the scores on the screen, together with a "Return to Module" and "Check Answers" buttons. On clicking on the latter, the `AnswersActivity` renders all the answers (wrong and right) to the attempted questions on the screen. On the other hand, on clicking on the former, the learner returns to the point of entry

(either of HPFA, LMFA, TMFA and TFFA) into the quiz session. The latter also provides the learner with the optional buttons of retaking another quiz or returning to the point of entry or HP using the Home button icon on the AB.

4.5 Framework APIs

A set of APIs is exposed to the content provider through the HomePageFragmentActivity (HPFA) to enable him to instantiate the framework. (See Fig. 1.6 for the HPFA class.) The APIs together with their arguments and return types are presented in Table 4.2. The first in the Table 4.2 is setAppName(…) with a String parameter. It is used to set the name which the CP wants his NMMLA to bear and thus reflect alongside the application logo on the left hand corner of the AB. Similarly, setAppPackageName(…) and setAppDatabasePath(…)are used to convey the CP's project root package name and database path which are required internally by the framework to enable it load files such as audio and video, and databases respectively from the appropriate folder in the filesystem. See the SetupActivity in Appendix C.

Table 4.2 Framework APIs

S/N	HomePageFragmentActivity API	Argument Type	Return Type
1.	setAppName(…)	String	Void
2.	getAppName()	-	String
3.	setAbout(…)	About	Void
4.	getAbout()	-	AtomicComponent
5.	setHelp(…)	Help	Void
6.	getHelp(…)	-	ModularComponent
7.	setSearch(…)	-	Void
8.	getSearch()	Search	AtomicComponent
9.	setAppDatabasePath(…)	String	Void
10.	getAppDatabasePath()	-	String
11.	setAppPackageName(…)	String	Void
12.	getAppPackageName()	-	String
13.	setAppCourseBundle(…)	CourseBundle	Void
14.	getAppCourseBundle()	-	CourseBundle
15.	setAppThemeBundle(…)	ThemeBundle	Void
13.	getAppThemeBundle()	-	ThemeBundle

4.6 Key SetupActivity's Data Models and Constructors

Table 4.3 shows the main constructors of key data models used within the SetupActivity during the instantiation of the framework. The first File has four (4) arguments, the first of which is the file name, which should display as an item in a listview and on the AB when the item is selected. The second is the image that should display alongside the item's text, i.e. file name. The second Quiz is the data model used in retrieving quizzes from the SQLite database. The first argument stands for the database name, the second the table name while the third is the number of questions that should be administered to the learner. This is left at the discretion of

the content provider. The third `Module` is used to retrieve a module comprising file items from a database and/or folder in the filesystem. The first argument "fileName" stands for the name of the module that should display as an item in a `listview activity` or on the AB when selected. The second "isIconType" determines whether the file items in a `listview` should display an image along the text or not. The third argument "simulations" stands for the simulations that are associated or assigned to a module. These simulations are manually added in the `SetupActivity` page (see Appendix C). It could be one or more. The fourth argument "quiz" stands for the quiz assigned to the module in question. The fifth argument "renderingMode" represents the render mode which the CP prefers the module's items to render as, which could either be list or tab. Finally, the last argument "mediaType" determine the type of media which the composed items are: HTML, image, video or audio? The explanation for the rest classes follows from the names of the arguments in the constructors.

Table 4.3. SetupActivity's Data Models and Constructors

S/N	Model	Constructor Arguments
1.	`File`	`(String fileName, int image, String fileEntryName, int fileType)`
2.	`Quiz`	`Quiz(String databaseName, String tableName, int numQuestions)`
3.	`Module`	`(String name, boolean isIconType, List<File> files, List<Simulation> simulations, Quiz quiz, RenderingMode renderingMode, int mediaType)`
4.	`AtomicComponent`	`(String name, int image, Content content)`
5.	`ModularComponent`	`(String name, int image, List<Module> modules, RenderingMode renderingMode)`
5.	`SimComponent`	`(String name, int image, ModularComponent component)`
6.	`EvalComponent`	
7.	`FilesDBHelper`	`(Context context, String databaseName, String tableName, int numItems, String databasePath)`
8.	`QuestionsDBHelper`	`(Context context, String databaseName, String tableName, int numQuestions, String databasePath)`
9.	`Course`	`(String name, List<Object> components)`
10.	`Theme`	It encompasses all and follows from the order of the attributes in the `Theme` class in Fig. 1.6.
11.	`CourseBundle`	`(String name, int image, List<Course> courses)`
12.	`ThemeBundle`	`(String name, int image, List<Theme> themes)`

Chapter 5
Presentation and Discussion of Results

In this chapter, we present the result of the implementation of the framework, which is graphical (user interfaces) by the very nature of this thesis. This outcome is realised by feeding the framework within the Eclipse IDE with a test input (multimedia learning content), running it on an Android emulator (phone and tablet) and getting the output, the first of which is the Homepage User Interface (HUI), which the learner can start interacting with in order to explore and experience the different functionalities, features, looks and feels, offered by the framework. First, we present the key features of the instantiated framework application. Second, we present the outcome (with relevant screenshots) of the instantiation.

5.1 Framework Key Features

The framework has a number of special features which enable it to realize the functional requirements specified in the UCD. These features provide the learner with a heightened learning user experience. They are briefly discussed in the following subsections. See Table 5.1 for a summary of these features.

5.1.1 About

This is a HTML file, represented by an icon, pinned to the AB of the HUI. Usually, it provides information on what the application is about and does.

5.1.2 Course Menu

This is a menu, which representative icon is pinned to the AB (see Fig. 5.1) of the HUI. It enables the learner to choose from a menu of courses provided by the content developer.

5.1.3 Theme Menu

This is a menu (icon) pinned to the AB of the HUI. It enables the learner to decide the look and feel of his application by being able to choose any theme of his choice. The theme can be customized by the content provider by uploading his own set of themes.

5.1.4 Quiz Menu

This is a menu provided in all the FAs' UI ABs except HPFA. Before selecting a module to study, apart from the HPFA, the learner can pre-evaluate himself within the application by choosing from the quiz options offered in the menu. Also, this capability—just one quiz item represented by an icon or text on the actionbar—is offered in the detail view page of each file contained in a quiz module.

5.1.5 Sim Menu

The Sim menu just like the Quiz menu is provided in all the FAs' UI ABs except HPFA. This enables the learner to select and navigate to a simulation activity when he is studying a module.

5.1.6 Search

The Search tool enables the learner to look up words in the application's dictionary if provided. It is pinned to the actionbar of every `activity` throughout the application.

5.1.7 Help

This is a utility component pinned to the AB of every `activity`. This enables the learner to seek help on the usage of any component or part of the application.

5.1.8 Screen Mode

The Screen mode is provided in every `activity`, except HPFA (see Table 5.1). This enables the learner to make a choice from the two options provided, namely, full screen and default screen modes. The former allows the learner to have a full view of the current content on the screen by hiding the actionbar. The default mode restores the actionbar to the screen.

5.1.8 Render Mode

The Render mode is provided in HPFA and LMFA. This enables the learner to make a choice, according to his preference, from the two options provided, namely, LM and TM. See Table 5.1 for its location in the application.

5.1.9 Sequencing Capability

The framework provides the learner with the ability to sequence through (back and forth) his RLOs such as HTML, audio and video files. It also enables him to go back and forth quiz's questions during an evaluation session until he has seen his score in the `ScoreActivity` in which case he cannot return to a taken quiz except he decides to retake a new one.

Table 5.1 Framework Application Key Features.

Features	**Content/Items**	**Location**
About	HTML file	HPFA only
Help utility	List of HTML files	All `activities`' ABs
Search utility	Text file rendered in a listview	All activities' Abs
Course menu	Course 1, Course 2, Course 3 ...	HPFA only
Theme menu	Black, Blue, Green	HPFA only
Quiz menu	Quiz 1, Quiz 2, Quiz 3 …	All FAs except HPFA
Sim menu	Sim 1, Sim 2, Sim 3 …	All FAs except HPFA
Screen mode	Default and Full Screen	Some `Activities`/FAs
Render mode	List and Tab	HPFA and LMFA
Sequencing	Previous and next	File Handler `Activities` and `QA`

5.2 Instantiating the Framework

We shall create an application for three Computer Science courses, namely, Automata, Compiler and Data Structures. These are some of the important courses in Computer Science, as understanding them helps students in the area of programming, which is fundamental to Computer Science. Each of the courses shall contain components such as Introduction, Learn (Doc, Slides and Video), Simulate, Evaluate, Resources and Help.

To have a complete functional NMMLA, the framework has to be instantiated as shown in Fig. 1.2 in Chapter 1. To realize this, it is added as a library to the content provider's NMMLA project in Eclipse IDE and the required classes are imported into the `SetupActivity`. Then, content is fed into the framework by setting the appropriate `HPFA`'s class or static attributes (i.e. APIs). The setup activity also serves as a splash screen.

The content, which includes HTML/images/quiz, video and simulation classes, are fetched from the `asset`, `raw` and `src` folders respectively. For the HTML and images (slides), each file is hierarchically stored in a module folder, contained in their respective folders named with an "html" and "slide" appendage respectively. For example, as shown in Fig. 5.1, for a Learn-Doc component, an HTML file could be saved in "module1" folder, and "module1" in "learn_html_doc" folder, while this folder is in turn saved in the asset folder. The same applies to the images. Each of the files is read directly into its equivalent `File` model from the containing module folder during project setup. (See the `SetupActivity` in Appendix C.) However, for the video content (just like audio), the entry name of each video item is looked up from the SQLite database stored in the `asset` folder, while the video files are loaded from the `raw` folder. Since the `raw` folder does not allow the organizations of files in hierarchy, the video files are stored in the same way as in Fig. 5.1, only that in place of the folder names we have appendages to the filenames. For example, for a file item in "module 1" in Learn-Video component, the name is written thus "learn_video_module1_filename". Finally, for the quiz, no look up is required, the quiz, since it comprises texts, is directly loaded from the SQLite database file stored in the `asset` folder. (See the `SetupActivity` in Appendix C.) For single file component such as Resources, the HTML file is directly stored in the resource folder ("resource_html") hosted in the `asset` folder.

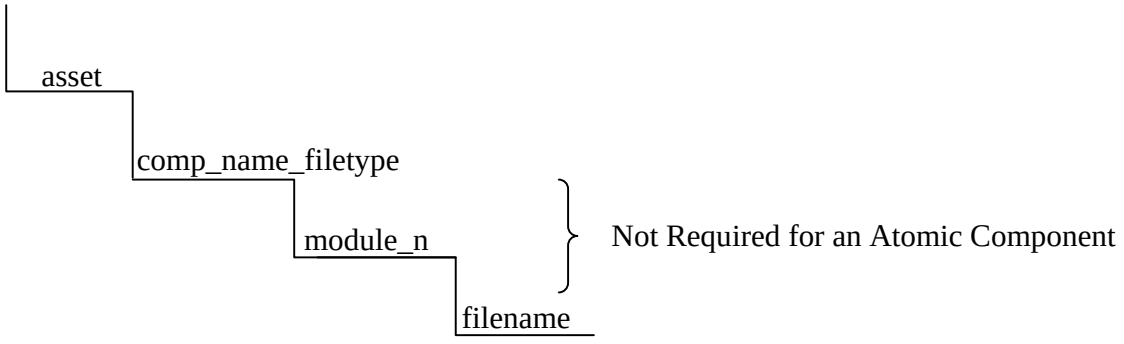


Figure 5.1. Organization of HTML and Image Files in Asset Folder

5.2.1 Setting up the Application

To set up a mobile application, the content provider has to carry out the following:

1. Set up his Android development environment in the Eclipse or other IDE by installing the Android SDK and plugging in the necessary development tools such as ADT.
2. Acquire the framework library "NMMLAFramework" and ActionbarSherlock library.
3. Create an Android application project in the IDE.
4. Create an `activity` named `SetupActivity` (or any other name) by extending Android's `activity` class or `SherlockActivity`.
5. Store his multimedia contents into the appropriate folders as follows:
   a) HTML/Image files into `asset` folder
   b) Audio and video files into `raw` folder
   c) SQLite (and module) databases into `asset` folder
   d) Application launch icon and customized thematic resources (optional) into the appropriate `drawable` subfolders, which is contained in `res` folder
6. Import the NMMLAFramework and ActionbarSherlock libraries and follow the `SetupActivity` example provided in Appendix C.
7. Duplicate the declarations in the NMMLAFramework manifest file in Appendix C by copying it to your own manifest. This is necessary as explained in Section 6.4.

5.2.2 Running the Application

After running the application on the test emulators (phone and tablet), we will be presented straightaway with the HUI. Fig. 5.2 shows the resultant application with all the components laid out in the GV and on the AB. Also shown is the navigation level (NL). The displayed components—Introduction, Learn (Doc, Slide and Video), Simulate, Evaluate, Resources and Help—offer a complete functional NMMLA. Introduction introduces each of the components in the application. The Learn components contain a number of modules, which in turn contain file items. The various Learn components allow different learners with different learning preferences to make a choice. Simulate component collects by reference to Learn-Doc (see Fig. 1.6) all the modular simulation items and presents them as a component on the HP. As such, the learner is able to have quick access to the application simulations on the HP without having to drill deep into the application. The same applies to the Evaluate component on the HP, which collects by reference to Learn-Doc likewise all the quiz items (one per module) and presents them as a list. This enables the learner to pre-evaluate himself on the HP before going into the application to study any of the modules in the Learn components. Resources is a HTML file, which presents a list of hyperlinks to online resources. This enables learners with internet access to get more information on learnt topics online. Finally, Help provides a list of items, which dwell on the usage of the application. This is provided in the HP GV for quick access in addition to that on the AB. Both of them point to the same component (see Appendix C.) The rainbow icon item on the AB stands for the theme menu, while the rightmost triple-square symbol represents other menus as shown in Fig. 5.2a.

Figure 5.2a. Homepage User Interface on Phone (NL = 0)

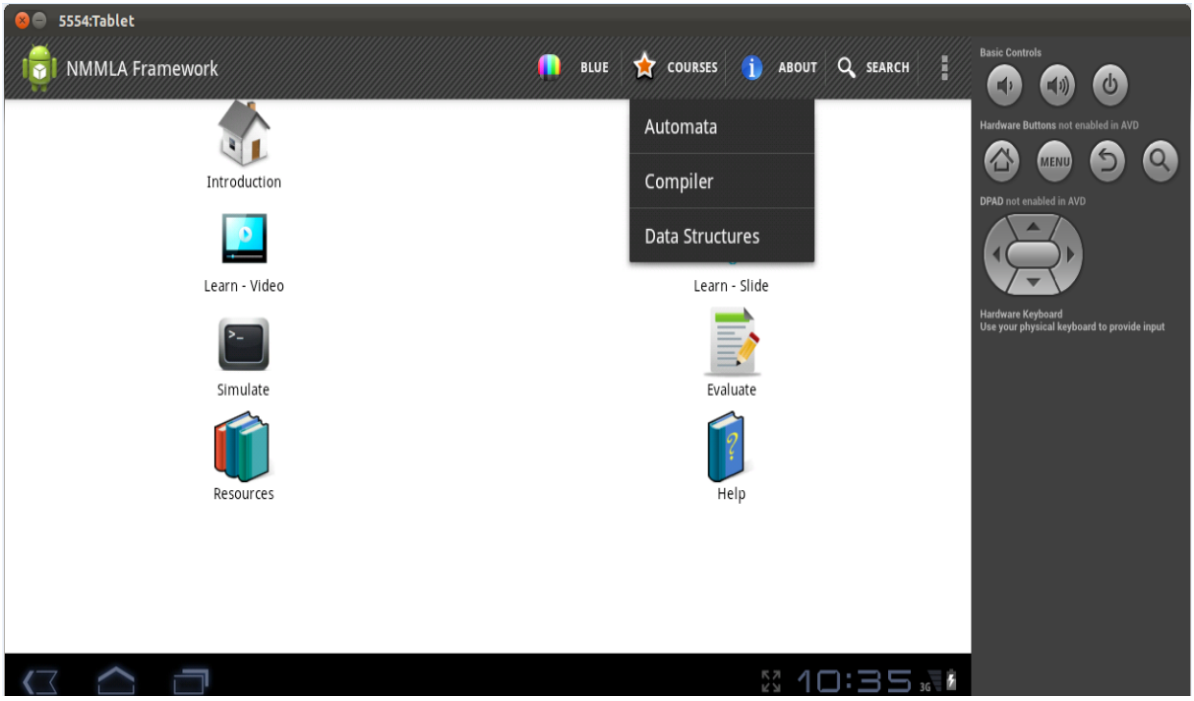


Figure 5.2b. Homepage User Interface on Tablet (NL =0)

5.2.3 Navigating Through the Application

In this section, we present a series of snapshots of the major components in Course 1 as the learner navigates through or drill deep into the application.

5.2.3.1 Introduction

Fig 5.3 shows the Introduction component. The first snapshot shows it in list mode, with each item opening up a detail view when clicked, while the second shows it in tab mode with each tab introducing every other component in the application. Fig. 5.3b is the tablet equivalent.

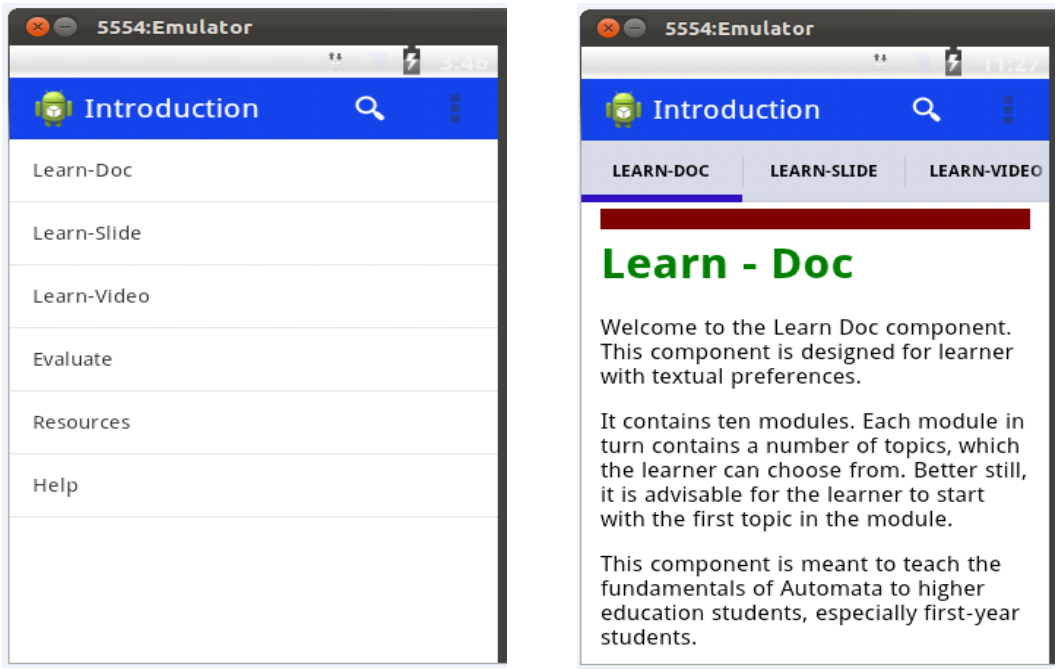


Figure 5.3a. Introduction of Learn-Doc Component on Phone (NL=1, NL=1)

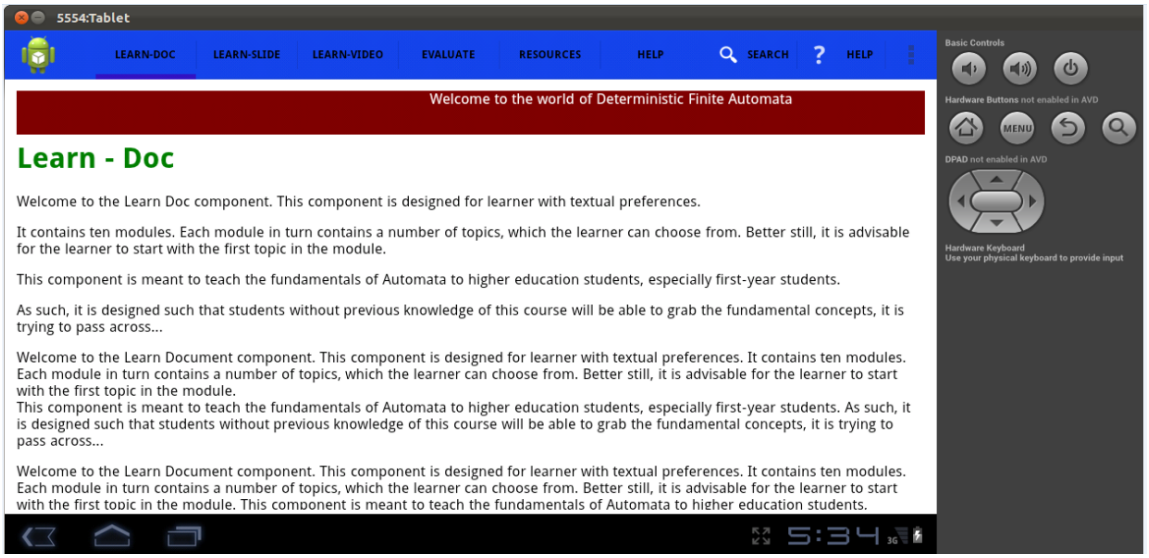


Figure 5.3b. Introduction of Learn-Doc Component on Tablet (NL=1)

5.2.3.2 Learn – Doc

Fig 5.4a shows a listview of all the HTML modules contained in the Learn-Doc component and the tabview of Module 1's HTML files on an Android phone. The learner can select a module in the left to open a tabview as on the right. Alternatively, the learner can choose the listview of a module's items using the Render Mode menu. Fig. 4b, for example, shows the list of items in Module 1 as well as all the modules of the List-Doc component on a tablet.

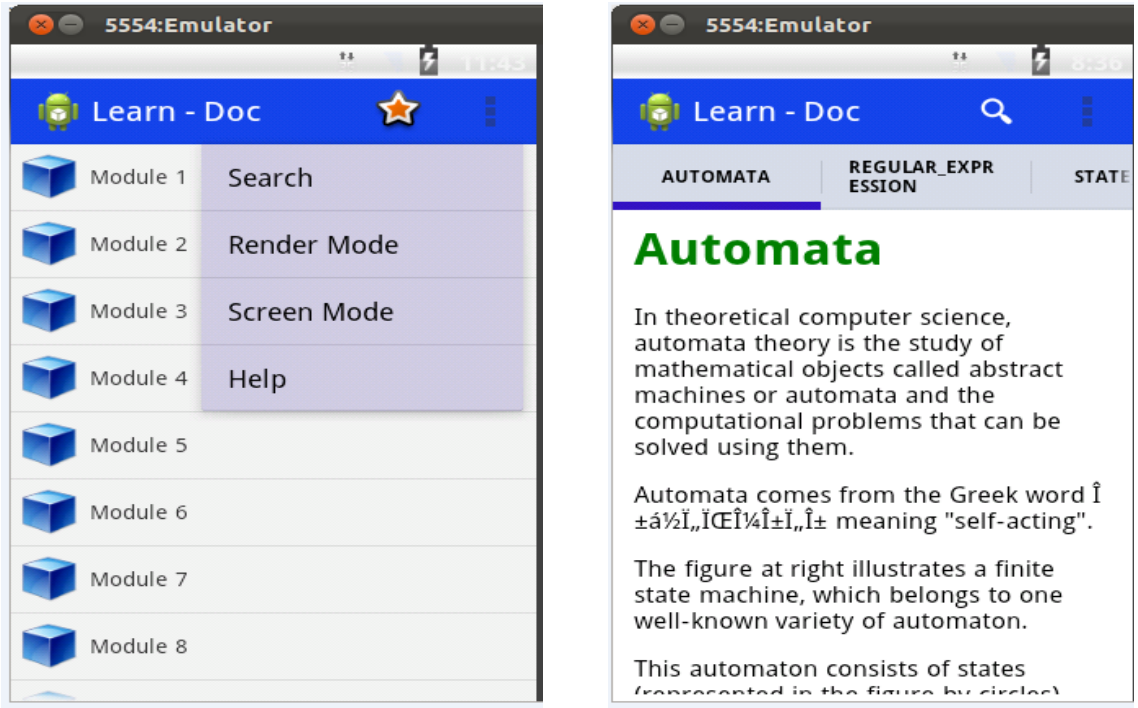


Figure 5.4a. Learn-Doc Modules and Module 1's Tabview on Phone (NL=1, NL=1)

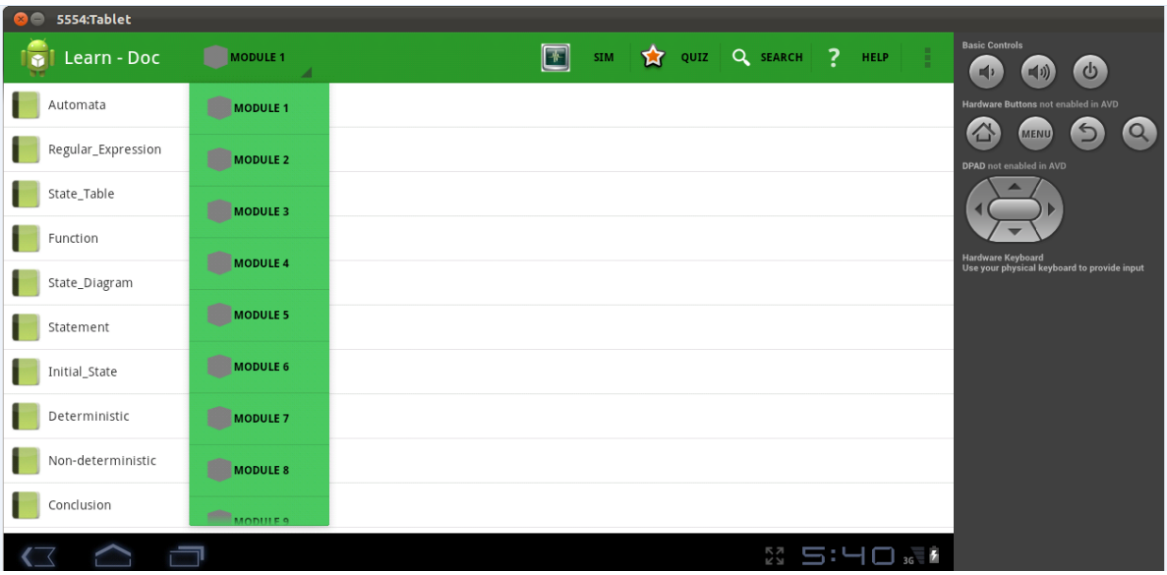


Figure 5.4b. Learn-Doc List of Module 1's Items and HTML Modules on Tablet (NL=1)

5.2.3.3 Learn –Video

Fig 5.5a shows a listview of the video items in a particular module and a detail view of a clicked video item on the phone. The star icon on the AB allows the learner to quickly take a quiz, when done watching a video. Also, Fig. 5.5b shows the equivalent of the second snapshot on a tablet.

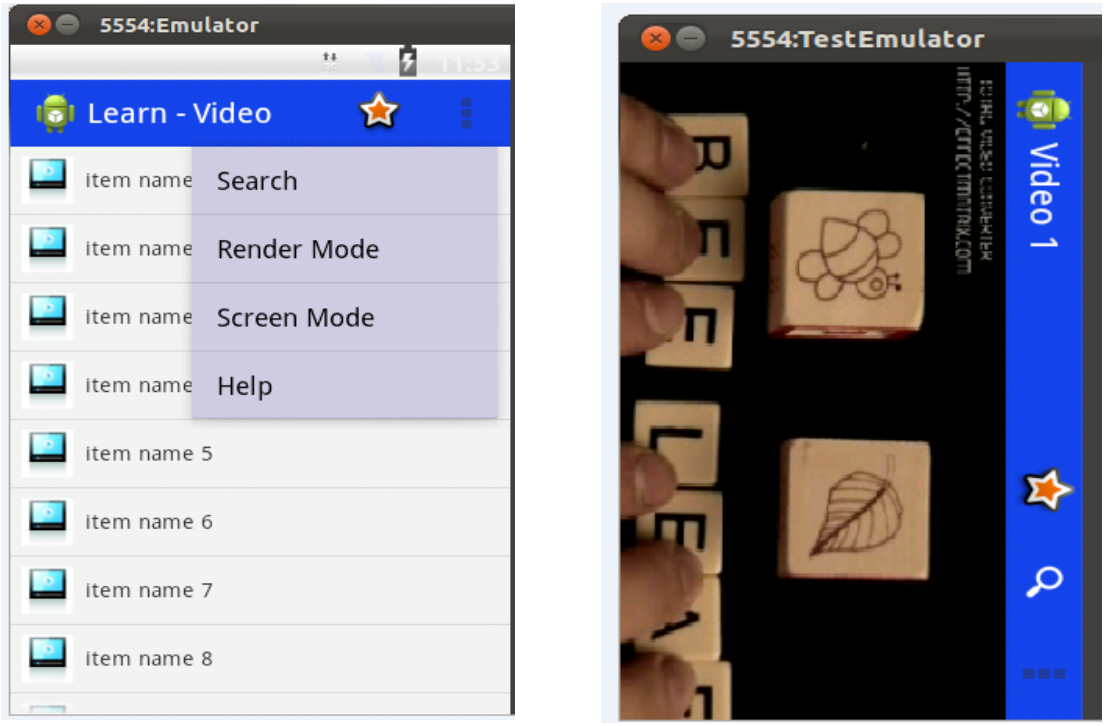
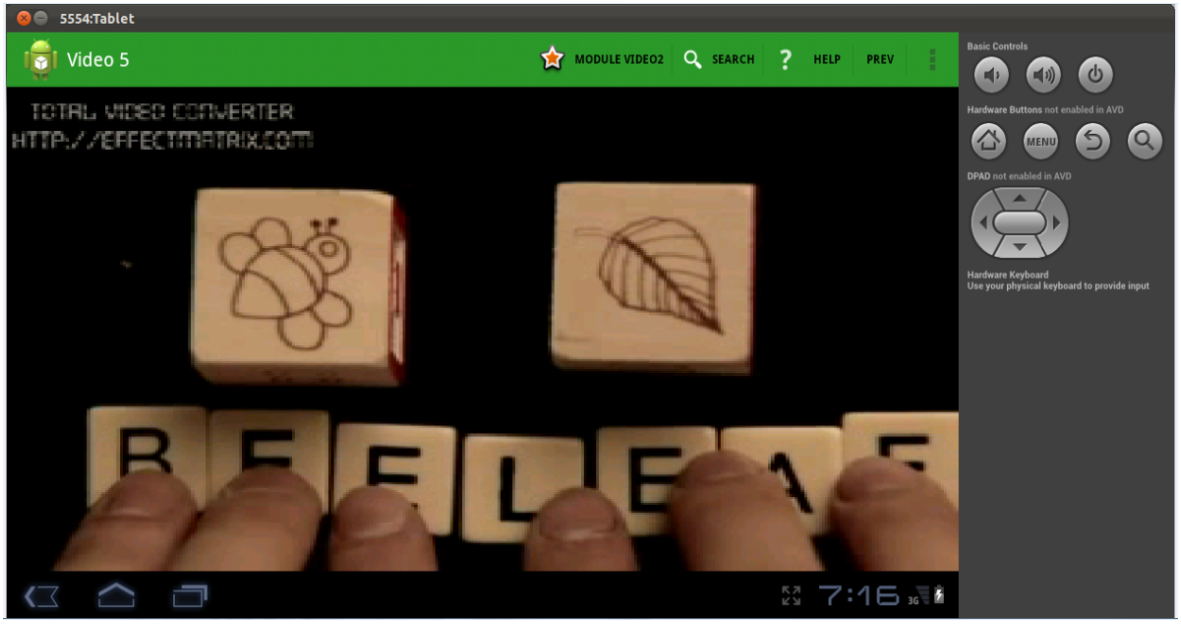


Figure 5.5a. Learn-Video: Listview/Detail View of Video Item(s) on Phone (NL=2, NL=3)



]

Figure 5.5b. Learn-Video: Detail View of Video Item on Tablet (NL=3)

5.2.3.4 Learn–Slide

Fig 5.6a shows the list of Module 1's items and a detail view of the fifth item. The latter is an image item, which portrays the fact that the framework can be used to render presentational materials such as slides. Fig 5.6b shows the equivalent of the second snapshot on a tablet but with a different image item. The sequencing arrows in the second and third snapshots can be used to navigate to the next or previous item in the module in question.

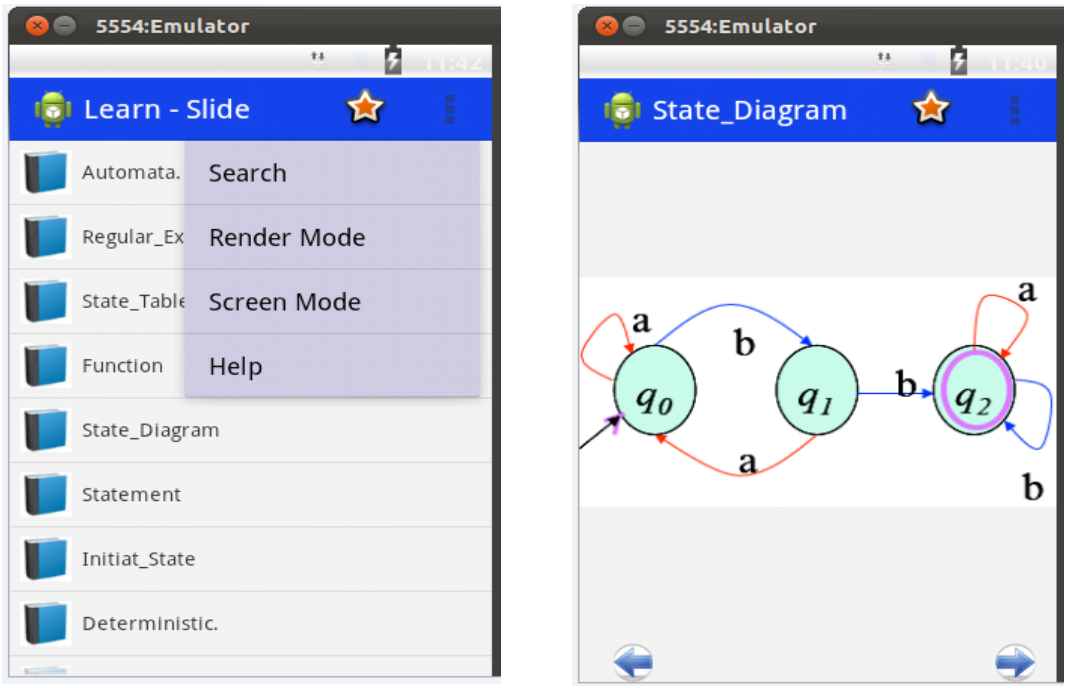


Figure 5.6a. Learn-Slide: List/Detail View of Item(s) on Phone (NL=2, NL=3)

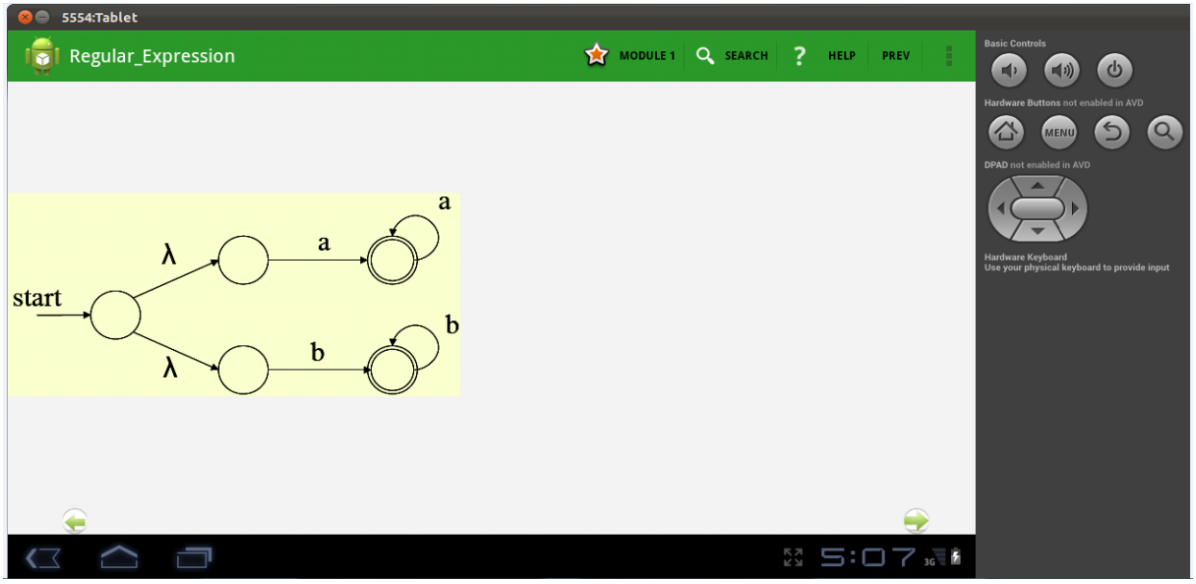


Figure 5.6b. Learn-Slide: Detail View of an Image Item on Tablet (NL=3)

5.2.3.5 Simulate

Fig 5.7a shows the Simulate component when clicked on the HP and a detail view of the first item contained in the first module (whose items are not shown) on an Android phone. The learner can interact with the latter, which portrays a Stack data structure, by pushing and popping the balls, which represent programming variables, into and off the stack respectively. Fig. 5.7b shows the equivalent of the second snapshot on a tablet.

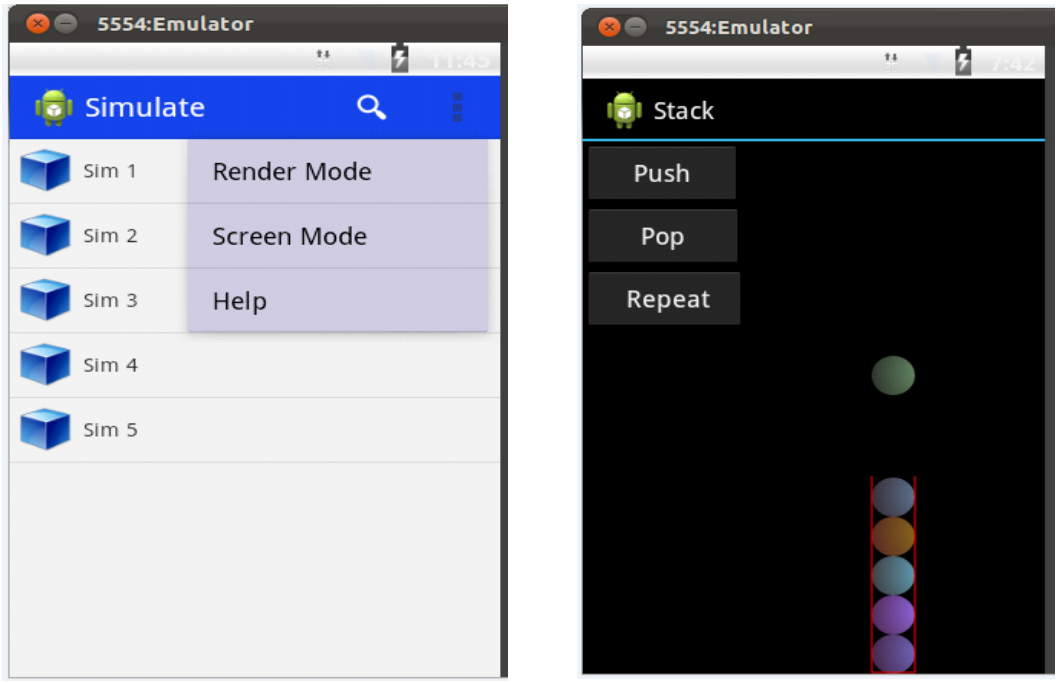


Figure 5.7a. Simulate: Listview of Modules/Detail View of Item on Phone (NL=1, NL=3)

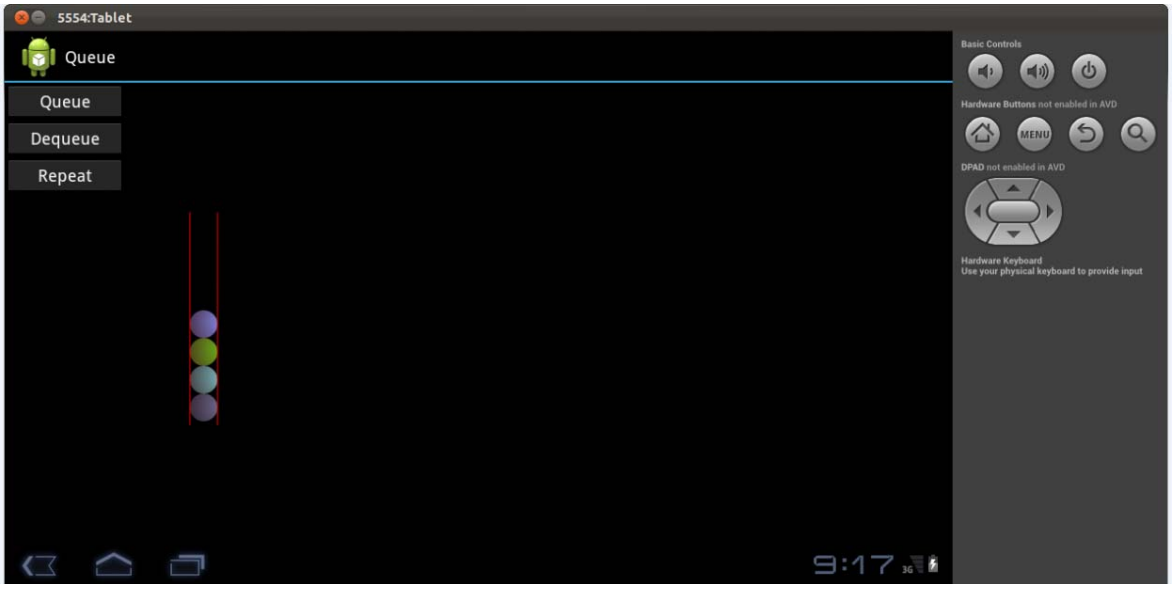


Figure 5.7b. Simulate: Detail View of Simulation Item on Tablet (NL=3)

5.2.3.6 Evaluate

Fig. 5.8 shows all the `activities` during a quiz. The first snapshot presents a snapshot of the Evaluate component when clicked on the HP. The second shows the `QuestionActivity` when the learner selects the first item, while the third the `ScoreActivity` when the learner is done answering all the questions. The fourth screenshot presents the `AnswersActivity` on a tablet. Note the learner can navigate through the questions back and forth as well as return to the first `activity` from the third and fourth by using the left button.

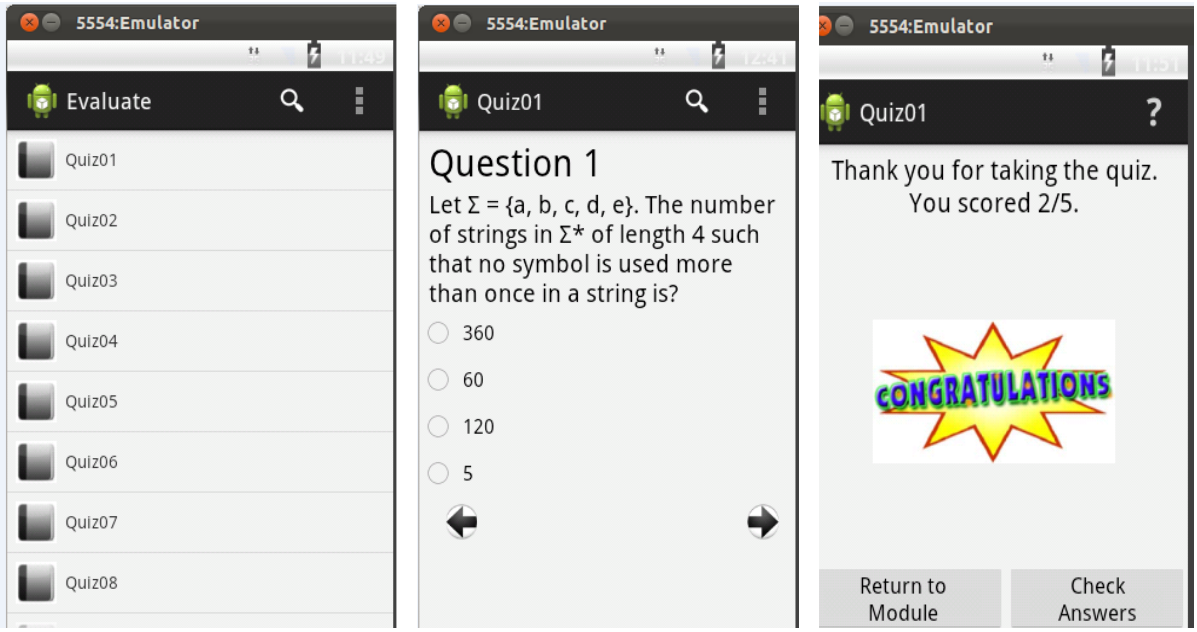


Figure 5.8a. Evaluate: LMFA/QuestionActivity/ScoreActivity on Phone (NL=1, NL=2, NL=3)

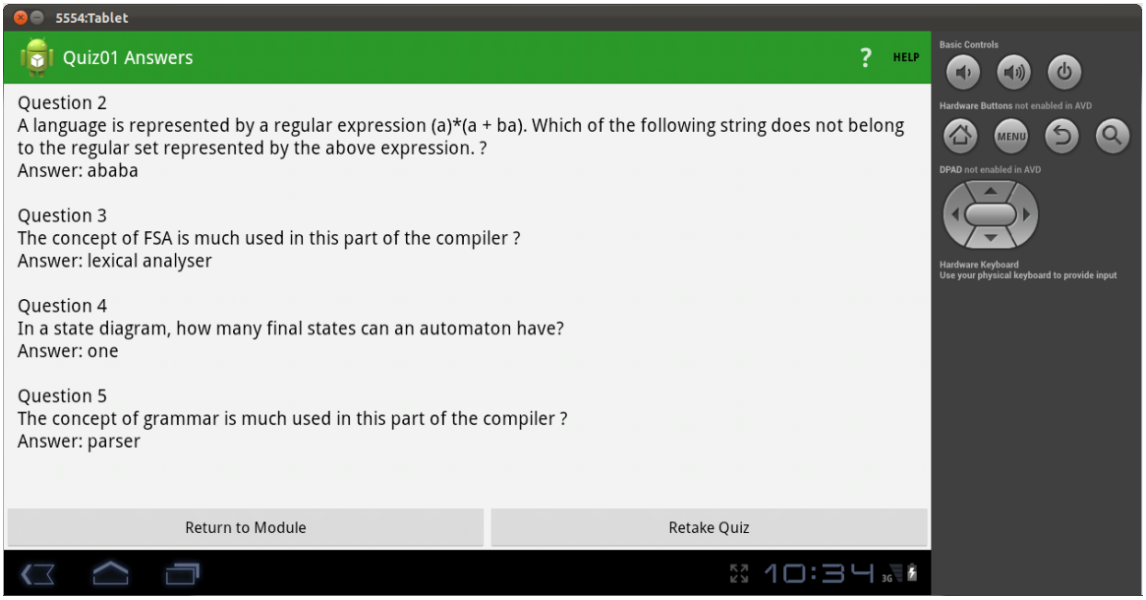


Figure 5.8b. Evaluate: AnswersActivity on Tablet (NL=4)

5.2.3.7 Resources

Fig. 5.9 shows the detail view of the Resources component when it is clicked on the HP. The three screenshots, which are HTML files, are the same and show a list of hyperlinks to online resources. This component allows learners with internet connectivity to access online resources.

Figure 5.9a. Resources: Detail View of Component on Phone (NL=1, NL=1)

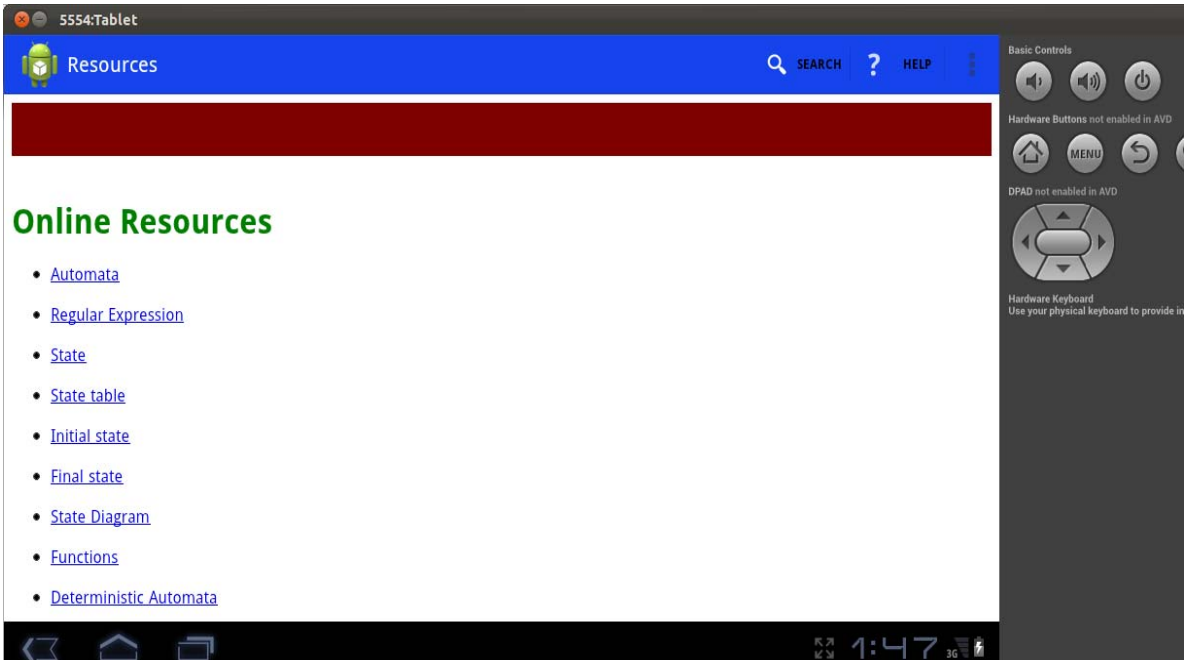


Figure 5.9b. Resources: Detail View of Component on Tablet (NL=1)

5.2.3.8 Help

Fig. 10 shows the Help component at different navigation levels. The first snapshot shows a listview of HTML file items. This opens up when the Help component is clicked on the HP GV or AB. The second and third snapshots show the detail view of the first item in the Help list on a phone and tablet respectively.

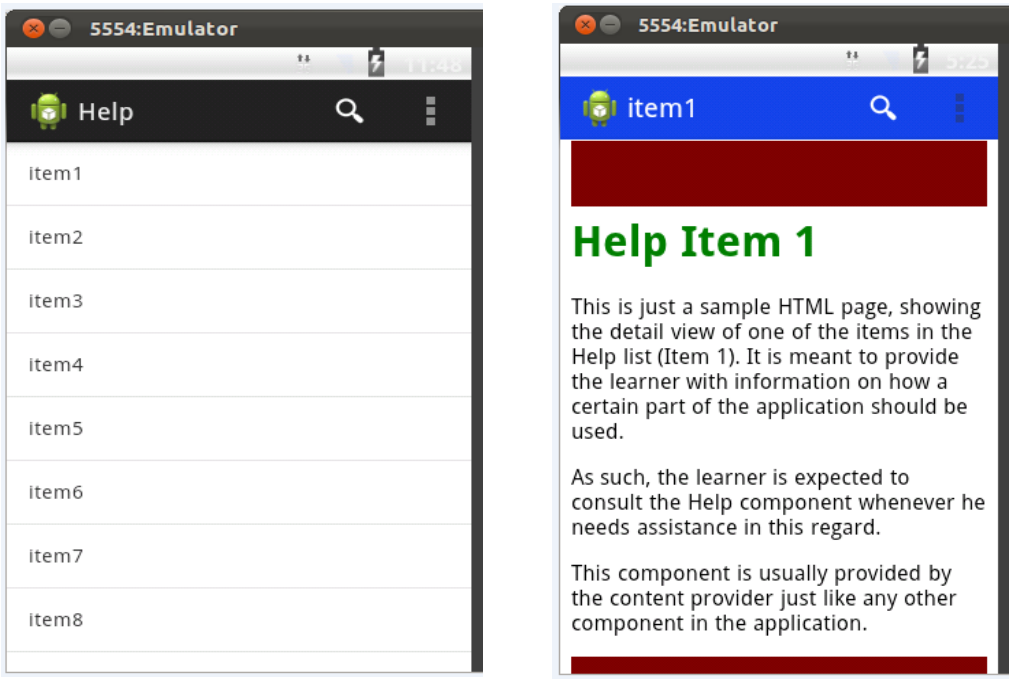


Figure 5.10a. Help: Listview of Help Items and Detail View of Item on Phone (NL=1, NL=2)

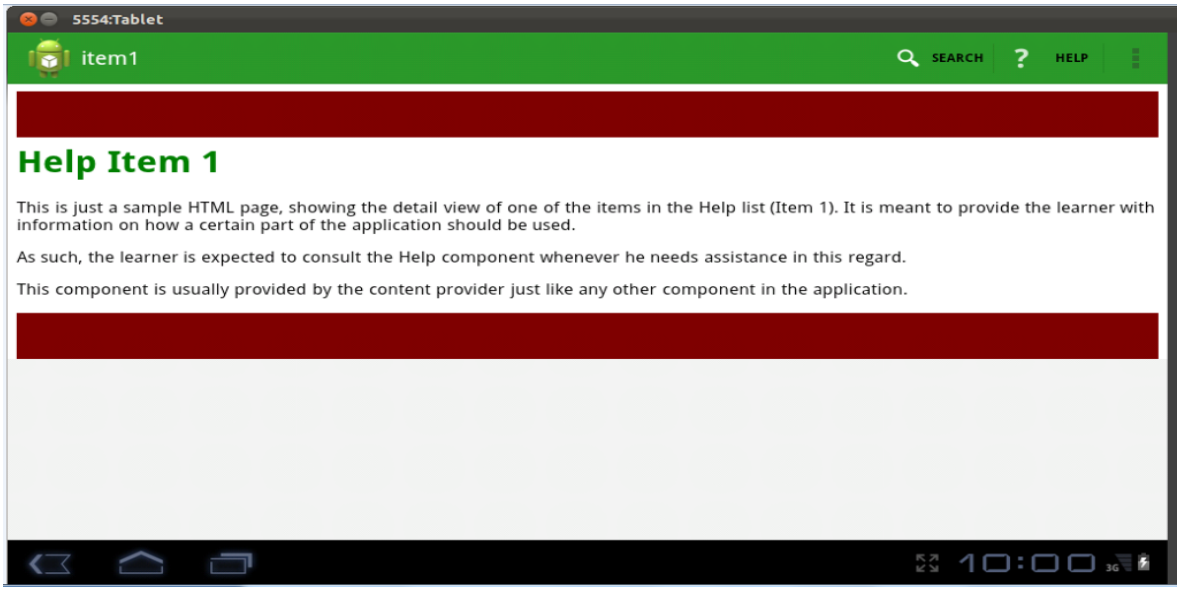


Figure 5.10b. Help: Detail View of Help Item on Tablet (NL=2)

Chapter 6
Conclusion

6.1 Summary of Framework and Key Features

The Native Mobile Multimedia Learning Application (NMMLA) Framework is a framework developed to facilitate the development and deployment of native mobile multimedia learning applications on the Android platform. It was designed using Java, Universal Modelling Language (UML) and the Eclipse Integrated Development Environment (IDE) with the Android Development Tool (ADT) and other required development tools plugged in. The framework can support multimedia learning content such as Hypertext Markup Language (HTML), image, video, audio and simulation. It provides a number of themes, which include Black, Blue and Green. It also offers a wide range of well-selected and pre-defined image resources, which are scaled for different sizes of Android devices with different screen densities such as mdpi, hdpi and xdpi. This provision was made to enable the content provider to quickly and easily create intuitive UIs for the learner in order to enhance his user experience (UX). They include thematic images for components, modules, HTML, image, audio, video, simulation, quiz and sequencing items. Other provided images and icons include those for Course menu, Theme menu, About, Help and Search. They are pre-defined in the `AppController` class as `final` and `static` constants by using intuitive names that suggest what they represent. The content provider can leverage any of these in defining the look and feel of his entire application. Moreover, by virtue of Android's `FragmentActivity` class and the Actionbar Sherlock Library [20, 21], leveraged in developing the library, the framework can support both phones and tablets alike, ranging from API 8 (Android 2.2) to API 17 (Android 4.2) platforms, as evident in the screenshots in the previous chapter. Table 5.2 shows some of these provisions (features) and their location within the framework. Finally, the learner can either pre-evaluate himself on the homepage or post-evaluate himself within the application after finishing taking a module. The content provider, at the setup of the application, usually determines (presets) the number of questions to be administered to the learner during each quiz session. These questions are randomly drawn from an SQLite database stored in the `asset` folder within the Integrated Development Environment.

6.2 Summary of Work and Results

With the results (graphical user interfaces and explanations) presented in the preceding chapter, we can conclude that we have been able to provide a significant answer to the research question posed in Chapter 1: "*How can African students with different learning preferences learn anywhere and anytime without the cost or lack of internet connectivity being a barrier?*" With our framework, Native Mobile Multimedia Learning Application Framework on Android Platform, implemented as a library, educators and content providers in HEI's and training/learning organizations can now easily and quickly deploy their multimedia learning

content to the Android mobile platform, which is becoming more and more ubiquitous each day. This will prevent them from reinventing the wheel and reduce time to market. The benefits of this research are summarized as follows:

- The proposed systematic Content Flow Algorithm Tree will promote and facilitate the development and deployment of interactive and multimedia learning content in dual views (list and tab) on any subject or course of study both on the Android and other platforms, as it provides a blueprint which developers can leverage.
- The software product of our research and implementation will help students, especially distance learners with different learning preferences on the African continent, where internet connectivity and bandwidth still pose a great challenge, to learn outside the classroom as their courses can now be deployed to their Android devices.
- It will also help employees who work and study at the same time to be able to study their course work while on the move or at work at their leisure time.

However, educational and training content developers in localized contexts should be prepared to develop and provide the right multimedia content which will make learning on mobile devices without internet connectivity a great experience, as the research community comes up with new and better ways of delivering content to learners.

6.3 Contribution

The contribution made by this thesis can be summarised as follows:

- Provision of a systematic blueprint known as Content Flow Algorithm Tree for the design and implementation of a native mobile multimedia learning application, which supports the delivery of rich interactive and modular multimedia learning contents, such as HTML, images, audio, video and simulations, in dual views (list and tab views), as well as multiple courses and customizable themes.
- Showing how a framework application can be designed and implemented as a library on a ubiquitous platform such as Android by using software engineering design principles and development methodology within an IDE like Eclipse equipped with the relevant development tools such as ADT.
- Providing more understanding of the area of mobile multimedia learning application development and a basis for future improved mobile learning frameworks and systems that integrate native and web-based platforms.

6.4 Challenges

During the development of the framework, many a challenge was faced. However, one stood out. In the course of test-running the framework, having the content-provider application, hosted in a different Android project other than the framework (library) project, to see the `activity` declarations in the framework's Android manifest was a great challenge. However, this problem was worked around by duplicating or porting the `activity` declarations in the framework's manifest to the content-provider project's Android manifest for the time being. Consequently, the application started running well (without errors) as expected. But then, we found out

thereafter in the Android online documentation that this move was actually necessary, as the repeatedly declared framework's `activities` in the content-provider project's Android manifest will eventually synchronize with the declared `activities` in the library's manifest as the program runs.

6.5 Future Work

In future work, we intend, first, to improve the rendering of listview items on a tablet as well as when they are selected. Currently, the list of items and detail view of each item are rendered separately on different screens. However, we hope to utilize the master/detail format offered by Android such that the master list of items displays on the left pane while the detail view of each item on the right of the tablet. Second, as the situation of internet connectivity improves across the African continent, we hope to integrate and extend our framework to support multimedia content that can be streamed and downloaded from the Internet and cached locally in the Android file system. This will enable learners to have internet access to frequently updated learning content on one hand and local access when there is no internet connectivity. As such, we will be looking at how our framework can support web-based technologies such as HTML5, CSS and JavaScript. We also hope to extend our framework to provide much more robust and collaborative mobile learning environment, where learners can participate as active constructors of knowledge and teachers as facilitators of the learning process.

References

[1] J. Traxler, "Defining, discussing, and evaluating mobile learning: The moving finger writes and having writ….," International Review of Research in Open and Distance Learning , vol. 8, no. 2, 2007, pp. 1-12.

[2] Upside Learning, "Mobile learning - a quick start guide." Retrieved April 5, 2013 from http://www.upsidelearning.com

[3] A. Heiphetz, "How Mobile Technology Can Enhance Student Learning and Workforce Training," 2011. Retrieved from http://www.mcgraw-hillresearchfoundation.org

[4] M. Shanmugapriya and A. Tamilarasia, "Designing an m-learning application for ubiquitous learning environment in the Android based mobile devices using web services, Indian Journal of Computer Science and Engineering (IJCSE) , 2011, pp. 22-30.

[5] D. S. Metcalf II and J. M. De Marco, "mLearning: Mobile learning and Performance in the Palm of Your Hand," HRD Press, Inc., 2006.

[6] Android Developers, Android Online Documentation. Retrieved March 28, 2013 from http://developer.android.com

[7] K. Oyibo and M. Hamada, "A Framework for Instantiating Native Mobile Multimedia Learning Applications on Android Platform," unpublished.

[8] TechTarget, "Introducing Android." Retrieved April 8, 2013 from http://media.techtarget.com/searchMobileComputing/downloads/Introducing_Android.pdf

[9] R. E. Mayer, "Multimedia learning: are we asking the right questions?" Educational Psychologist, vol. 32, no. 1, pp. 1-19, 1997.

[10] ADL M-learning Guide. Retrieved March 28, 2013 from http://mlearn.adlnet.mobi/

[11] J. Wharton. "Actionbar Sherlock Library." Retrieved January 3, 2013 from http://actionbarsherlock.com

[12] T. L. Friedman, "The world is flat: a brief history of the twenty-first century," April 5, 2005.

[13] R. Kurzweil, Futurist, Handheld Learning '09. The Economic Times and GSM Association, 2009. Retrieved April 5, 2013 from http://articles.economictimes.indiatimes.com/2012-02-27/news/31104598_1_mobile-operators-number-of-mobile-connections-gsma)

[14] Voxxi, "4Afrika looks to cash in on exploding smartphone market in Africa." Retrieved March 28, 2015 from http://www.voxxi.com/4afrika-smartphone-market-in-africa/#ixzz2Omjxjv9g

[15] B. Coker, "Nigeria ahead smartphones penetration in Africa," Business Day, Tuesday, 27 November 2013. Retrieved March 28, 2013 from http://www.businessdayonline.com/NG/index.php/markets/companies-and-market/48068-nigeria-ahead-smartphones-penetration-in-african

[16] D. Okeytola, "25% of Nigerian mobile subscribers use smartphones–TNS," Punch, January 8, 2013. Retrieved March 28, 2013 from http://www.punchng.com/business/technology/25-of-nigerian-mobile-subscribers-use-smartphones-tns/

[17] P. B. Muyinda, J. T. Lubega, and K. Lynch, "Mobile learning objects deployment and utilization in developing countries," International Journal of Computing and ICT Research, Special Issue vol. 4, no. 1, pp. 37 - 46.

[18] D. Ayanda, S. Eludiora, , D. Amassoma and M. Ashiru. "Towards a model of e-Learning in Nigerian higher Institutions: an evolutionary software modeling approach," Information and Knowledge Management, vol 1, no. 1, 2011, pp. 1-10.

[19] J. Chimombo, "Issues in basic education in developing countries: an exploration of policy options for improved delivery," CICE Hiroshima University, Journal of International Cooperation in Education, vol. 8, no. 1, 2005 129-152.

[20] Smashing Magazine, "Designing for Android," Smashing Media GmbH, Fresburg, Germany, September 2012.

[21] Samsung Developers. "Handling multiple screen size in Android." Retrieved April 27, 2013 http://developer.samsung.com/android/technical-docs/Handling-Multiple-Screen-Size-in-Android

[22] R. E. Mayer, ""Cognitive theory of multimedia learning," In R.E. Mayer (Ed.), The Cambridge Handbook of Multimedia Learning, New York: Cambridge University Press, 2005.

[23] A. Pocatilu, and A. Pocovnicu, "Multimedia applications and technologies for m-Learning. Economy Informatics," vol. 9, no. 1, 2009, p. 64.

[24] P. Doolittle, A. McNeill, K. Terry and S. Scheer, "Multimedia, cognitive load and pedagogy," Interactive Multimedia in Education and Training, pp. 184-212, Idea Group Publishing, USA, 2005.

[25] W. L. Leite, M. Svinicki and Y. Shi, "Attempted validation of the scores of the VARK: learning styles inventory with multitrait-multimethod confirmatory factor analysis models," p 2, SAGE Publications, 2009.

[26] T. F. Hawk and A. J. Shah, "Using Learning Style Instruments to Enhance Student Learning," Decision Sciences Journal of Innovative Education, 2007.

[27] LdPride. "What are learning styles?" Retrieved October 17, 2012 from http://www.ldpride.net/learningstyles.MI.htm

[28] Wikipedia. Retrieved April 12, 2013 from http://wikipedia.org

[29] D. Kolb, "Experiential learning: experience as the source of learning and development," Englewood Cliffs, NJ: Prentice-Hall, 1984.

[30] M. Smith, "David A. Kolb on experiential learning," 2001. Retrieved April 17, 2013, from http://www.infed.org/biblio/b-explrn.htm

[31] Sun Microsystem, "Model view controller." Retrieved April 8, 2013 from http://java.sun.com/blueprints/patterns/MVC-detailed.html

[32] EmirWeb, "Model view controller," Android Workshop, University of Toronto. Retrieved May 1, 2013 from http://www.emirweb.com/AndroidTutorial.php

[33] PriceCheck, "Cellular Phones." Retrieved March 29, 2013, from http://www.pricecheck.co.za/categories/f/126/Cellular+Phones/f~f/62~269/Samsung~Android/0/tp.weekly/DESC/

[34] S. Dowuona, "High bandwidth cost, unfriendly regulations hindering broadband growth in Africa." Retrieved March 28, 2013 from http://business.myjoyonline.com/pages/news/201206/88283.php

[35] H. F. Hanaf, K. Samsudin, "Mobile Learning Environment System (MLES): the case of Android-based learning application on undergraduates' learning," International Journal of Advanced Computer Science and Applications, vol. 3, no. 3, 2012.

[36] Gartner, "Gartner says worldwide mobile phone sales declined 1.7 percent in 2012." Retrieved from March 28, 2013 from http://www.gartner.com/newsroom/id/2335616

[37] G. Kurubacak, "Identify research priorities and needs for mobile learning technologies in open and distance education: a Delphi study," College of Open Education, Anadolu University, Anadolu, 2007.

[38] J. Pettit, and A. Kukulska-Hulme, "Going with the grain: mobile devices in practice," Australasian Journal of Educational Technology, vol. 23, no. 1, 2007, pp. 17-33.

[39] F. Motiwalla, "Mobile learning: A framework and evaluation," In Computers & Education vol. 49 no. 3, pp. 581-596, 2007.

[40] M. Sharples, J. Taylor, and G. Vavoula, "Towards a theory of mobile learning," 4th World Conference on M-learning, 2005, Cape Town, SA.

[41] D. Keegan, "The incorporation of mobile learning into mainstream education and training," 4th World Conference on M-learning, 2005, Cape Town, SA.

[42] M. Fayad, and D. Schmidt, "Object-oriented application frameworks. Special issue on object-oriented application frameworks," vol. 40, no. 10, October 1997. Retrieved April 8, 2013 from http://www1.cse.wustl.edu/~schmidt/CACM-frameworks.html

[43] DocForge, "Framework," Software development resources." Retrieved March 28, 2013 from http://docforge.com/wiki/Framework

[44] UPRM, "Design Patterns." Retrieved April 8, 2013 from http://www.ece.uprm.edu/~borges/patterns.pdf

[45] PoinCare, "Model view controller (MVC) architecture." Retrieved April 8, 2013 from http://poincare.matf.bg.ac.rs/~andjelkaz/pzv/cas4/mvc.pdf

[46] Rhodes. Retrieved May 2, 2013 from http://www.motorolasolutions.com/US-EN/RhoMobile+Suite/Rhodes

[47] OpenMobile, "Open source framework for mobile application development." Retrieved April 8, 2013 from http://www.openmobileis.org/openmis_about.html

[48] PhoneGap. Retrieved April 8, 2013 from http://phonegap.com/

[49] D. Parsons, H. Ryu and M. Cranshaw, "A design requirements framework for mobile learning environments," Journal of Computers, vol. 2, no. 4, June 2007.

[50] A. Mostakhdemin-Hosseini and J. Tuimala, "Mobile learning framework," IADIS International Conference Mobile Learning, 2005, p. 205.

[51] T. Leacock and J. Nesbit, "A framework for evaluating the quality of multimedia learning resources," Educational Technology & Society, vol. 10 no.2, 2007, pp. 44-59.

[52] C. Diezmann and J. Watters, "A theoretical framework for multimedia resources: a case from science education," In Proceedings Australian Association for Research in Education Conference, Brisbane, 2002.

[53] G. Fenstermacher, "Philosophy of research on teaching: three aspects," In M. C. Wittrock (Ed.), Handbook of Research on Teaching, New York: Macmillan, 1986, pp. 37-49.

[54] Moodle. Retrieved April 8, 2013 from http://docs.moodle.org

[55] J. Leyva, "UMM: unofficial Moodle mobile app." Retrieved April 8, 2013 from https://moodle.org/plugins/view.php?id=175

[56] Blackboard. Retrieved April 28, 2013 from https://www.blackboard.com

[57] B. Boehm, "A Spiral model of software development and enhancement. ACM Sigsoft Software Engineering Notes", "ACM", vol. 11 no.4, pp. 14-24, August 1986.

[58] L. Osterweil, "A process programmer looks at the spiral model: a tribute to the deep insights of Barry W. Boehm," International Journal of Software and Informatics, vol. 5, no. 3, 2011, pp. 457. Retrieved April 15, 2013 from http://www.ijsi.org.

[59] NASA, "NASA software safety guidebook," NASA-GB-8719.13, p.56-57, March 31, 2004.

[60] Berkeley, "Model-view-controller: a design pattern for software," 2004. Retrieved April 8, 2013 from http://ist.berkeley.edu/as-ag/pub/pdf/mvc-seminar.pdf

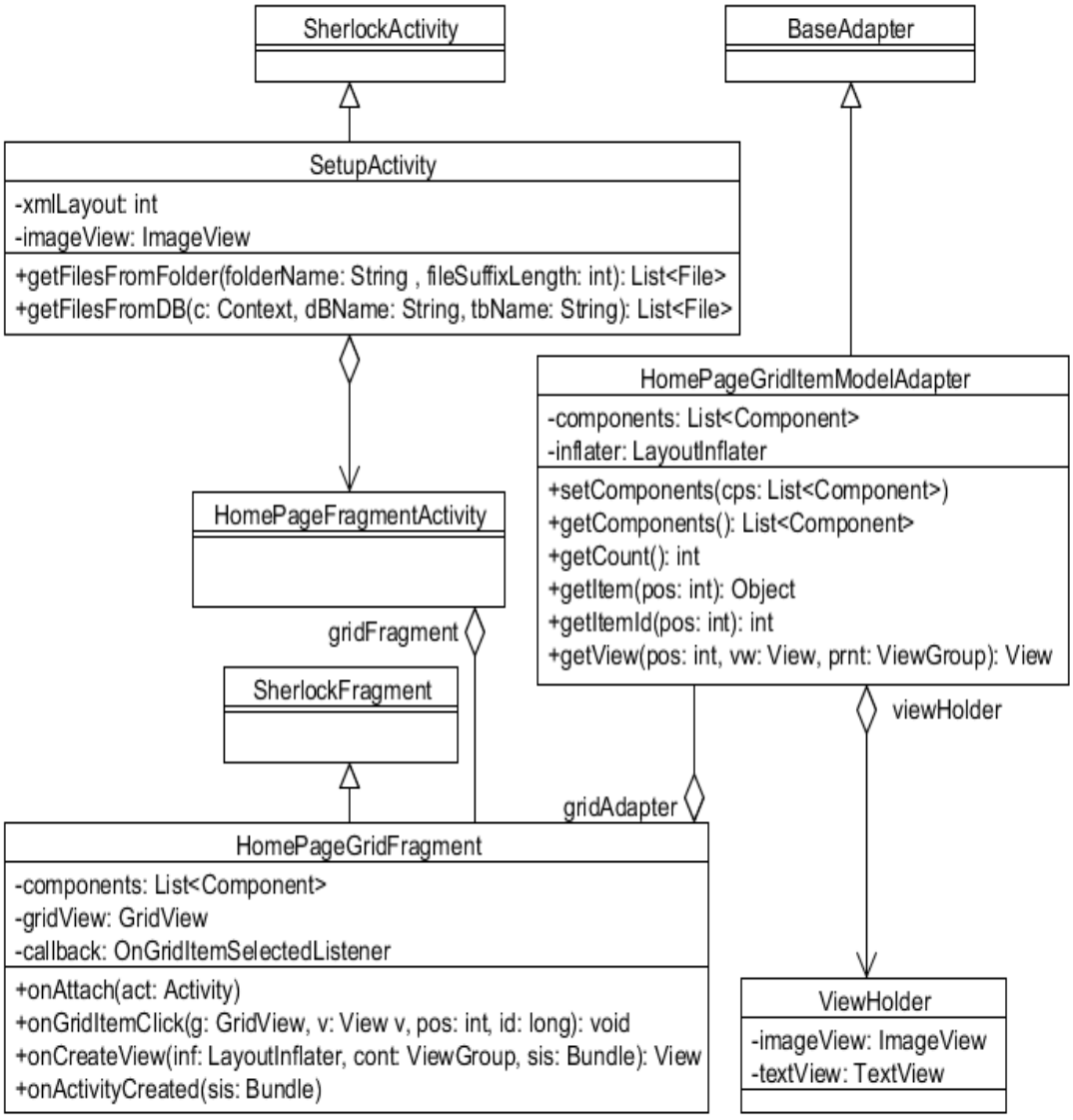
[61] Eclipse. Retrieved March 20, 2013 from http://www.eclipse.org/downloads/moreinfo/jee.php

[62] Android Asset Studio. Retrieved March 20, 2013 from http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html

[63] J. Gilfelt, Android Action Bar Style Generator. Retrieved March 20, 2013 from http://jgilfelt.github.io/android-actionbarstylegenerator/

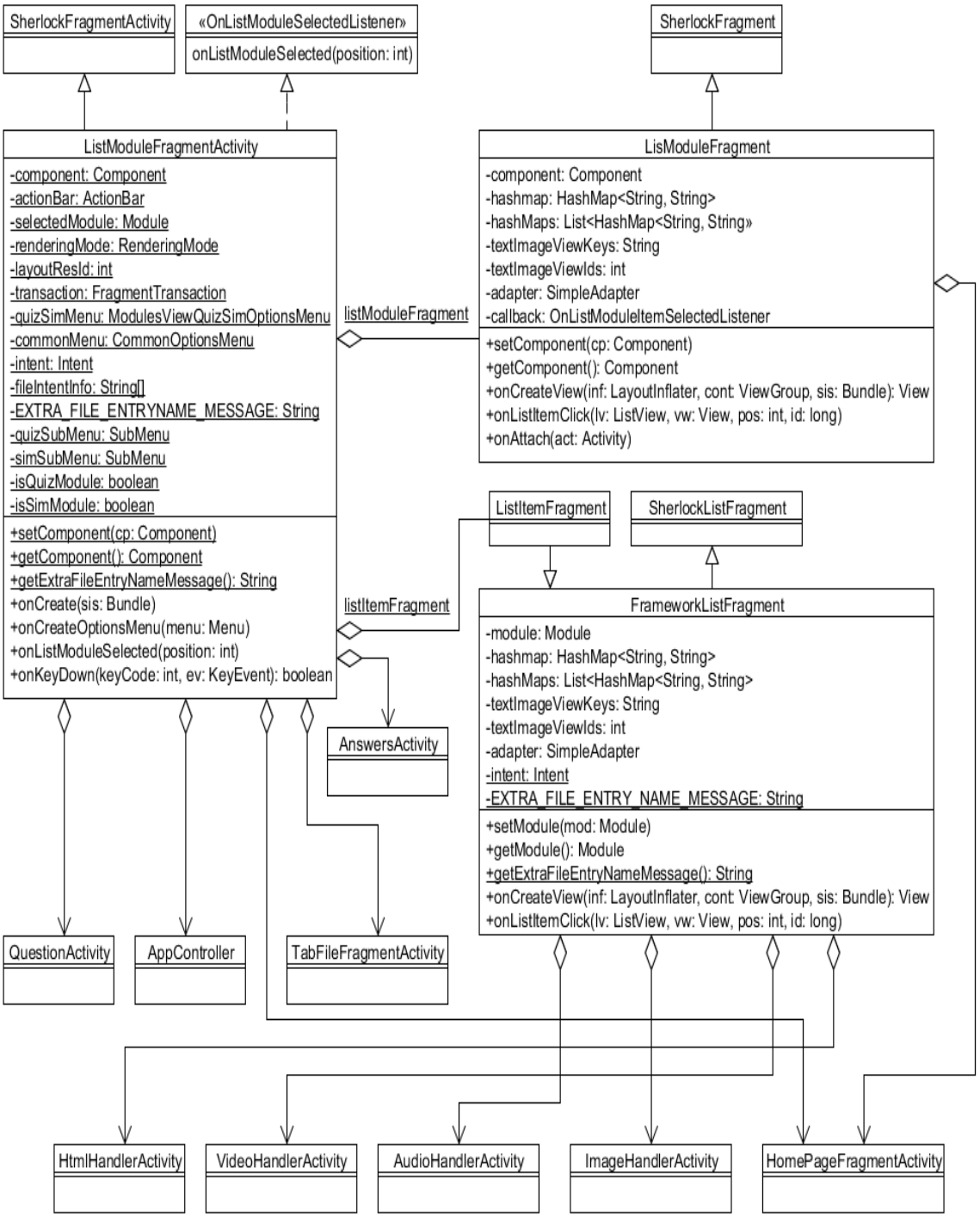
.

Appendix A1
SetupActivity and Related Class Diagrams

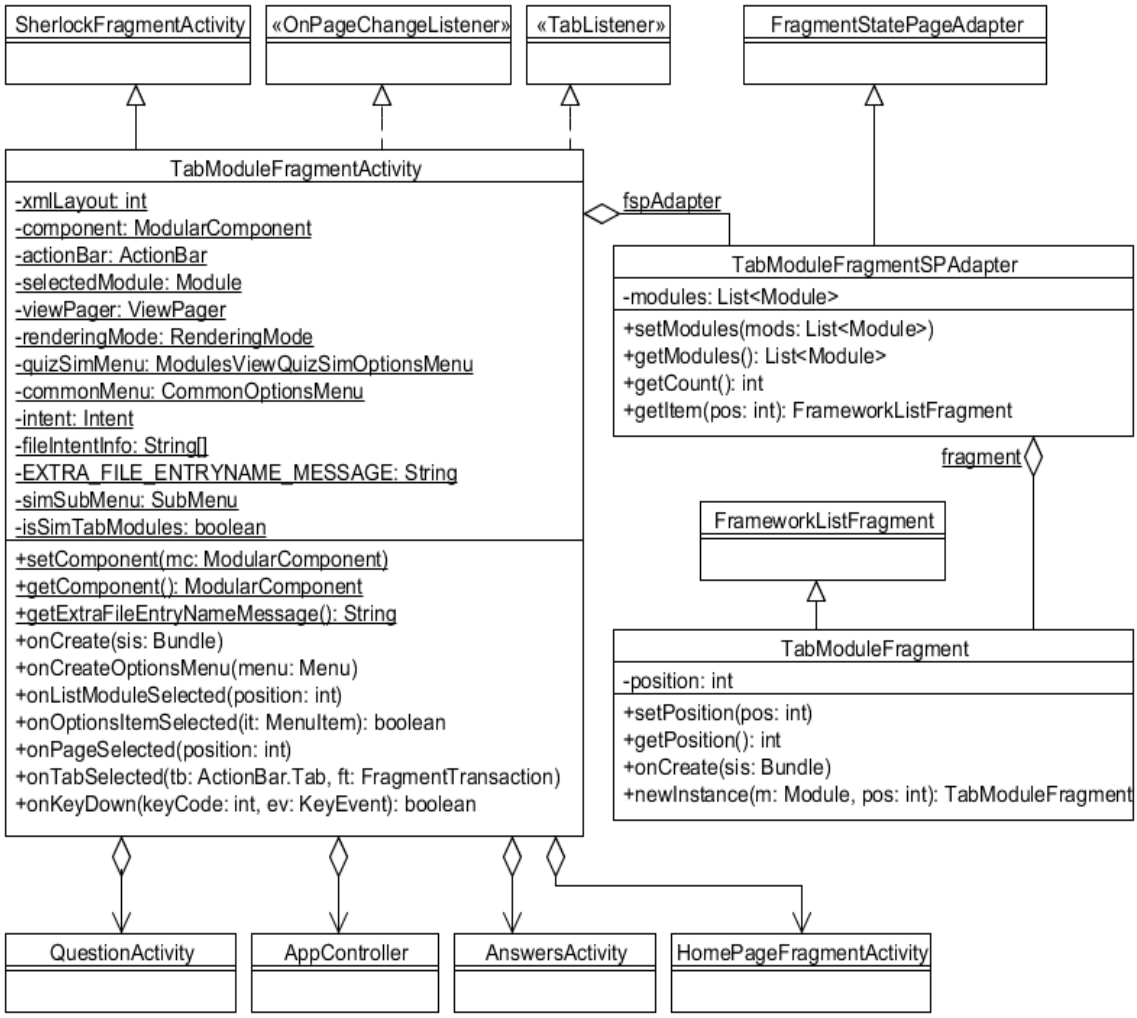


Note: The attributes and methods of HomePageFragmentActivity are shown alongside the Framework Data Model.

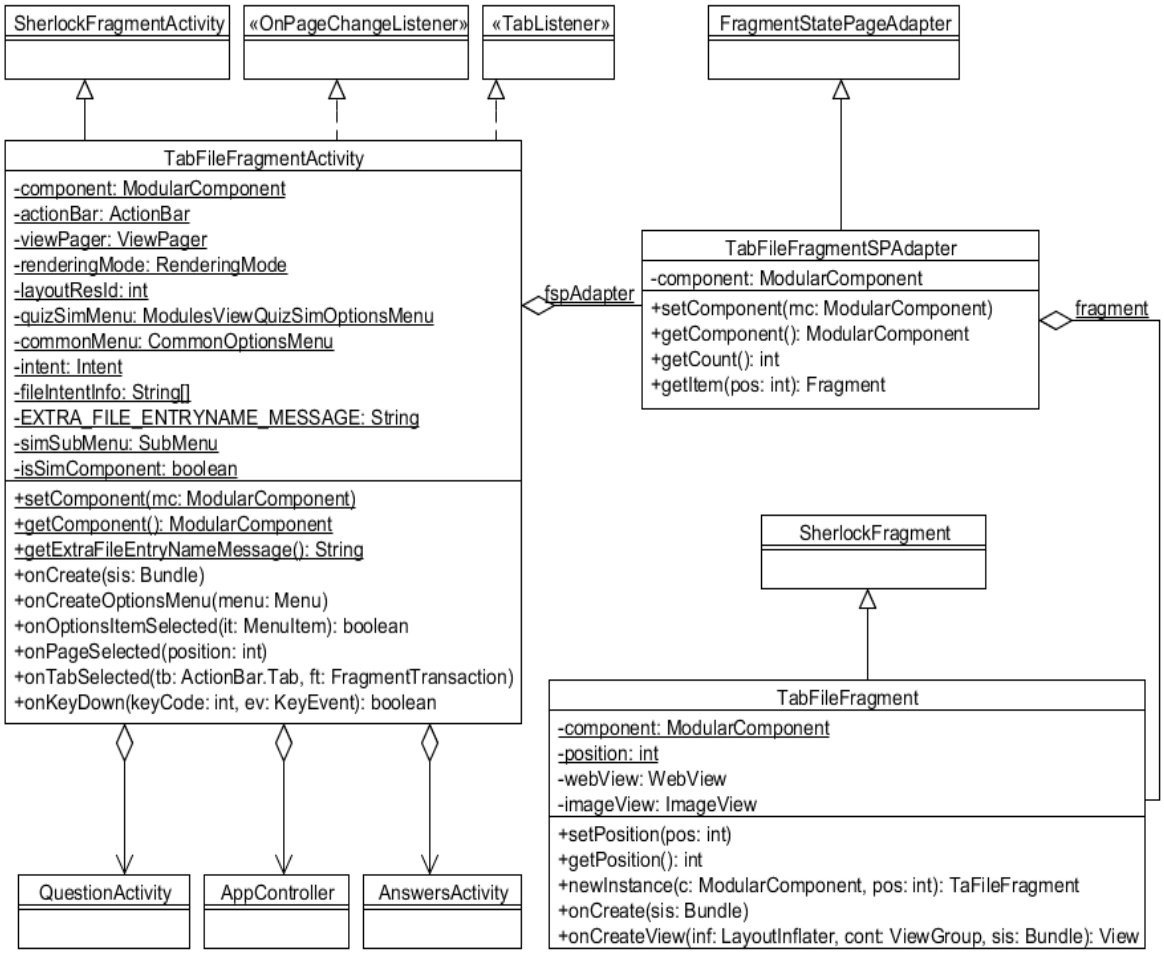
Appendix A2
ListModuleFragmentActivity and Related Class Diagrams



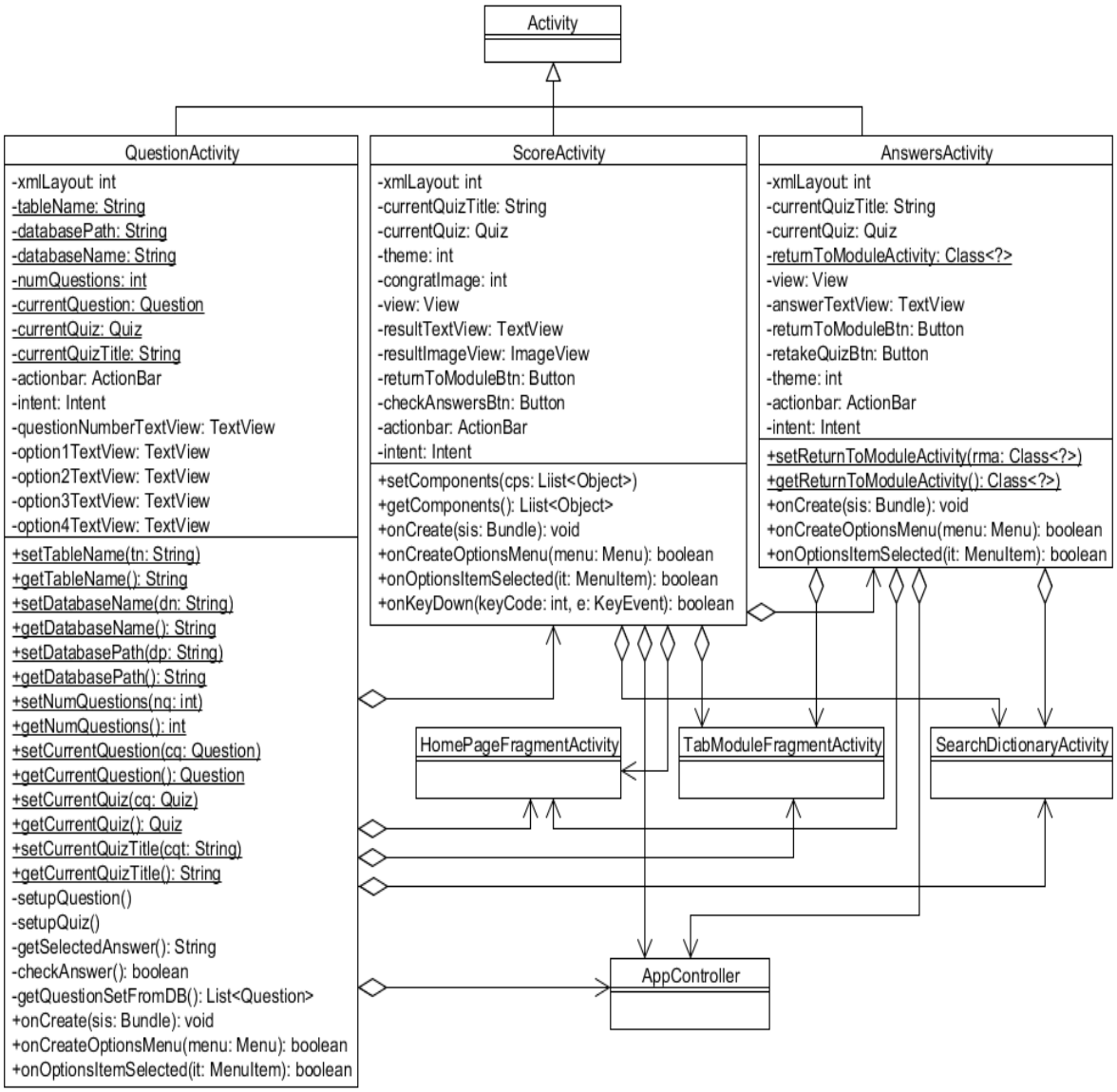
Appendix A3
TabModuleFragmentActivity and Related Class Diagrams



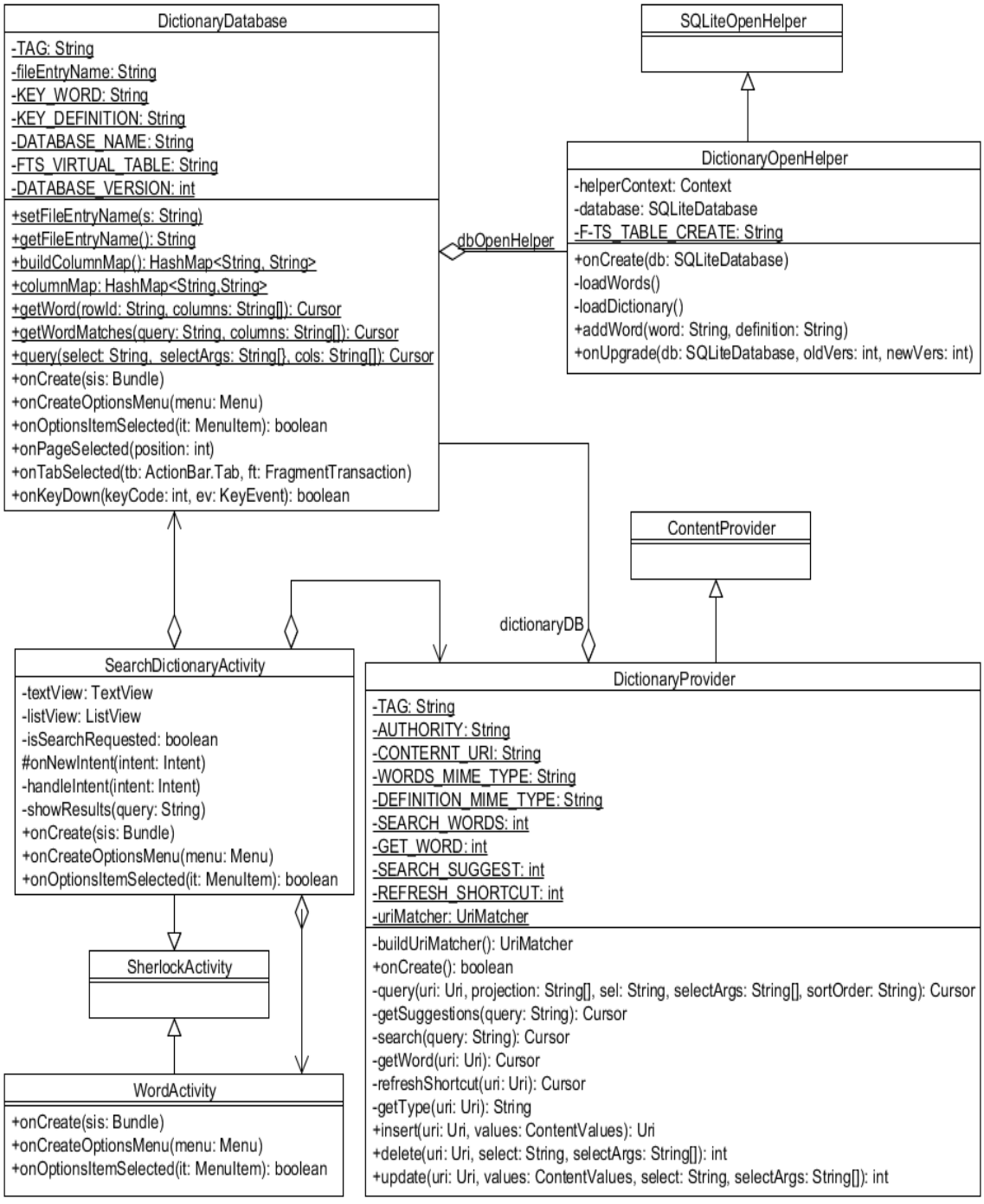
Appendix A4
TabFileFragmentActivity and Related Class Diagrams



Appendix A5
Evaluation Activities and Related Class Diagrams

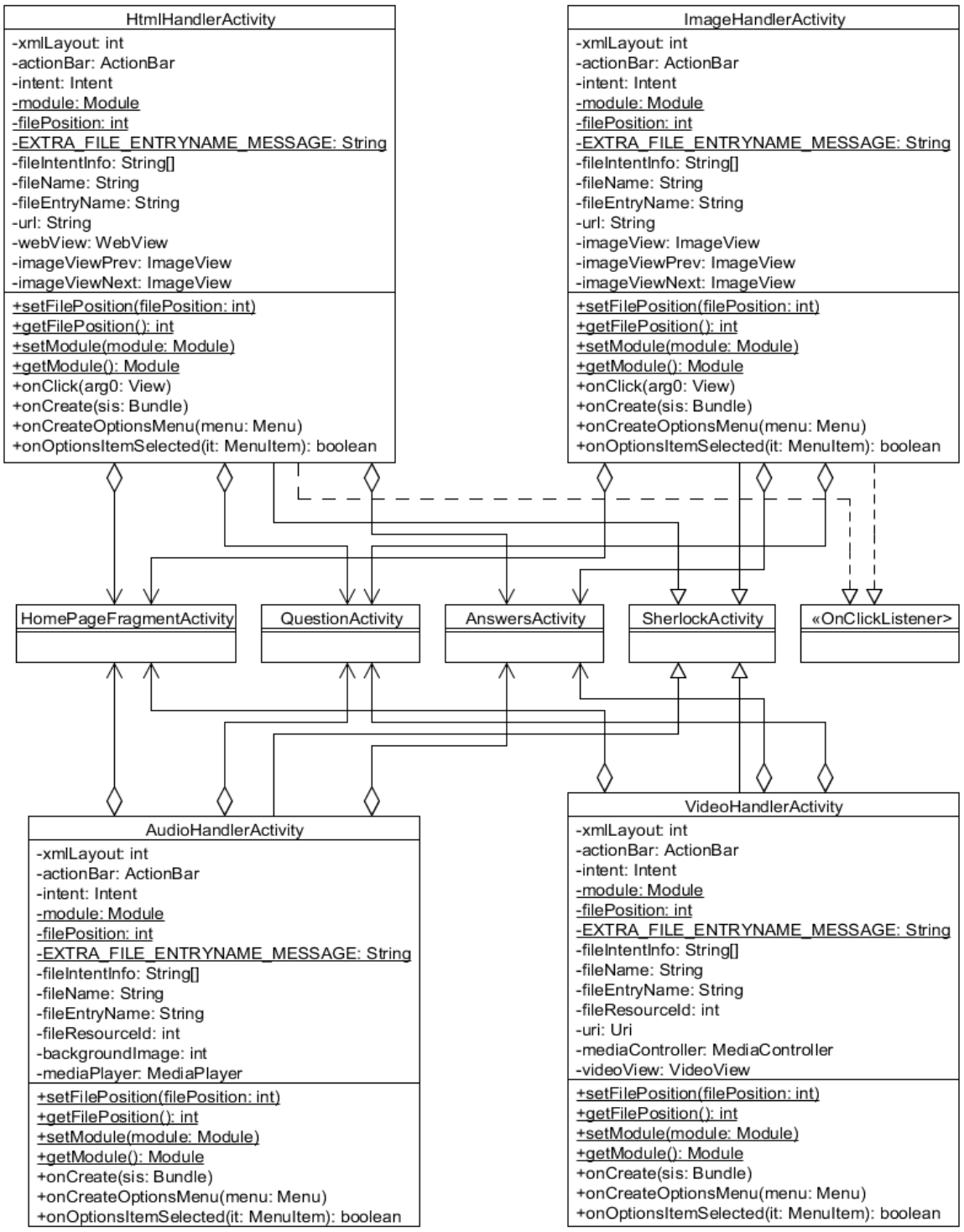


Appendix A6
Search Activities and Related Class Diagrams

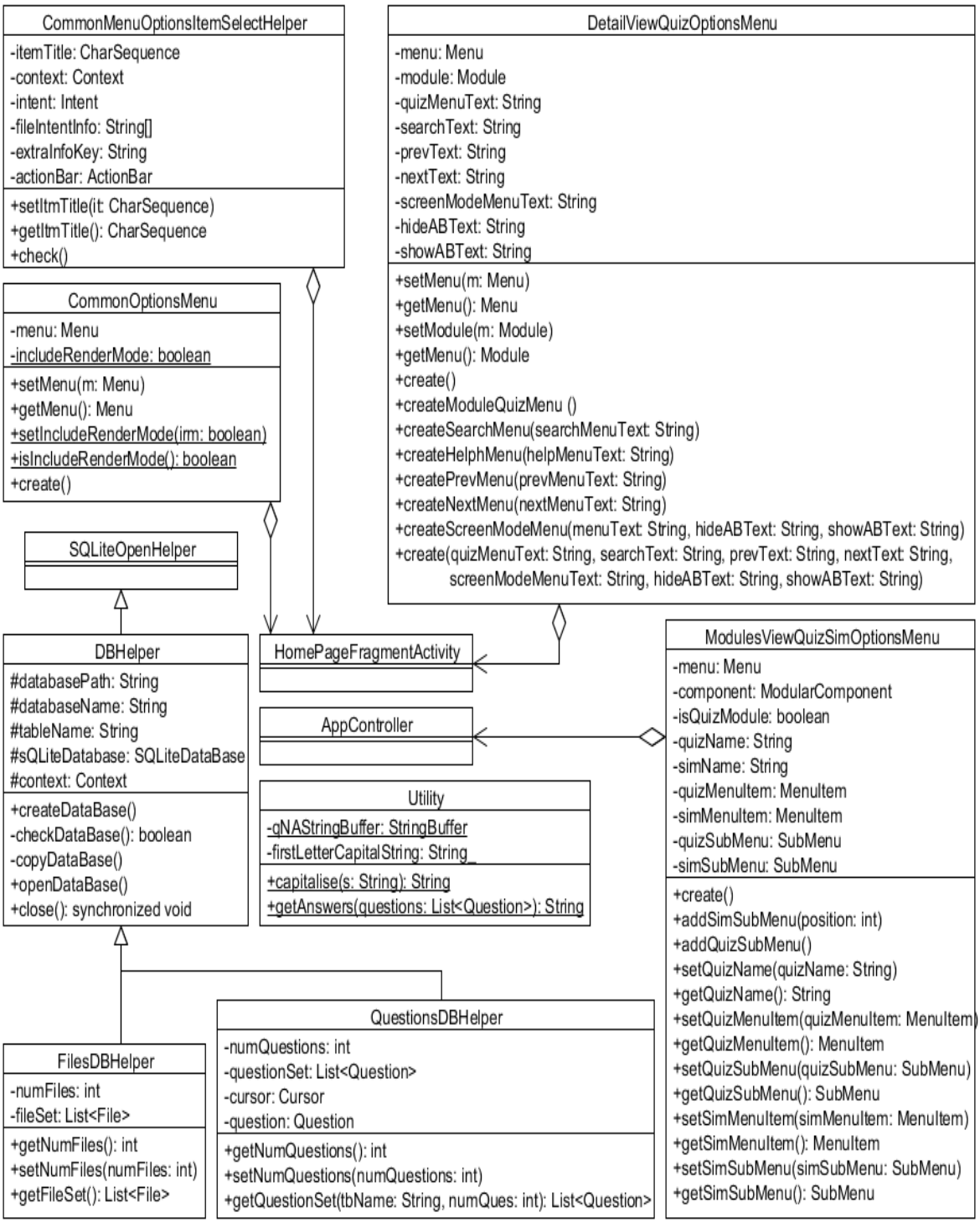


Adapted from Android Online Documentation [6]

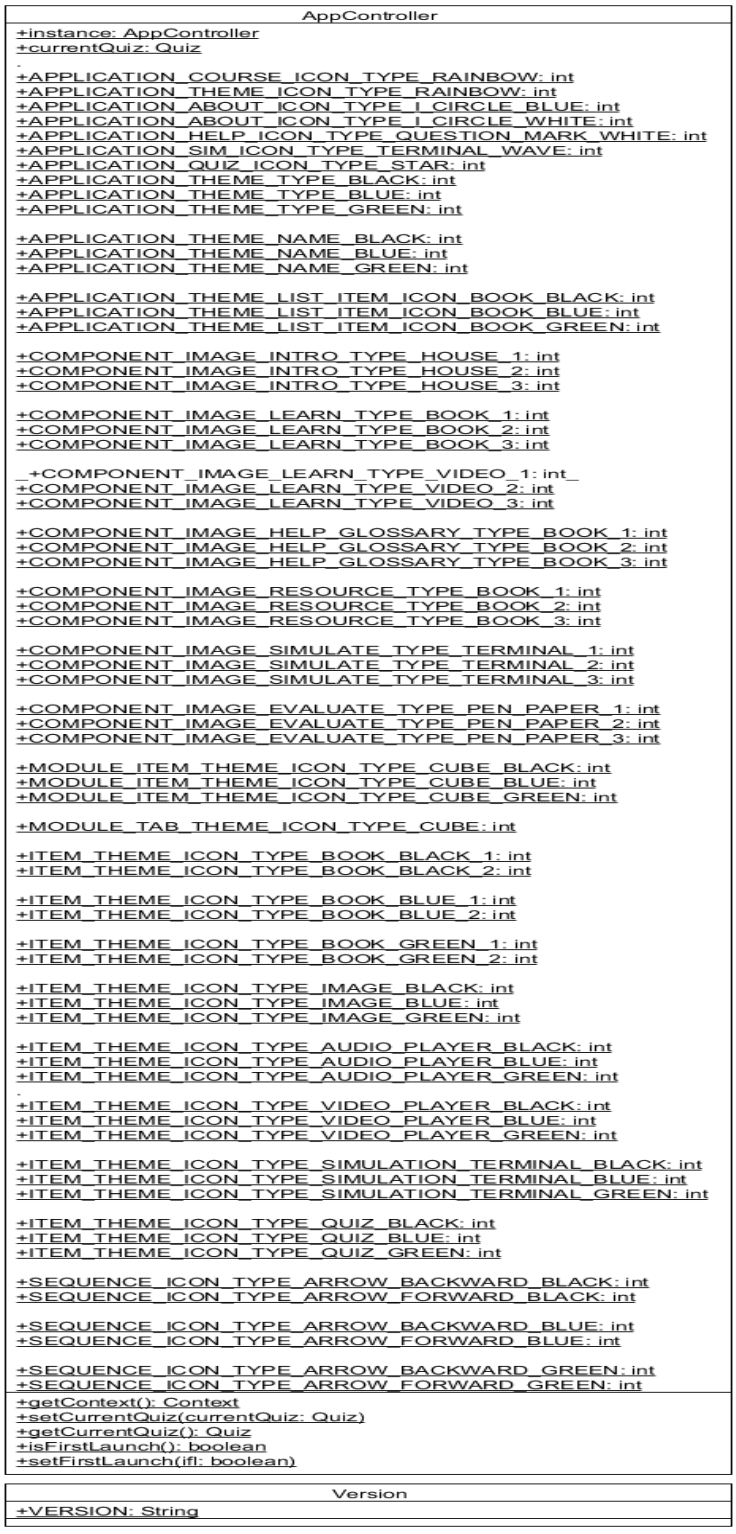
Appendix A7
File Handler Activities and Related Class Diagrams



Appendix A8
Utility Activities and Related Class Diagrams



Appendix A9
AppController and Version Class Diagrams



Appendix B
Content Provider Project Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.contentprovider"
    android:versionCode="1"
    android:versionName="1.0" >

      <uses-permission android:name="android.permission.INTERNET" />
      <!-- Keep the screen from dimming or the processor from sleeping,
      or      uses      the      MediaPlayer.setScreenOnWhilePlaying()      or
MediaPlayer.setWakeMode()
      methods, you must request this permission. -->
      <uses-permission android:name="android.permission.WAKE_LOCK" />

      <supports-screens
            android:largeScreens="false"
            android:normalScreens="true"
            android:smallScreens="true"
            android:anyDensity="true" />

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

    <activity
            android:name="com.example.contentprovider.SetupActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
     </activity>

     <activity
            android:name="com.example.contentprovider.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action
android:name="com.example.contentprovider.MainActivity" />

                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
     </activity>
```

```
     <activity                          android:label="@string/activity_name"
android:name="com.austuniaizu.koyham.nmmla.framework.homepage.HomePageFragmen
tActivity" android:theme="@style/Theme.Styled"

android:configChanges="keyboardHidden|orientation|screenLayout|screenSize|sma
llestScreenSize">
            <intent-filter>
                <action
android:name="com.austuniaizu.koyham.nmmla.framework.homepage.HomePageFragmen
tActivity"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>


        <activity
android:name="com.austuniaizu.koyham.nmmla.framework.util.HtmlHandlerActivity
" android:theme="@style/Theme.Styled">
            <intent-filter>
                <action
android:name="com.austuniaizu.koyham.nmmla.framework.util.HtmlHandlerActivity
"/>
                <category
android:name="com.austuniaizu.koyham.nmmla.framework.DEFAULT"/>
            </intent-filter>
        </activity>

                <activity
android:name="com.austuniaizu.koyham.nmmla.framework.util.AudioHandlerActivit
y" android:theme="@style/Theme.Styled"
            android:screenOrientation="landscape"
android:configChanges="keyboardHidden|orientation|screenLayout|screenSize|sma
llestScreenSize">>
            <intent-filter>
                <action
android:name="com.austuniaizu.koyham.nmmla.framework.util.AudioHandlerActivit
y"/>
                <category
android:name="com.austuniaizu.koyham.nmmla.framework.DEFAULT"/>
            </intent-filter>
        </activity>

        <activity
android:name="com.austuniaizu.koyham.nmmla.framework.util.VideoHandlerActivit
y" android:theme="@style/Theme.Styled"
            android:screenOrientation="landscape"
android:configChanges="keyboardHidden|orientation|screenLayout|screenSize|sma
llestScreenSize">>
            <intent-filter>
                <action
android:name="com.austuniaizu.koyham.nmmla.framework.util.VideoHandlerActivit
y"/>
                <category
android:name="com.austuniaizu.koyham.nmmla.framework.DEFAULT"/>
            </intent-filter>
        </activity>
```

```
        <activity

android:name="com.austuniaizu.koyham.nmmla.framework.util.ImageHandlerActivit
y"
            android:theme="@style/Theme.Sherlock" >
            <intent-filter>
                <action
android:name="com.austuniaizu.koyham.nmmla.framework.util.ImageHandlerActivit
y" />

                <category
android:name="com.austuniaizu.koyham.nmmla.framework.DEFAULT" />
        </intent-filter>
        </activity>

        <activity
android:name="com.austuniaizu.koyham.nmmla.framework.tabmodule.TabModuleFragm
entActivity" android:theme="@style/Theme.Styled"

android:configChanges="keyboardHidden|orientation|screenLayout|screenSize|sma
llestScreenSize">
            <intent-filter>
                <action
android:name="com.austuniaizu.koyham.nmmla.framework.tabmodule.TabModuleFragm
entActivity"/>
                <category
android:name="com.austuniaizu.koyham.nmmla.framework.DEFAULT"

android:parentActivityName="com.austuniaizu.koyham.nmmla.framework.homepage.H
omePageFragmentActivity"/>
                <meta-data
            android:name="android.support.PARENT_ACTIVITY"

android:value="com.austuniaizu.koyham.nmmla.framework.homepage.HomePage" />

            </intent-filter>
        </activity>

        <activity
android:name="com.austuniaizu.koyham.nmmla.framework.listmodule.ListModuleFra
gmentActivity" android:theme="@style/Theme.Styled"

android:configChanges="keyboardHidden|orientation|screenLayout|screenSize|sma
llestScreenSize">
            <intent-filter>
                <action
android:name="com.austuniaizu.koyham.nmmla.framework.datamodel.ComponentListF
ragmentActivity"/>
                <category
android:name="com.austuniaizu.koyham.nmmla.framework.DEFAULT"

android:parentActivityName="com.austuniaizu.koyham.nmmla.framework.homepage.H
omePageFragmentActivity"/>

            </intent-filter>
        </activity>
```

```
         <activity
android:name="com.austuniaizu.koyham.nmmla.framework.listmodule.LearnListTabF
ragmentActivityTest" android:theme="@style/Theme.Styled"

android:configChanges="keyboardHidden|orientation|screenLayout|screenSize|sma
llestScreenSize">
          <intent-filter>
              <action
android:name="comcom.austuniaizu.koyham.mlearnsys.learn.LearnListTabFragmentA
ctivityTest"/>
              <category
android:name="com.austuniaizu.koyham.nmmla.framework.DEFAULT"

android:parentActivityName="com.austuniaizu.koyham.nmmla.framework.homepage.H
omePageFragmentActivity"/>
              <meta-data
          android:name="android.support.PARENT_ACTIVITY"

android:value="com.austuniaizu.koyham.nmmla.framework.homepage.HomePage" />
          </intent-filter>
      </activity>

        <activity
android:name="com.austuniaizu.koyham.nmmla.framework.tabfile.TabFileFragmentA
ctivity" android:theme="@style/Theme.Styled"

android:configChanges="keyboardHidden|orientation|screenLayout|screenSize|sma
llestScreenSize">>
            <intent-filter>
                <action
android:name="com.austuniaizu.koyham.nmmla.framework.tabfile.HtmlFragmentActi
vity"/>
                <category
android:name="com.austuniaizu.koyham.nmmla.framework.DEFAULT"/>
            </intent-filter>
        </activity>

         <activity
android:name="com.austuniaizu.koyham.nmmla.framework.evaluate.QuestionActivit
y" android:theme="@style/Theme.Styled"

android:configChanges="keyboardHidden|orientation|screenLayout|screenSize|sma
llestScreenSize">
            <intent-filter>
                <action
android:name="com.austuniaizu.koyham.nmmla.framework.evaluate.QuestionActivit
y"/>
                <category
android:name="com.austuniaizu.koyham.nmmla.framework.DEFAULT"

android:parentActivityName="com.austuniaizu.koyham.nmmla.framework.homepage.H
omePage"/>
                <meta-data
            android:name="android.support.PARENT_ACTIVITY"

android:value="com.austuniaizu.koyham.nmmla.framework.homepage.HomePage" />
            </intent-filter>
```

```
        </activity>

        <activity
android:name="com.austuniaizu.koyham.nmmla.framework.evaluate.AnswersActivity
" android:theme="@style/Theme.Styled"

android:configChanges="keyboardHidden|orientation|screenLayout|screenSize|sma
llestScreenSize">
            <intent-filter>
                <action
android:name="com.austuniaizu.koyham.nmmla.framework.evaluate.AnswersActivity
"/>
                <category
android:name="com.austuniaizu.koyham.nmmla.framework.DEFAULT"/>
            </intent-filter>
        </activity>

        <activity
android:name="com.austuniaizu.koyham.nmmla.framework.evaluate.ScoreActivity"
android:theme="@style/Theme.Styled"

android:configChanges="keyboardHidden|orientation|screenLayout|screenSize|sma
llestScreenSize">
            <intent-filter>
                <action
android:name="com.austuniaizu.koyham.nmmla.framework.evaluate.EndActivity"/>
                <category
android:name="com.austuniaizu.koyham.nmmla.framework.DEFAULT"/>
            </intent-filter>
        </activity>

        <!-- The default activity of the app; displays search results. -->
        <activity
android:name="com.austuniaizu.koyham.nmmla.framework.search.SearchDictionaryA
ctivity"
                  android:launchMode="singleTop"
android:theme="@style/Theme.Styled">

            <intent-filter>
                <action
android:name="com.austuniaizu.koyham.nmmla.framework.search.SearchDictionaryA
ctivity" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>

            <!-- Receives the search request. -->
            <intent-filter>
                <action android:name="android.intent.action.SEARCH" />
                <!-- No category needed, because the Intent will specify this
class component-->
            </intent-filter>

            <!-- Points to searchable meta data. -->
            <meta-data                    android:name="android.app.searchable"
android:resource="@xml/searchable" />

        </activity>
```

```
        <!-- Displays the definition of a word. -->
        <activity
android:name="com.austuniaizu.koyham.nmmla.framework.search.WordActivity"
android:theme="@style/Theme.Styled"/>

        <!-- Provides search suggestions for words and their definitions. -->
        <provider
android:name="com.austuniaizu.koyham.nmmla.framework.search.DictionaryProvide
r"

android:authorities="com.austuniaizu.koyham.nmmla.framework.search.Dictionary
Provider" />

        <!-- Points to searchable activity so the whole app can invoke
search. -->
        <meta-data android:name="android.app.default_searchable"
                   android:value=".SearchDictionaryActivity" />

     <activity              android:name="com.example.myfirstapp.TestActivity"
android:theme="@style/Theme.Styled"

android:configChanges="keyboardHidden|orientation|screenLayout|screenSize|sma
llestScreenSize">
            <intent-filter>
                <action android:name="com.example.myfirstapp.TestActivity"/>
                <category android:name="com.example.myfirstapp.DEFAULT"/>
            </intent-filter>
        </activity>

          <activity
android:name="com.example.myfirstapp.DisplayMessageActivity"
android:theme="@style/Theme.Styled"

android:configChanges="keyboardHidden|orientation|screenLayout|screenSize|sma
llestScreenSize">
            <intent-filter>
                <action
android:name="com.example.myfirstapp.DisplayMessageActivity"/>
                <category android:name="com.example.myfirstapp.DEFAULT"/>
            </intent-filter>
        </activity>

    </application>

</manifest>
```

Appendix C
Content Provider Project SetupActivity

```
package com.example.contentprovider;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;
import android.content.Context;
import android.content.Intent;
import android.content.res.AssetManager;
import android.database.SQLException;
import android.os.Bundle;
import android.widget.ImageView;
import com.actionbarsherlock.app.SherlockActivity;
import com.austuniaizu.koyham.nmmla.framework.AppController;
import com.austuniaizu.koyham.nmmla.framework.R;
import com.austuniaizu.koyham.nmmla.framework.datamodel.AtomicComponent;
import com.austuniaizu.koyham.nmmla.framework.datamodel.Component;
import com.austuniaizu.koyham.nmmla.framework.datamodel.ModularComponent;
import com.austuniaizu.koyham.nmmla.framework.datamodel.Content;
import com.austuniaizu.koyham.nmmla.framework.datamodel.Course;
import com.austuniaizu.koyham.nmmla.framework.datamodel.FilesDBHelper;
import com.austuniaizu.koyham.nmmla.framework.datamodel.EvalComponent;
import com.austuniaizu.koyham.nmmla.framework.datamodel.File;
import com.austuniaizu.koyham.nmmla.framework.datamodel.SimComponent;
import com.austuniaizu.koyham.nmmla.framework.datamodel.Theme;
import com.austuniaizu.koyham.nmmla.framework.datamodel.Module;
import com.austuniaizu.koyham.nmmla.framework.datamodel.RenderingMode;
import com.austuniaizu.koyham.nmmla.framework.datamodel.Simulation;
import com.austuniaizu.koyham.nmmla.framework.evaluate.Quiz;
import com.austuniaizu.koyham.nmmla.framework.homepage.CourseBundle;
import com.austuniaizu.koyham.nmmla.framework.homepage.ThemeBundle;
import
com.austuniaizu.koyham.nmmla.framework.homepage.HomePageFragmentActivity;

public class SetupActivity extends SherlockActivity {

      private ImageView imageView;

      public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setTheme(AppController.APPLICATION_THEME_TYPE_BLACK);
        /** Set Content View to Splashscreen/Welcome Page Image **/
        setContentView(R.layout.background_image_view);
        imageView = (ImageView) findViewById(R.id.screen_image);
        imageView.setBackgroundResource(R.drawable.bgd_image);

  /*************************************************************/
  /** File Types for Single or Atomic files not loaded from DB, but
    * specified on this setup page, e.g. About **/
 /*************************************************************/
       /**
```

```
         *  Text  - 0   (Used to indicate Search dictionary file)
         *  Html  - 1
         *  Image - 2
         *  Audio - 3
         *  Video - 4
         **/
/*****************************************************************/
      /**  Media  Types  for  Modules  being  loaded  from  SQLite  DB  **/
/*****************************************************************/
        /**
         *  Text  - 0    (Not used)
         *  Html  - 1
         *  Image - 2
         *  Audio - 3
         *  Video - 4
         */
/*****************************************************************/
   /**
    *  If you prefer to define your own theme bundle, or re-order
    *  the default themes, do it as follows; Otherwise you may
    *  decide to use the default bundle provided by the framework.
   **/

      Theme theme1 = new Theme(
      AppController.APPLICATION_THEME_NAME_BLUE,
      AppController.APPLICATION_THEME_TYPE_BLUE,
      AppController.MODULE_TAB_THEME_ICON_TYPE_CUBE,
      AppController.MODULE_ITEM_THEME_ICON_TYPE_CUBE_BLUE,
      AppController.ITEM_THEME_ICON_TYPE_BOOK_BLUE,
      AppController.ITEM_THEME_ICON_TYPE_IMAGE_BLUE,
      AppController.ITEM_THEME_ICON_TYPE_AUDIO_PLAYER_BLUE,
      AppController.ITEM_THEME_ICON_TYPE_VIDEO_PLAYER_BLUE,
      AppController.ITEM_THEME_ICON_TYPE_SIMULATION_TERMINAL_BLUE,
      AppController.ITEM_THEME_ICON_TYPE_QUIZ_BLUE,
      AppController.SEQUENCE_ICON_TYPE_ARROW_BACKWARD_BLUE,
      AppController.SEQUENCE_ICON_TYPE_ARROW_FORWARD_BLUE);

      Theme theme2 = new Theme(
      AppController.APPLICATION_THEME_NAME_BLACK,
      AppController.APPLICATION_THEME_TYPE_BLACK,
      AppController.MODULE_TAB_THEME_ICON_TYPE_CUBE,
      AppController.MODULE_ITEM_THEME_ICON_TYPE_CUBE_BLACK,
      AppController.ITEM_THEME_ICON_TYPE_BOOK_BLACK,
      AppController.ITEM_THEME_ICON_TYPE_IMAGE_BLACK,
      AppController.ITEM_THEME_ICON_TYPE_AUDIO_PLAYER_BLACK,
      AppController.ITEM_THEME_ICON_TYPE_VIDEO_PLAYER_BLACK,
      AppController.ITEM_THEME_ICON_TYPE_SIMULATION_TERMINAL_BLACK,
      AppController.ITEM_THEME_ICON_TYPE_QUIZ_BLACK,
      AppController.SEQUENCE_ICON_TYPE_ARROW_BACKWARD_BLACK,
      AppController.SEQUENCE_ICON_TYPE_ARROW_FORWARD_BLACK);

      Theme theme3 = new Theme(
      AppController.APPLICATION_THEME_NAME_GREEN,
      AppController.APPLICATION_THEME_TYPE_GREEN,
      AppController.MODULE_TAB_THEME_ICON_TYPE_CUBE,
      AppController.MODULE_ITEM_THEME_ICON_TYPE_CUBE_GREEN,
      AppController.ITEM_THEME_ICON_TYPE_BOOK_GREEN,
```

```
      AppController.ITEM_THEME_ICON_TYPE_IMAGE_GREEN,
      AppController.ITEM_THEME_ICON_TYPE_AUDIO_PLAYER_GREEN,
      AppController.ITEM_THEME_ICON_TYPE_VIDEO_PLAYER_GREEN,
      AppController.ITEM_THEME_ICON_TYPE_SIMULATION_TERMINAL_GREEN,
      AppController.ITEM_THEME_ICON_TYPE_QUIZ_GREEN,
      AppController.SEQUENCE_ICON_TYPE_ARROW_BACKWARD_GREEN,
      AppController.SEQUENCE_ICON_TYPE_ARROW_FORWARD_GREEN);

      List<Theme> themes = new ArrayList<Theme>();

            themes.add(theme1);
            themes.add(theme2);
            themes.add(theme3);

      ThemeBundle userThemeBundle = new ThemeBundle(
         "Theme", AppController.APPLICATION_THEME_ICON_TYPE_RAINBOW, themes);

/*****************************************************************/    /**
Module Quizzes - Create app's quiz objects **/
/*****************************************************************/

      /** Quiz -- These quizzes will be assigned to modules **/
      // For space only few quiz items as elsewhere, are added
      Quiz module1_quiz = new Quiz("module1_quiz", "Quiz01", 5);
      Quiz module2_quiz = new Quiz("module2_quiz", "Quiz02", 5);

/*****************************************************************/
/** Component - Simulate - List_Item_Simulation **/
/*****************************************************************/
   /**
    *  Declare the array list of simulation activities. For now,
    *  they are manually added because we have not found a way
    *  of converting the class names read as strings from DB to
    *  classes of type Class<?> which can be sent an intent
   **/

   /** Create objects of simulation activities **/
   Simulation simulate_activity1 = new Simulation("Sim Topic 1",
                        com.example.myfirstapp.TestActivity1.class);
   Simulation simulate_activity2 = new Simulation("Sim Topic 2",
                        com.example.myfirstapp.TestActivity2.class);
   Simulation simulate_activity3 = new Simulation("Sim Topic 3",
                        com.example.myfirstapp.TestActivity3.class);
   Simulation simulate_activity4 = new Simulation("Sim Topic 4",
                        com.example.myfirstapp.TestActivity4.class);

   List<Simulation> listActivities = new ArrayList<Simulation>();
                        listActivities.add(simulate_activity1);
                        listActivities.add(simulate_activity2);

   List<Simulation> listActivities2 = new ArrayList<Simulation>();
                        listActivities2.add(simulate_activity3);
                        listActivities2.add(simulate_activity4);

/*****************************************************************/
      /** Component Introduction - One module, TAB MODE **/
```

```
/*****************************************************************/

   //List<File> intro_files = getFilesFromDB(this,"intro_db","Introduction");
   //The above is an alternative to retrieve from DB
    Module intro_module = new Module("Introduction", false,
                         getFilesFromFolder("intro_html_folder/module",4),
                         null, null, RenderingMode.TAB,1);
    List<Module> intro_module_list = new ArrayList<Module>();
    intro_module_list.add(intro_module);

    Component intro_component = new ModularComponent("Introduction",
                         AppController.COMPONENT_IMAGE_INTRO_TYPE_HOUSE_1,
                         intro_module_list, RenderingMode.TAB);

/*********************************************************************/
/** Component Learn - Doc **/
/*********************************************************************/

    Module learn_doc_module1 = new Module("Module 1",
                         getFilesFromFolder("learn_html_folder/module1",4),
                         listActivities, module1_quiz, RenderingMode.LIST,1);
    Module learn_doc_module2 = new Module("Module 2",
                         getFilesFromFolder("learn_html_folder/module2",4),
                         listActivities2, module2_quiz, RenderingMode.TAB,1);

    /*** Adding all the modules to a list  ***/
    List<Module> learn_doc_modules_list = new ArrayList<Module>();
    learn_doc_modules_list.add(learn_doc_module1);
    learn_doc_modules_list.add(learn_doc_module2);

    Component learn_doc_component = new ModularComponent("Learn - Doc",
                         AppController.COMPONENT_IMAGE_LEARN_TYPE_BOOK_1,
                         learn_doc_modules_list, RenderingMode.LIST);

/*********************************************************************/
/** Component Learn - Video **/
/*********************************************************************/

    Module learn_video_module1 = new Module("Module 1",
                         getFilesFromFolder("video_folder/module1",3), null,
                         module1_quiz, RenderingMode.LIST,4);
    Module learn_video_module2 = new Module("Module 2",
                         learn_video_module2_files, null, module2_quiz,
                         RenderingMode.LIST,4);

    /*** Adding all the modules to a list ***/
    List<Module> learn_video_modules_list = new ArrayList<Module>();
    learn_video_modules_list.add(learn_video_module1);
    learn_video_modules_list.add(learn_video_module2);

    Component learn_video_component = new ModularComponent("Learn - Video",
                         AppController.COMPONENT_IMAGE_LEARN_TYPE_VIDEO_1,
                         learn_video_modules_list, RenderingMode.LIST);

/*********************************************************************/
```

```
/*************************************************************************/
/** Component Learn - Slide **/
/*************************************************************************/

    Module learn_slide_module1 = new Module("Module 1",
                          getFilesFromFolder("learn_slide_folder/module1",3),
                          null, module1_quiz, RenderingMode.LIST,2);
    Module learn_slide_module2 = new Module("Module 2",
                          getFilesFromFolder("learn_slide_folder/module2",3),
                          null, module2_quiz, RenderingMode.TAB,2);

    List<Module> learn_slide_modules_list = new ArrayList<Module>();
    learn_slide_modules_list.add(learn_slide_module1);
    learn_slide_modules_list.add(learn_slide_module2);

    Component learn_slide_component = new ModularComponent("Learn - Slide",
                          AppController.COMPONENT_IMAGE_LEARN_TYPE_BOOK_2,
                          learn_slide_modules_list, RenderingMode.LIST);

/*************************************************************************/
** Component - Evaluate on HP using Reference to "learn_doc_component" **/
/*************************************************************************/

   Component evaluate_component = new EvalComponent("Evaluate",
                     AppController.COMPONENT_IMAGE_EVALUATE_TYPE_PEN_PAPER_1,
                         (ModularComponent) learn_doc_component);

/*************************************************************************/
/** Component - Simulate on HP using Reference to "learn_doc_component" **/
/*************************************************************************/

   Component simulate_component = new SimComponent("Simulate",
                      AppController.COMPONENT_IMAGE_SIMULATE_TYPE_TERMINAL_1,
                         (ModularComponent) learn_doc_component);

/*************************************************************************/
/** Component - Resources **/
/*************************************************************************/

    File resources_file = new File ("Resources",
                      "resource_html_folder/Resources.html",1);

    Component resources_component = new AtomicComponent("Resources",
                      AppController.COMPONENT_IMAGE_RESOURCE_TYPE_BOOK_3,
                      resources_file);

/*************************************************************************/
/** Component Help - Single_Module_Html without icon and quiz **/
/*************************************************************************/

    List<File> help_files = getFilesFromDB(this, "help_db", "Help");
    Module module_dbtest = new Module("Help", false, help_files,
                      null, null, null, 1);
    List<Module> help_modules_list = new ArrayList<Module>();
    help_modules_list.add(module_dbtest);
    ModularComponent help_component = new ModularComponent("Help",
                     AppController.COMPONENT_IMAGE_HELP_GLOSSARY_TYPE_BOOK_1,
```

```
                    help_modules_list, RenderingMode.LIST);

/*************************************************************************/
/** Component - Help -- On ActionBar **/
/*************************************************************************/

    ModularComponent help_component_AB = new ModularComponent("Help",
                AppController.APPLICATION_HELP_ICON_TYPE_QUESTION_MARK_LIGHT,
                    help_modules_list, RenderingMode.LIST);

/*************************************************************************/
/** Component - About -- On ActionBar **/
/*************************************************************************/

   Content about_file = new File("About",0,"about_html_folder/about.html",1);

    AtomicComponent comp_about = new AtomicComponent("About",
        AppController.APPLICATION_ABOUT_ICON_TYPE_I_CIRCLE_BLUE, about_file);

/*************************************************************************/
/** Component - Search -- On ActionBar **/
/*************************************************************************/

    Content search_file = new File ("definitions", 0,"definitions", 0);
         // The third is used as the text file for the dictionary
    AtomicComponent comp_search = new AtomicComponent("Search",
         AppController.COMPONENT_IMAGE_INTRO_TYPE_HOUSE_1, search_file);


/*************************************************************************/
/** Putting the list of components into Course Object instances **/
/** Note: Same components were added to Course 2 and Course 3 **/
/*************************************************************************/

    List<Component> course1_components = new ArrayList<Object>();
    course1_components.add(intro_component);
    course1_components.add(learn_doc_component);
    course1_components.add(learn_video_component);
    course1_components.add(learn_slide_component);
    course1_components.add(simulate_component);
    course1_components.add(evaluate_component);
    course1_components.add(resources_component);
    course1_components.add(help_component);

    /** Course 2**/
    List<Component> course2_components = new ArrayList<Object>();
    course2_components.add(intro_component);
    course2_components.add(learn_doc_component);
    course2_components.add(learn_video_component);
    course2_components.add(learn_slide_component);
    course2_components.add(simulate_component);
    course2_components.add(evaluate_component);
    course2_components.add(resources_component);
    course2_components.add(help_component);

    /** Course 3 **/
    List<Component> course3_components = new ArrayList<Object>();
```

```
course3_components.add(intro_component);
course3_components.add(learn_doc_component);
course3_components.add(learn_video_component);
course3_components.add(learn_slide_component);
course3_components.add(simulate_component);
course3_components.add(evaluate_component);
course3_components.add(resources_component);
course3_components.add(help_component);

Course course1 = new Course("Automata",course1_components);
Course course2 = new Course("Compiler",course2_components);
Course course3 = new Course("Data Structures",course3_components);

List<Course> courses = new ArrayList<Course>();
courses.add(course1);
courses.add(course2);
courses.add(course3);

CourseBundle courseBundle = new CourseBundle("Courses",
            AppController.APPLICATION_QUIZ_ICON_TYPE_STAR, courses);

HomePageFragmentActivity.setAppDatabasePath(
            "/data/data/com.example.contentprovider/databases/");
HomePageFragmentActivity.setAppPackageName(
            "com.example.contentprovider"); //CP's root package name
HomePageFragmentActivity.setAppName("NMMLA Framework");
HomePageFragmentActivity.setAppCourseBundle(courseBundle)
HomePageFragmentActivity.setAppThemeBundle((userThemeBundle));
HomePageFragmentActivity.setAbout(comp_about);
HomePageFragmentActivity.setSearch(comp_search);
HomePageFragmentActivity.setHelp(help_component_AB);

Intent intent = new Intent(this, HomePageFragmentActivity.class);
startActivity(intent);

   // Destroy the activity, which presents a blank splash screen

   finish();
  }

  /** To read files (e.g. HTML, Image) and Quiz items From DB **/

  public static List<File> getFilesFromDB(Context context, String
            databaseName, String tableName) throws Error {

        /** DBHelper assists in retrieving items from DB **/

        FilesDBHelper dbHelper = new FilesDBHelper(
              context, databaseName, tableName,
              "/data/data/com.example.contentprovider/databases/");

        try {
              dbHelper.createDataBase(); /** 1. Create DB **/
        } catch (IOException ioe) {
              throw new Error("Unable to create database");
        }
        try {
```

```
                dbHelper.openDataBase(); /** 2. Open DB **/
            }catch(SQLException sqle){
                throw sqle;
            }

            List<File> items = dbHelper.getFileSet();
                                        /** 3. Get Dataset **/
            dbHelper.close();           /** 4. Close DB **/
            return items;
      }

      /** To read modular files from a folder in asset folder **/

public List<File> getFilesFromFolder(String folderName, int
            fileSuffixLength) throws Error {

            AssetManager assetManager = getAssets();
            List<File> module_files = new ArrayList<File>();
            // To get names of all files inside the "folderName"
            try {
                String[] files = assetManager.list(folderName);
                for(int i=0; i<files.length; i++)
                {
                    String fileName = files[i].substring(3,
                  files[i].length()-(fileSuffixLength+1));
                    // to remove suffix e.g ".png"
                    File file = new File(fileName,
                    folderName+"/"+files[i]);
                    module_files.add(file);
                }
            } catch (IOException e1) {
                e1.printStackTrace();
            }

            return module_files;
      }
}
```