Teaching Mobile App Development: Choosing the best development tools in practical labs

Dr. Daniel Dang
Eastern Institute of Technology
ddang@eit.ac.nz

Dr. David Skelton
Eastern Institute of Technology
dskelton@eit.ac.nz

ABSTRACT

As mobile app development becomes more mainstream, many educational institutions in New Zealand recognize a need to integrate this subject into the programming-related curriculum. However, one challenge is to select the best framework and development tool to use in the practical labs amongst many available frameworks. Choosing the right tool and language is an important factor for success in teaching and learning within this subject. This paper first presents a detailed discussion on different development tools (language) - Android Studio (Java/Kotlin), Xcode (Swift), Flutter (Dart), React Native (JavaScript), Ionic (HTML5, CSS3, TypeScript), Xamarin (C#), Cordova (HTML5, CSS3, JavaScript), and Appcelerator (JavaScript). This paper concludes by suggesting that the Android Studio IDE (Java) and Flutter Framework are ideal choices for use within practical labs. By using Android Studio, students are able to develop native apps, which are usually high-performance and good user interface, for Android devices. By using the Flutter framework, which is integrated into Android Studio, students gain the capability of creating cross-platform apps running on both Android and iOS smartphones. Furthermore, the paper recommends a collection of eight specific topics which should be covered in the practical sessions to provide students with sufficient technical skills to develop all types of mobile apps at their disposition. While the focus is on the Android platform, these topics can be used to teach other native iOS or cross-platform apps.

Keywords: Mobile app development, native and cross-platform apps, practical labs, frameworks.

1. INTRODUCTION

A mobile application, also referred to as an app in short, describes software applications designed and developed to run on a mobile device, such as a mobile phone or tablet. As mobile app development becomes more mainstream, most educational institutions recognize a need to integrate this subject into the programming-related curriculum. *How to integrate the course of mobile app development in the Computer science curricula?* Minaie et al. (2011) and Mahmoud (2008) discussed various approaches that are used by different colleges and universities around the world to integrate mobile computing into their computer science and engineering curricula. The authors classified the integration into five models: (1) offering undergraduate courses on Mobile Computing, (2) offering graduate courses on Mobile Computing, (3) integrating Mobile Computing concepts into their traditional courses, (4) combining model 1, 2, and 3, and (5) offering Mobile Computing as an area of research to their graduate students. Seyam et al. (2016) stated that this integration requires a good understanding of mobile software development that is not just a collection of theory topics but also delivery modes such as lectures, interactive tutorials. and pair programming. Esakia & McCrickard (2016) presented an adapted model for teaching mobile app development. Their model suggested using a framework that enables mobile-specific topics like location, notification, and sensors. El-Tawab et al. (2018) proposed a project-based methodology of teaching mobile app development for undergraduate students, the authors also described the challenge that educators faced was in choosing the best modern programming language and framework to prepare coding materials when teaching the subject. This includes the choice between native apps or cross-platform apps. In another related study, Khmelevsky & Voytenko (2016) presented a new paradigm for teaching mobile application development, focusing on the development of mobile applications during capstone projects and involving industrial customers.

This quality assured paper appeared at the 10th annual conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2019) and the 32nd Annual Conference of the National Advisory Committee on Computing Qualifications, Nelson, NZ, Oct. 9-11, during ITx 2019.

This paper, however, investigates the challenges in selecting development tools and content to be taught in mobile app development courses to undergraduate students. The paper then recommends a collection of eight specific topics which should be covered in the practical sessions to provide students with sufficient technical skills to develop all types of mobile apps at their disposition.

Section 2 of this paper discusses and compares many available development tools such as Android Studio (Java/Kotlin), Xcode (Swift), Flutter (Dart), React Native (JavaScript), Ionic (HTML5, CSS3, TypeScript), Xamarin (C#), Cordova (HTML5, CSS3, JavaScript), Appcelerator Titanium (JavaScript) in order to assess the best choice to be used in practical labs. Section 3 presents the list of eight topics: *(1) User Interface Design, (2) Event Handling, (3) Multi-threading Handling, (4) Graphics and Animation, (5) Use of Mobile Sensor/Components, (6) Wireless Connectivity (WiFi, Bluetooth), (7) Internal Data Storage, and (8) Real-time Database (Firebase)*. These topics along with concrete practical exercises provide students with not only foundation theories but also programming skills to implement all types of mobile apps such as gaming, business, educational, lifestyle, entertainment, and utility apps. These topics have been proven to be a success when they were taught at the Bachelor of Creative Software (BCS) at former Animation College since July 2016.

2. DISCUSSION ON DEVELOPMENT TOOLS AND FRAMEWORKS

There are currently two common types of mobile apps: native and multi-platform apps. The native apps are those which are developed for a single mobile operating system (Android, iOS, Window phone, Blackberry, etc.) exclusively. Their major advantage is high performance, optimized and the best user experience because they use native device user interface. It appears from scanning recruitment channels that most mobile job advertisements are looking for those who are able to develop native apps. However native apps cannot be run on all devices, so the company normally has to establish two different teams to develop the same app, one for iOS devices and another for Android devices. This leads to "duplication codes" and higher cost compared to other types of apps. Multi-platform apps are a mix of native apps and web technologies (HTML5, CSS, and JavaScript); so multi-platform apps are able to remove the "duplication codes". However, they often exhibit poor performance, lack of optimization and app GUI inability to look exactly the same way on different devices.

As of 2017, when we started teaching mobile app development courses – CS102 and CS103 - on the Bachelor of Creative Software (BCS) programme, we faced the challenge to select the most suitable framework and development tool to use in the practical labs amongst many frameworks available as shown in table 1. Choosing the right development tool and the right programming language is a key factor for success in both teaching and learning.

Table 1: Available frameworks to develop native and cross-platform mobile apps

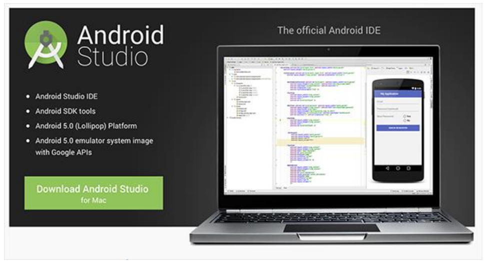
Frameworks	Features
Android Studio *(*)*	Introduced in 2013 by Google. Open-source framework to develop native Android apps. Development language: Java/Kotlin.
Xcode *(*)*	Introduced in 2003 by Apple. Open-source framework to develop native iOS apps. Development language: Swift. Xcode *(*)*
Flutter	Introduced in 2017 by Google. Open-source framework to develop cross-platform apps. Development language: Dart.
React Native	Introduced in 2015 by Facebook. Open-source framework to develop cross-platform apps. Development language: JavaScript.
Ionic	Introduced in 2013 by MIT. Open-source framework to develop cross-platform apps. Development language: HTML5, CSS3, TypeScript.
Xamarin	Introduced in 2011, now owned by Microsoft. Licensed framework to develop native and multi-platform apps. Development language: C#.
Cordova	Introduced in 2009 (Former Phonegap), owned by Adobe System. Open-source framework to develop cross-platform apps. Development language: HTML5, CSS3, JavaScript.
Appcelerator Titanium	Introduced in 2008 by Appcelerator Inc. Open-source framework to develop cross-platform apps. Development language: JavaScript.

(*) Tools and frameworks to develop native apps.

Some lecturers argued that students have learnt website development and web technologies (HTML5, CSS3, JavaScript) so the cross-platform frameworks such as Ionic, Cordova or React Native would be a good choice. It would be a smooth continuation in learning and a seamless transition to learn new domain based on existing knowledge. However, the course learning outcomes state that the course provides students with the understanding of native mobile device architecture, native libraries and the best practices for the app development; it means that Android Studio and Xcode are the best choices to meet this learning outcome. The cross-platform apps are in fact mobile-designed websites and disguised in a native wrapper. Moreover, the chosen tool should support team work and project collaboration. In this point, the Android Studio and Xcode are the right choices because they support GitHub which leverage the collaborative projects. As a result, we chose both Android Studio and Xcode in practical sessions.

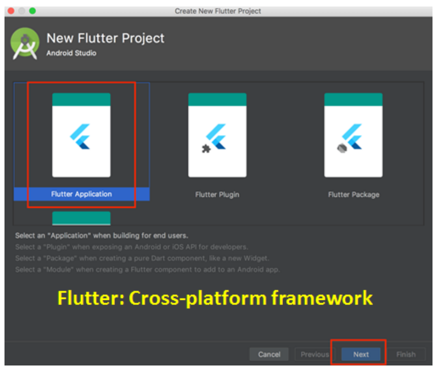
More specifically, students learnt Android architecture and Android app development in the first half of the semester, then moved to learn iOS apps in the second half. The learning result was quite positive. Students gained a strong foundation in both Android and iOS architecture, the ability to design and implement native apps for both iOS and Android platforms. However, from our observation, students had trouble when learning two different platforms in one course. Student's engagement in the second part (iOS native app development) dropped significantly compared to the first part. In the survey at the end of the semester, students commented that they sensed they had learnt the theory, concepts, and practical work twice because there were many similarities between two platforms – Android and iOS. They said they only learnt the syntax of a new language (Swift) and the use of a new tool (Xcode) but they did not learn new programming skills. As for projects, even though students were at disposition to choose to implement their apps with Android IDE or Xcode IDE, more than 90% of groups chose Android Studio as the development tool to carry out their projects.

We also observed that students wasted too much time in learning to use the Xcode tool and Swift syntax but faced little use or application later on in their projects or assignments. Therefore, in semester 1, 2018, we made a proposition to students, instead of learning Xcode IDE and Swift, they would learn the Flutter framework (Dart) to develop a cross-platform app for both iOS and Android OS by still using Android Studio IDE. Students showed their interest in Flutter by picking up the Flutter framework to implement their apps because they could publish their apps on both the App Store and the Google App Store. Most importantly, they did not feel that they were repeating their learning.



- Android Studio IDE
- Android SDK tools
- Android 5.0 (Lollipop) Platform
- Android 5.0 emulator system image with Google APIs

Figure 2.1: Android Studio IDE to develop Android apps.



Select an "Application" when building for end users.
Select a "Plugin" when exposing an Android or iOS API for developers.
Select a "Package" when creating a pure Dart component, like a new Widget.
Select a "Module" when creating a Flutter component to add to an Android app.

Figure 2.2: Create a new cross-platform app using Flutter framework in Android Studio IDE.

We have recently investigated two major educational institutions in New Zealand who have been teaching mobile app development to see which platforms they are using in practical labs. Most universities (AUT, Auckland University, University of Otago) integrate mobile app development in their curriculum but do not offer it as an independent course. Massey University offers a paper called “Mobile Application Development” in the Bachelor of Computer Science degree. This paper is an introduction to mobile app design where students gain practical skills in Android apps. The University of Canterbury has a paper on “Mobile computing platforms” offered by the Computer Science and Software Engineering department. The course uses the Android Platform as the main demonstration and practical works; the other native platform (iOS and WinPhone) and cross-platform approach are brief consideration. Edenz College teaches a “Mobile application development” paper and they use Android Studio as the main development tool. Open Polytechnic offers an online course for developing mobile apps using the C# programming language and Xamarin tool. It can be seen that most educational institutions in NZ use Android Studio as the development tool in practical labs.

In summary, we recommend that the best choice of tool, programming language, and framework are Android Studio IDE, Java language, and Flutter framework for teaching mobile app development to undergraduate students in any “mobile app development” paper.

3. SUGGESTED PRACTICAL TOPICS

El-Tawab et al. (2018) present a methodology to teach mobile app development for undergraduate students in project-based classes. In this approach, students were responsible for teaching themselves with the supervision of the instructor. However, students only learn specific topics and specific technical skills required for their own projects. They do not gain a deep understanding of the core skills to implement different types of mobile app projects. Furthermore, the development of mobile applications is a multidisciplinary field that includes UI design, user interaction, software development, programming, database, networking, security, and a number of other traditional areas of computer science (Mahmoud, 2008). In another related study, Gordon (2013) proposes six topics for mobile programming: user interface, device cooperation, hardware issues, data handling, and application interaction and programming issues. Muyan-Özçelik (2017) argued that we should focus on object-oriented programming (abstraction, encapsulation, inheritance, and polymorphism) and design patterns in mobile application development.

So what topics should be taught in the mobile application development course particularly in practical labs? To the best of the current author’s knowledge, there is minimal research which addresses this question. This section presents eight separate technical topics along with practical exercises that would not only demonstrate the essential concepts and principles of the Android framework but also provide enough technical skills for students to create any type of mobile apps at their disposition as shown in table 2. These topics are based on major chapters in Allan book (2013) and Iversen. & Eierman book (2014). Moreover, these topics have been proven a success while we used them to teach at the BCS degree for the former Animation College. Those topics not only demonstrate the essential concepts and principles of the Android framework but also provide enough technical skills for students to create any type of mobile apps at their disposition.

Table 2: Suggested topics in practical sessions

Topic	Content
User Interface Design	Layout, UI Controls, Styles & Themes, App Resources, App Multiple-activities
Event Handling	Click, Swipe, Drag-and-Drop
Multi-threads	Services, Multi-threads Handling
Graphics and Animation	Canvas, Animation techniques (Frames, Tween, Property)
Device hardware and sensor	Camera, GPS, Accelerometer, Compass, Microphones, Speaker
Wireless connectivity	Wi-Fi, Bluetooth
Internal data storage	Shared Preferences, SQLite database
Real-time database	Cloud Firebase

3.1 Topic 1: Design App User Interface

It is easy to understand why this topic must be the first priority in any app development course as every mobile app requires a graphical and touch-sensitive screen that allows users to interact with the app. The first step in app development is to design its UI based on app mock-ups and wireframes. Figure 3.1 is an example of the outcome that students developed in their practical session.

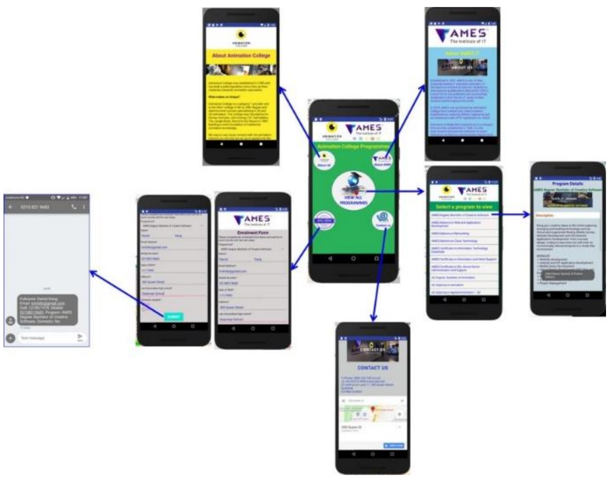


Figure 3.1: Building an app UI by using different layouts, UI controls, and multiple-activities.

In order to implement the above app UI, students learn different kinds of layouts (LinearLayout and RelativeLayout), common UI controls (TextView, Button, EditText, ImageView,CheckBox, RadioButton, ListView, DatePicker, etc.). They also learn to organize and store app resources such as images, colors, strings, layout, text files, etc. Students are able to use the Activity component and navigate through multiple Activities by using an Intent object.

3.2 Topic 2: User Event Handling

The second topic in the practical sessions is to handle events in mobile apps. An event is a user’s interaction with the app. There are different types of user events such as click (single or double-tap), swipe, drag-and-drop, and pinch-to-room, as shown in figure 3.2a.

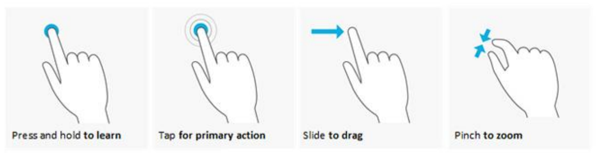


Figure 3.2a: Some types of user's events

In order to capture the event and then execute an action responding to that event, students learn to set UI controls to "Event Listener" and write codes in the relevant event method. The figure 3.2b shows an outcome of a practical lab where students are able to create a simple quiz app in Android. The quiz app reads questions from a string-array stored in the strings.xml file and display questions one by one. When users select an answer, a Toast Message will pop up to indicate whether the answer is right or wrong. Users click the "Next question" button to proceed.

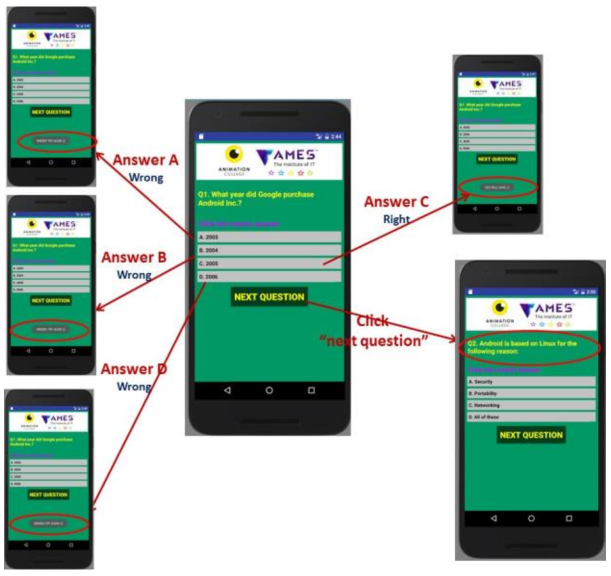


Figure 3.2b: A simple multiple-choice quiz app: a button "Next Question" to proceed.

3.3 Topic 3: Services & Multiple threads used in Internet connection

In Android, a service is a component that runs in the background to perform long-running operations without needing to interact with the user and it works even if the application is destroyed. In order to learn about service and multiple threading in Android, the practical exercise in figure 3.3 guides students to create a simple "utility" app. This "utility" app uses background services for multiple tasks: playing a song, downloading files (audio file, image file, and text file) from a given web page (URL). Through this exercise, students also learn to connect their apps to the internet or any other local network and perform network operations by using HttpURLConnection class.

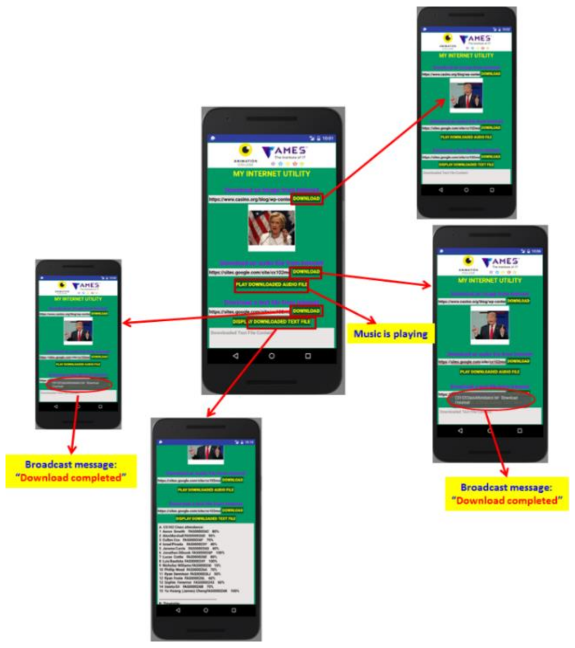


Figure 3.3: Network Connection and Download Files using Service and Multi-threading in Android

3.4 Topic 4: Graphics and Animation

Mobile games are one of the most common apps and are popular with students. In this practical topic, students learn a rich set of powerful animation techniques and Canvas in Android. The canvas provides 2D-drawing methods for rendering custom graphics onto a canvas as shown in figure 3.4a.

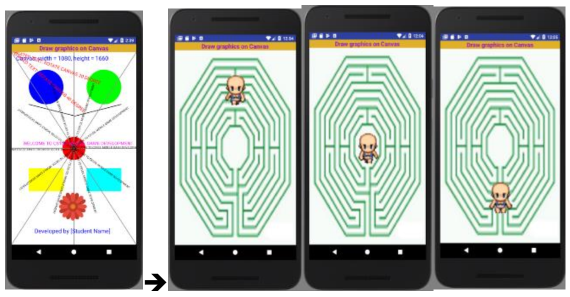

Draw graphics on Canvas
Canvas width = 1080, height = 1660
Developed by [Student Name]

Figure 3.4a: Draw 2D-graphics on canvas and create an illusion of motion

Through this lab, students also learn three different animation techniques in Android: frame, tween, and properties to create an animated scene as shown in figure 3.4b. The scene has a running clock, sun-rise, and flying dragon.

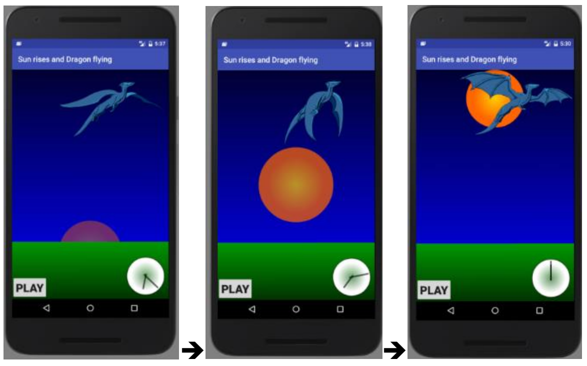


Figure 3.4b: An animated scene uses Animation techniques: frame, tween, and property.

3.5 Topic 5: Device hardware and sensors

Most mobile devices have built-in sensors that measure orientation, motion, and various environmental conditions. All smartphones now are equipped with at least one camera. Many mobile apps use GPS and cameras as integral parts. In this practical exercise, students learn to manipulate a camera device either by using an existing Android Camera app or directly using Camera API as shown in figure 3.5. Students also learn to use other built-in sensors such as GPS, accelerator, speaker and microphone.

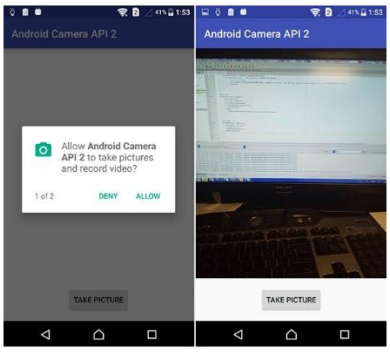


Figure 3.5: A practical lab to show how to use Android Camera2 API

3.6 Topic 6: Wireless connectivity

Wireless connectivity is an important topic in mobile app development. The Android platform allows apps to access wireless connections at a very low level. In this exercise, students learn about java.net.Socket and java.net.ServerSocket classes to build a chat app as shown in figure 3.6a.

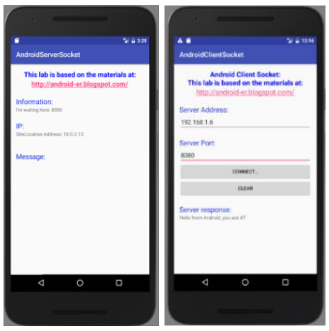


Figure 3.6a: Wireless connectivity: Server device (left) and Client device (right) using ServerSocket

They also learn to implement bi-direction Bluetooth communication between two Android devices by using BluetoothSocket class as shown in figure 3.6b.

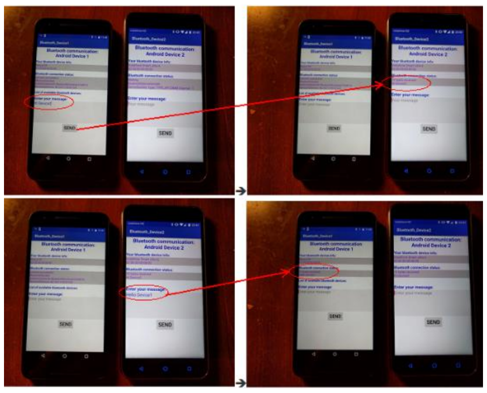


Figure 3.6.b: Bluetooth connectivity in Android.

3.7 Topic 7: Internal data storage

Most mobile apps require the ability to save information or data internally. In this lab, students learn to develop a “Note-taker” app using the Content Provider component - called Shared Preferences - as shown in figure 3.7a. This lab demonstrates how to develop an app that allows users to write new notes and then save them in the internal storage. Students can expand this exercise to a “to-do list” app.

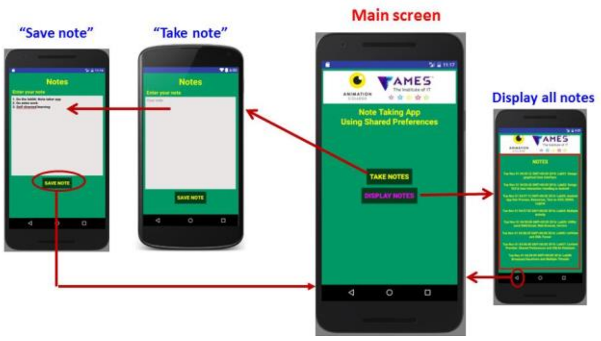


Figure 3.7a: Note-taker app using SharedPreferences

More advanced, students learn to use open-source SQLite database to store internal data such as student ID, student name, student email address, student lab journal submission status (1: Submitted, 0: Not Submitted) using a built-in database in the Android as shown in figure 3.7.b.

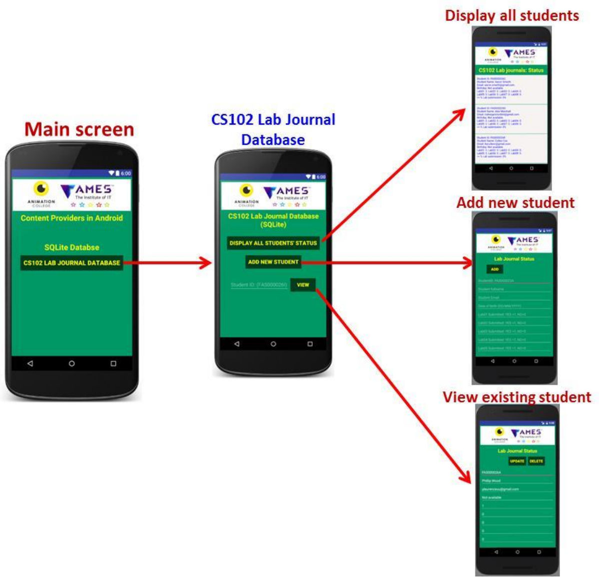
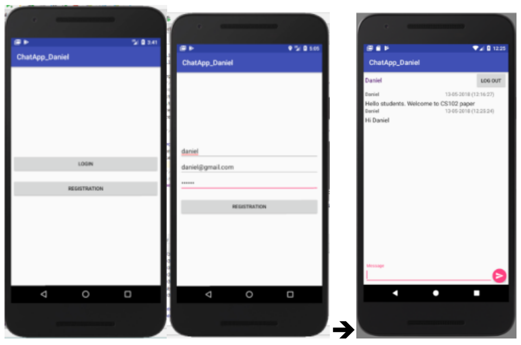

Display all students
Main screen
CS102 Lab Journal Database
Content Providers in Android
SQLite Databse
CS102 LAB JOURNAL DATABASE
CS102 Lab Journal Database (SQLite)
DISPLAY ALL STUDENTS' STATUS
ADD NEW STUDENT
VIEW
Add new student
View existing student
UPDATE
DELETE

Figure 3.7b: An Android app using SQLite database to store data

3.8 Topic 8: Real-time database

Mobile apps need a back-end server in order to perform tasks such as authenticating users and synchronizing user data across multiple devices. In this practical, students learn Google's Firebase to create a group chat app (a real-time social app) as shown in figure 3.8. This exercise is a simple demonstration with just one chat room, which is open to all users. Through this lab, students are able to use Firebase's real-time database for authentication function, real-time data storage, and real-time responsive app. Based on this practical exercise, students are able to develop other mobile apps similar to SnapChat, Whatsapp, Viber, Tinder, Skype, Uber, etc.



Daniel 13-05-2018 (12:16:27)
Hello students. Welcome to CS102 paper
Daniel 13-05-2018 (12:25:24)
Hi Daniel

Figure 3.8: Use a real-time database (Firebase) to develop a chatroom.

4. DISCUSSION AND CONCLUSION

Many educational institutions in NZ are increasingly interested in mobile app development and integration in their programming-related curriculum. However, a challenge is to select the best framework and development tool to use in the practical labs amongst a plenty of frameworks available. In this paper, we discussed and compared the available frameworks in detailed: Android Studio (Java/Kotlin), Xcode (Swift), Flutter (Dart), React Native (JavaScript), Ionic (HTML5, CSS3, TypeScript), Xamarin (C#), Cordova (HTML5, CSS3, JavaScript), and Appcelerator (JavaScript). We also investigated other institutions to see what development tools they used in practical labs. Finally, we suggested that Android Studio and Flutter framework are ideal choices.

What topics should be taught in the mobile application development course particularly in practical labs? To the best of the current author's knowledge, there is little research that addresses this question. We proposed 8 topics along with specific practical exercises - *(1) User Interface Design, (2) Event Handling, (3) Multi-threading Handling, (4) Graphics and Animation, (5) Use of Mobile Sensor/Components, (6) Wireless Connectivity (WiFi, Bluetooth), (7) Internal Data Storage, and (8) Real-time Database (Firebase)* – in order to provide students with enough technical skills and knowledge to create any type of apps at their disposition.

5. REFERENCES

Allan, A. (2013). *Learning iOS Programming, 3rd Edition from Xcode to App Store*. O'Reilly Media Publisher.

El-Tawab, S., Iskandarova, S., Almalag, M. & Ghazizadeh, P. (2018). A Methodology of Teaching Mobile Development for Undergraduate Students in Project-Based Classes. *Proceedings of Society for Information Technology & Teacher Education International Conference*. pp. 749-754.

Esakia, A. & McCrickard, D. S. (2016). An adaptable model for teaching mobile app development. *IEEE Frontier in Education Conference (FIE)*.

Gordon, A. J. (2013). Concepts for mobile programming. *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pp. 58-63.

Iversen, J. & Eierman, M. (2014). *Learning Mobile App Development: A Hands-on Guide to Building Apps with iOS and Android*. Addison-Wesley Professional Publisher.

Khmelevsky, Y. & Voytenko, V. (2016). A new paradigm for teaching mobile application development. *Proceedings of the 21st Western Canadian Conference on Computing Education*, pp. 8.

Mahmoud, Q. H. (2008). Integrating mobile devices into the computer science curriculum. *Frontiers in Education Conference 2008*. 38th Annual, pp. S3E-17.

Minaie, A., Sanati-Mehrizy, P., Sanati-Mehrizy, A., Sanati-Mehrizy, R. (2011). Integration of mobile devices into computer science and engineering curriculum. *Proc. ASEE Annual Conference & Exposition*.

Muyan-Özçelik, P. (2017). A hands-on cross-platform mobile programming approach to teaching OOP concepts and design patterns. *Software Engineering Curricula for Millennial (SECM) 2017 IEEE/ACM 1st International Workshop*, pp. 33-39.

Seyam, M.; McCrickard, D. S; Niu, S.; Esakia, A.; Kim, W. (2016). Teaching mobile application development through lectures, interactive tutorials, and Pair Programming, *IEEE Frontier in Education Conference (FIE)*.