MVVM

by M Syakir Arif

Submission date: 11-Mar-2020 07:14AM (UTC+0530)
Submission ID: 1273344905
File name: FULL_PAPER_MuhammadSyakirArif_ICSE_2019.pdf (863.26K)
Word count: 3379
Character count: 18881

Implementation of Model-View-ViewModel (MVVM) Architecture Pattern in the Sistem Informasi Akademik UNIDA Gontor Mobile Application

Article · November 2019

CITATIONS
0

READS
4

3 authors:

Muhammad Syakir Arif
University of Darussalam Gontor
5 PUBLICATIONS 0 CITATIONS
SEE PROFILE

Aziz Musthafa
University of Darussalam Gontor
6 PUBLICATIONS 1 CITATION
SEE PROFILE

Agus Muriyatmoko
University of Darussalam Gontor
20 PUBLICATIONS 4 CITATIONS
SEE PROFILE

Implementation of Model-View-ViewModel (MVVM) Architecture Pattern in the Sistem Informasi Akademik UNIDA Gontor Mobile Application

Muhammad Syakir A[1], Aziz Musthafa[2], Dihin Muriyatmoko[3]
Department of Informatics Engineering, Faculty of Science and Technology, University of Darussalam Gontor,
Jl. Raya Siman KM 5 Ponorogo 63471, Indonesia. Tel. +62-352-483762.
Email: muhammad.syakir@unida.gontor.ac.id[1], aziz@unida.gontor.ac.id[2], dihin@unida.gontor.ac.id[3]

Abstract. Software architecture primarily tuned for moderating the rising software complexities and changes. Model-View-ViewModel (MVVM) is a software architectural pattern that facilitates a separation of development of the graphical user interface from the development of the back-end logic (the data model). University of Darussalam Gontor (UNIDA Gontor) is having an Academic System called *Sistem Informasi Akademik* (SIAKAD) which supports the management of its college's data administration. Currently, the SIAKAD UNIDA Gontor's performance is still far from optimal, mainly because some of its features aren't user-friendly yet, especially when the SIAKAD UNIDA Gontor accessed from the mobile devices. Therefore, this research aims to implement the MVVM architecture pattern in the SIAKAD UNIDA Gontor mobile application, to increase its user-friendly aspect, especially for Android device users. This research is carried out with Waterfall development method, using Kotlin programming language and utilising Android Jetpack. Results from the app testing with Black Box method show that the application is running well and have no error. Next, results from the questionnaire distributed to the users (students, lecturers and BAAK staffs) show that the application run well and satisfying. From those trial results, show that the implementation of MVVM on SIAKAD UNIDA Gontor Android application has been successful according to the scenario and ready to be applied in the even semester of 2019/2020 campus' academic year and so on (https://play.google.com/store/apps/details?id=com.amoled.sidago). Further research and development, can be directed to the addition of new features and can support another mobile operating system, such as iOS, Tizen, Harmony OS, etc.

Keywords: Android Jetpack, Kotlin, MVVM, Sistem Informasi Akademik, UNIDA Gontor

Running title: Implementation of MVVM in SIAKAD Application

INTRODUCTION

Smartphones are very sophisticated media in accessing information and data services; this enables all areas of human life to be done more easily with the help of smartphones (Nuari, 2014). Therefore, according to the statistical data reported by Katadata states that in 2019, smartphone users in Indonesia will reach 92 million users. (Katadata, 2016).

Android, as one of the open-source smartphone operating systems, always achieves greater achievements every year. According to Google's CEO, Sundar Pichai at the 2017 Google I/O conference, he stated that at that time, Android had reached 2 billion monthly active users worldwide (Popper, 2017). While in Indonesia, statistics show that shares for the Android mobile operating system are also increasing every month, recorded at 88.37% in December 2017 ("Market share of mobile operating systems in Indonesia from January 2012 to December 2017," 2018).

Software systems have become very complicated and sophisticated to meet the demands of newer businesses. Software architecture is perfect for overcoming complexity and changing software (Raj, Raman, & Subramanian, 2017). Architectural patterns are well-known patterns as solutions to solve software architecture problems. The architectural pattern of software is the 'organisation' of the code as a whole (Tiari, 2015). Model-View-View Model (MVVM) is one of the software architectural patterns that carry a separation of graphical user interfaces from business logic processes or back-end logic. (Wikipedia contributors, n.d.)(Saleh, 2017).

University of Darussalam Gontor (UNIDA Gontor) is one of the Higher Education Institutions in Indonesia located in Ponorogo Regency, East Java. As one of the supports in the management of lecture data administration, UNIDA Gontor has an Academic System (SIAKAD) which is managed by the Pusat Pelayanan Teknologi Informasi dan Komunikasi (PPTIK). At present, SIAKAD cannot be accessed in a user-friendly manner when using a mobile device. Based on these constraints, up to now, the use of SIAKAD has not been considered optimal.

This study aims to implement the MVVM architecture pattern in the development of SIAKAD UNIDA Gontor application. Implementation is carried out using the Kotlin programming language and utilising Android Jetpack as an Android Architecture Component. The application is expected to provide safe, fast and easy SIAKAD services so that the products of this study are in harmony with Maqashid Shari'ah.

MATERIALS AND METHODS

Materials

The integrated development environment that used is the Android Studio IDE, with Kotlin programming language to handle back-end logic and XML for the front-end layout. Then the data source is from UNIDA Gontor's Application Programming Interface (API), as the main source of data retrieval. Then for local data sources using Room, which is an Android architecture component for the Android application database based on SQLite. Local data source functions as a temporary container of data taken from the API. MVVM architecture pattern implementation will be done by utilising Android Jetpack which has Android Architecture Components, such as Data Binding, LiveData, ViewModel, Room and Navigation.

Methods

The author used the Waterfall design method, which consisted of five stages as explained below:

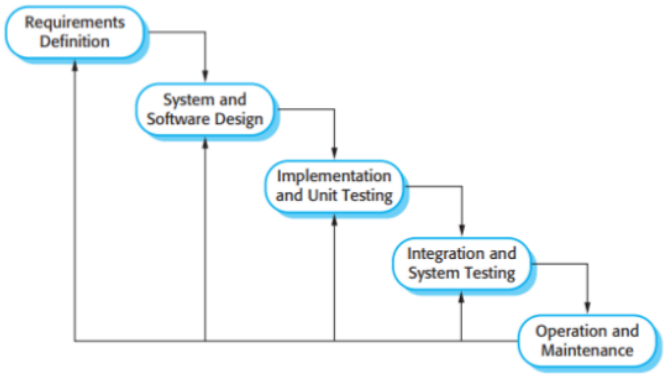

Figure 1. The phase of the Waterfall model (Sommerville, 2011)

Requirements Definition

Application users consist of the Student, Lecturer and Public categories. Therefore, each category has its own system requirements. The needs of each user category outlined in the following tables:

Table 1. System requirements for the Public user category

No	System Requirements
1	Users can view campus news
2	Users can view campus profiles, facilities, etc.

Table 2. System requirements for the Students user category

No	System Requirements
1	Students can log into the application
2	Students can make Study Plan Cards (KRS)
3	Students can confirm payment
4	Students can see a list of courses
5	Students can see course schedules
6	Students can see the Study Result Card (KHS)
7	Students can see their tuition bills

Table 3. System requirements for the Lecturers user category

No	System Requirements
1	Lecturers can log into the application
2	Academic Supervisor Lecturers (Dosen PA) can see the list of guidance student names
3	Academic Supervisor Lecturers (Dosen PA) can receive student's KRS reports

4	Academic Supervisor Lecturers (Dosen PA) can approve student KRS reports
5	Academic Supervisor Lecturers (Dosen PA) can permit changes in student KRS
6	Lecturers can see class schedules
7	Lecturers can see the value of student guidance

System and Software Design

Following are the Use Case Diagrams for the features in the Gontor SIAKAD UNIDA application:

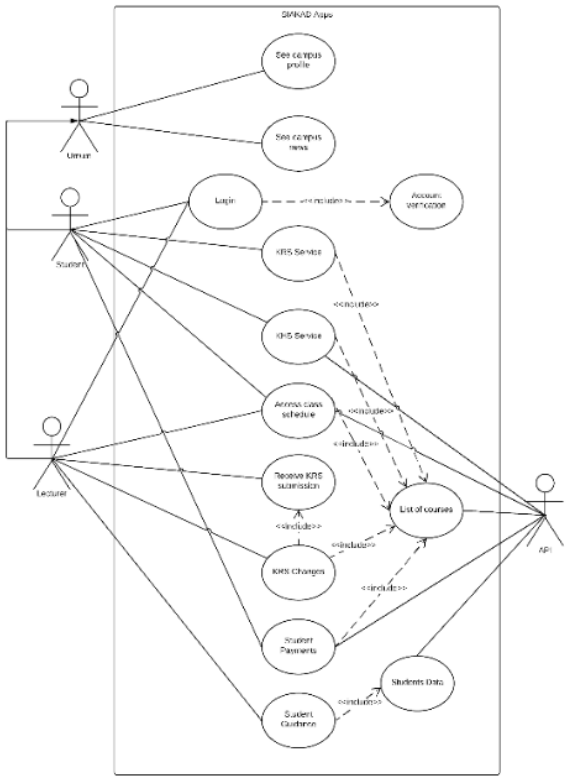

Figure 2. Use Case Diagram of the SIAKAD UNIDA Gontor application

Based on Figure 2, it can be seen that the Student and Lecturer users are derived from General users, where all the Public users can access some common features such as the news menu and campus profile. Then for features that can be accessed, each category of users has been shown in the diagram above, according to their respective categories. Furthermore, an explanation of each use case shown in Figure 2 is as follows:

Table 4. Use Case identification

No	Use Case Name	Description	Actor
1	See campus profile	Actors can view UNIDA Gontor's campus profile information	Public, Students, Lecturers
2	See campus news	Actors can view the UNIDA Gontor campus news	Public, Students, Lecturers
3	Login	Actors can enter into the SIAKAD account using the specified account	Students, Lecturers
4	Account verification	Account verification process so that actors can enter into the SIAKAD account	Students, Lecturers
5	KRS Service	Actors can access KRS service features	Students, API
6	KHS Service	Actors can access KHS service features	Students, API

7	List of courses	Course list data retrieval process	API
8	Access class schedules	Class schedule data retrieval process	Students, Lecturers, API
9	Receive KRS submissions	Lecturers can receive a list of KRS submissions from their guidance students.	Students, Lecturers, API
10	KRS Changes	Lecturers can permit to change the KRS data	Students, Lecturers, API
11	Student Payments	Actors can see a list of payment or tuition bills	Students, API
12	Student Guidance	Actors can monitor student guidance	Lecturers, API

Because this research will apply the MVVM architecture pattern, the following is the architectural design that will be applied in the SIAKAD UNIDA Gontor application:

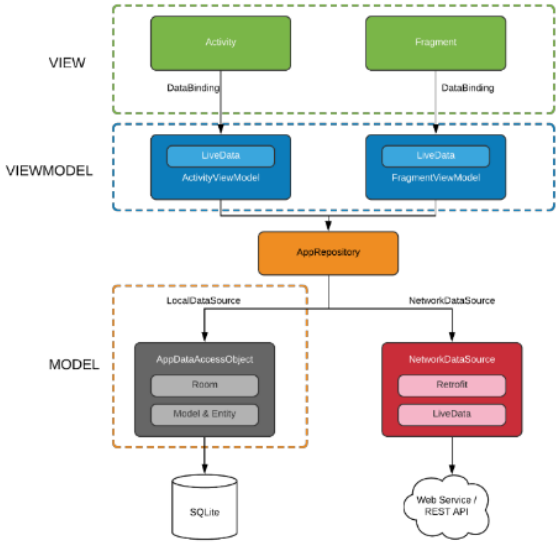

Figure 3. The application architecture design implements MVVM

In Figure 3 above, it is explained that the View category includes Activity and Fragment. Then for the ViewModel category, it includes the ActivityViewModel and FragmentViewModel classes. Both of these classes implement (inherit) LiveData libraries that handle the presentation of data to be displayed. The DataBinding library links the View and ViewModel components.

Then there is the AppRepository class which functions to handle connections to the database used by the application, both sourced from local data sources and network data sources.

Furthermore, the Model category includes classes that will handle data management in the local data source consisting of the Room library and the Model class (many Model classes are adapted to the data to be processed). Local data source uses SQLite storage.

Next, there is the NetworkDataSource class which handles data fetching from the API or from the local database.

4 *Implementation and Unit Testing*

For the Implementation phase, the MVVM architecture pattern is implemented when the author programs the SIAKAD UNIDA Gontor application. Then for Unit Testing, the author conducts trials using the Black Box method.

4 *Integration and System Testing*

For the Integration and System Testing phase, the author tests the application with respondents' representatives from the BAAK Staff, Lecturers and Students. The results of the trial were evaluated using questionnaire data from the respondents.

Operation and Maintenance

At this phase, the application is improved based on suggestion and evaluation from the user or supervisor, then updates the application. User evaluations can be taken based on trial results as well as assessments provided on the Google Play Store (if the application has been published).

RESULTS AND DISCUSSION

This chapter discusses research methods for the Implementation and Unit Testing (the third phase of Waterfall), Integration and System Testing (the fourth phase of Waterfall), and Operation and Maintenance (the fifth phase of Waterfall). Regarding the results and discussion, the author will focus only on discussing the results of the application of MVVM on the KRS feature. All features in the application have implemented MVVM, which is the same concept as the KRS feature.

Implementation and Unit Testing

For the sake of the neatness of the package structure and making it easier to place classes, the author groups the classes in the project as follows:

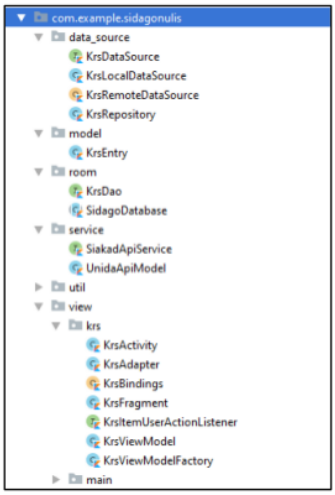

Figure 4. The class hierarchy in the project for KRS features

Figure 4 shows the classes used to treat KRS features with MVVM architecture pattern. Meanwhile, the following is an illustration showing the KRS Model to View through ViewModel intermediaries:

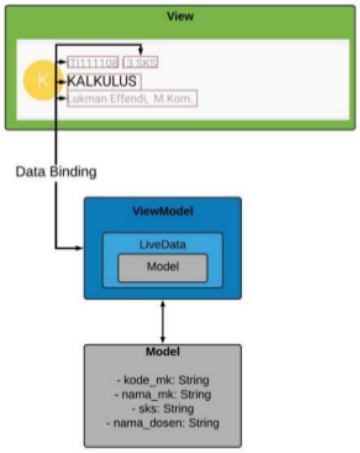

Figure 5. Illustration that is showing KRS Models to View through ViewModel

In Figure 5, it is shown that the KRS Model, which contains data `kode_mk`, `nama_mk`, `sks`, and `nama_dosen` is displayed to View with LiveData intermediaries in ViewModel. The use of Data Binding is very instrumental in this

process. Data Binding can directly respond to data changes that occur in LiveData in ViewModel, then display it in the User Interface (View). Here is an illustration:

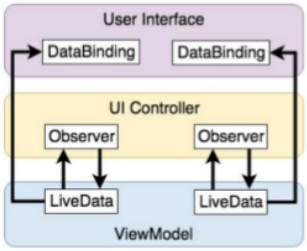

Figure 6. Illustration of the relationship between LiveData in ViewModel with Data Binding

Then, the results of the implementation of the MVVM architecture pattern are presented in the following illustration diagram:

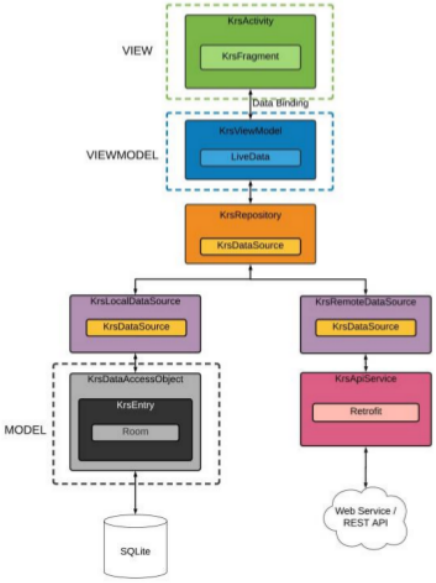

Figure 7. The illustration of MVVM architecture that was successfully applied, one of them is in the student KRS feature

From Figure 7, it can be explained that to implement MVVM in the KRS feature, it takes a hierarchy between classes as illustrated. For the View components, it is consists of `KrsActivity` and `KrsFragment`. Then for the ViewModel component, named `KrsViewModel`. Data Binding plays a role in channelling data between the View and ViewModel components. Then, the Repository class is tasked to manage two database sources in the application, namely `LocalDataSource` and `RemoteDataSource`. `RemoteDataSource` is responsible for retrieving KRS data from the API. Meanwhile, `LocalDataSource` is responsible for storing data from the API to the SQLite Database and sending the data if at any time requested by the `Repository` class. This will be useful when the application is not connected to the internet, so the KRS data that is stored locally will be displayed to the user.

Then for Unit Testing results, the author uses the Black Box method and states that all application features function well and there is no error or force close. Unit Testing is carried out using the latest version of Android 10 (Beta). Here are some examples of the display of the SIAKAD UNIDA Gontor application:

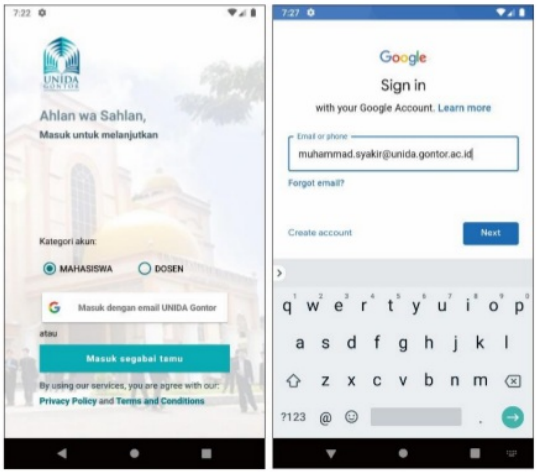

Figure 8. The login page uses the UNIDA Gontor's email

As shown in Figure 8, the user will first select the account category. Then after that, pressing the button '*Masuk dengan email UNIDA Gontor'* then the user will be asked to fill in the email data along with the password. Google Services handle the login process to the UNIDA Gontor's email account. Google account authentication services can be integrated with the application SIAKAD UNIDA Gontor as well.

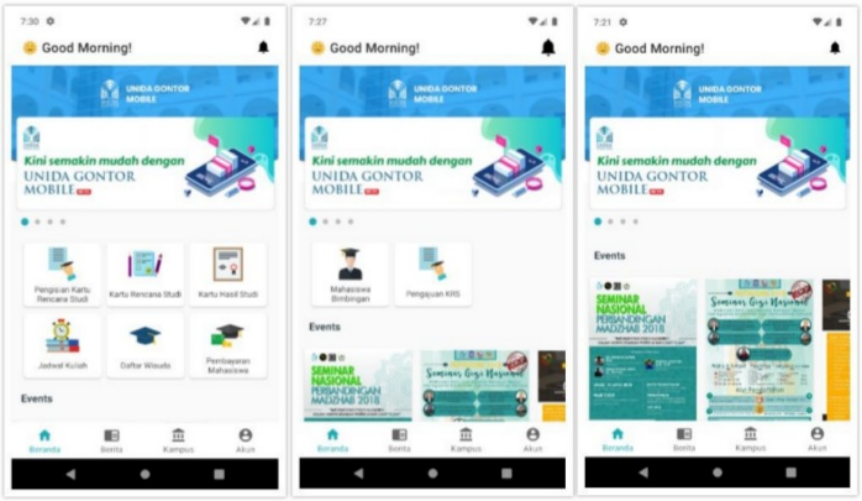

Figure 9. Comparison of the homepage interfaces for each category of Students, Lecturers and Public.

In Figure 9, it is shown that the home page for the student category includes KRS Submission menus, Study Plan Cards (KRS), Study Result Cards (KHS), Lecture Schedules, Graduation Lists and Student Payments. While the home page for the lecturer category contains a menu of Student Guidance and KRS Submission.

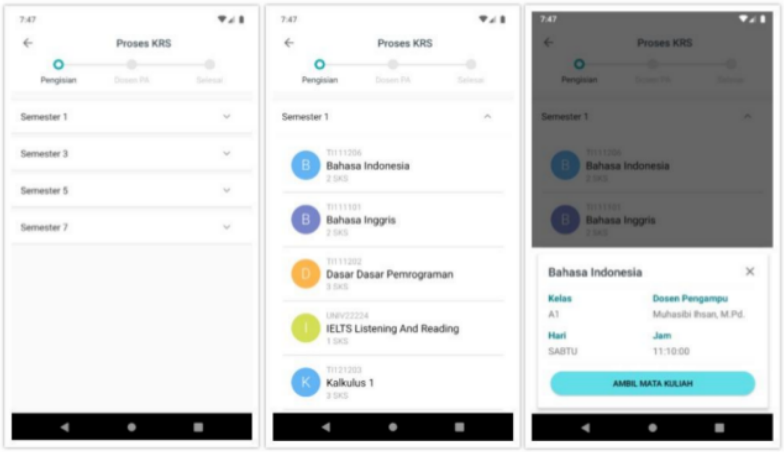

Figure 10. The Process of Filling KRS by Students

In Figure 10, the process of filling KRS by students is shown. The left-side figure shows the dropdown menu for each semester. Then the middle-side figure shows the list of courses in the related semester. Then in the right-side figure shows the details of the selected course.

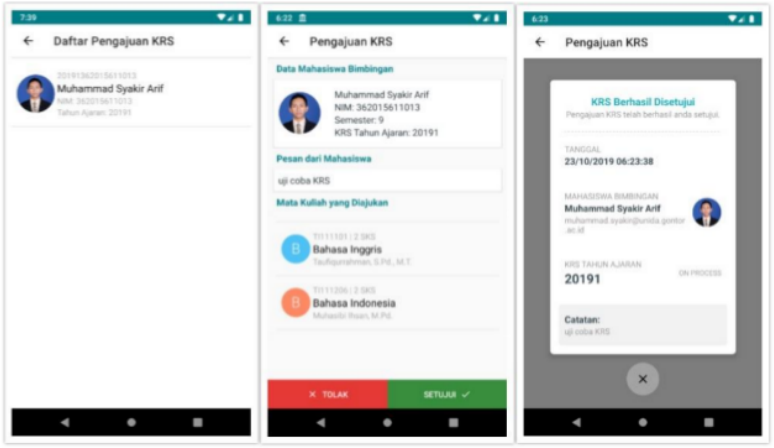

Figure 11. Display interface of KRS Submission Confirmation process by Lecturer

Figure 11 shows the process of KRS Submission Confirmation by the related Student Advisor. In the left-side figure, the lecturer looks at the list of KRS submissions from the student's guidance. Then in the middle-side figure, the KRS details of the student are displayed, then lecturer can approve or reject the submission. Then in the right-side figure shows the KRS information successfully approved or rejected.

Integration and System Testing

For this phase, the author tests the application to the user. The users are representatives of the BAAK Staff, Lecturers and Students of UNIDA Gontor who are willing to become respondents.

Application Testing by BAAK Staff

This test aims to ensure that the system integration in the application of SIAKAD UNIDA Gontor is matching the Standard Operational Procedure (SOP) set by the Academic and Student Administration Bureau (BAAK). This test was carried out by Deputy Head of BAAK UNIDA Gontor named Al-Ustadz Samsirin, M.Pd.I. The features tested are KRS Features and KHS Features. From these trials, the following questionnaire results were obtained:

No	Indicator	Question	Score
1	System Integration Aspects	The functions and features of the application are consistent with the purpose of the application	5
2		The flow of the KRS submission process in the application is matching the SOP	4
3		The flow of the process of seeing KHS in the application is matching the SOP set by BAAK UNIDA Gontor	4
4		The presented Course Data is matching with the data in SIAKAD	5
5		The presented Class Schedule is matching with the data in SIAKAD	5
6		The application runs smoothly, and no errors occur	4
7	Interfaces Interaction Aspects	The graphical interface is easy to recognise	3
8		The graphical interface is easy to remember	3
9		Posts in the application are easy to read	3
10		The application is easy to operate/use	4
11		The colour display of the application is comfortable to see and not boring	3
12		Easily access SIAKAD services through the application	4
Average			3,9166667

Note: The indicator values specified in the questionnaire are 5 = Strongly Agree, 4 = Agree, 3 = Normal, 2 = Disagree, 1 = Strongly Disagree.

Application Testing by Lecturers

The test was conducted with representative respondents from lecturers of Informatics Engineering UNIDA Gontor. The indicator values specified in the questionnaire are 5 = Strongly Agree, 4 = Agree, 3 = Normal, 2 = Disagree, 1 = Strongly Disagree. The following is an assessment diagram of the trial:

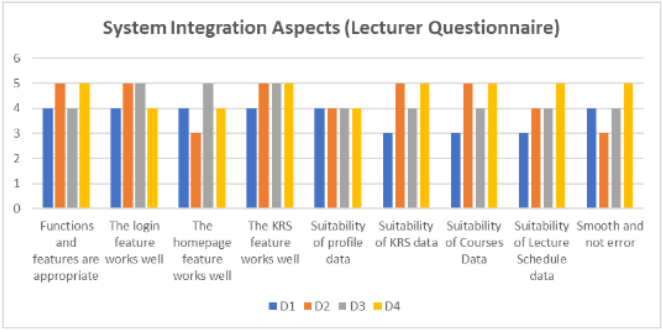

Figure 12. Lecturer Questionnaire Results for System Integration Aspects

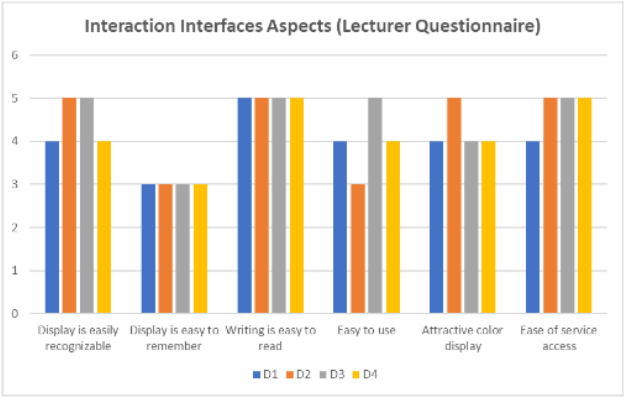

Figure 13. Lecturer Questionnaire Results for Interface Interaction Aspects

Application Testing by Students

The test was conducted with five respondents representing respondents from Informatics Engineering students at UNIDA Gontor. The indicator values specified in the questionnaire are 5 = Strongly Agree, 4 = Agree, 3 = Normal, 2 = Disagree, 1 = Strongly Disagree. The following is an assessment diagram of the trial:

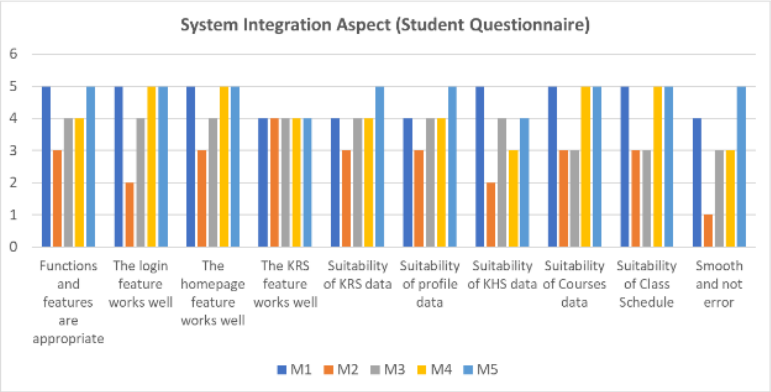

Figure 14. Student Questionnaire Results for System Integration Aspects

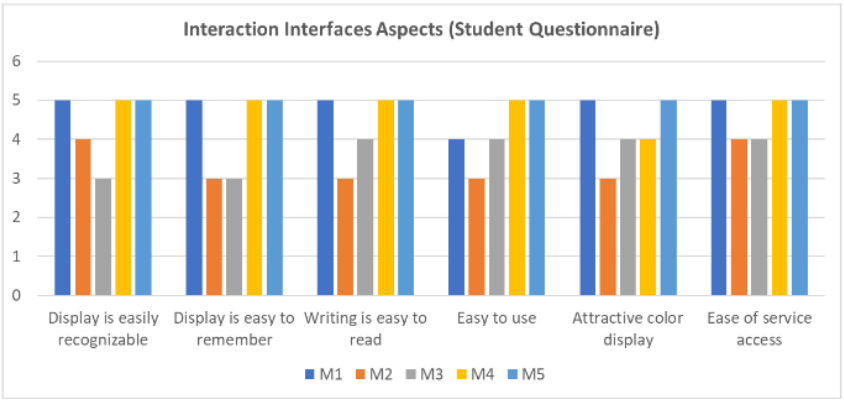

Figure 15. Student questionnaire results for Interface Interaction Aspects

Operation and Maintenance

Improvements in the System Integration Aspect

No	Improvements	Status
1	Please disable auto-login that caused by user history, when users sign in their account using email	Done
2	Improved data duplication of the KHS and KRS features	Done

Improvements in Interface Interactions Aspect

No	Improvements	Status
1	The User ID display does not need to be displayed	Done
2	*"NIY"* to be changed to *"Kode Dosen"*	Done
3	Set the EditText to non-editable	Done
4	In KRS feature, there are still duplications in certain subjects	Done
5	Give a confirmation dialogue when the lecturer approves/rejects KRS submission	Done

Discussion

The implementation of MVVM on SIAKAD UNIDA Gontor for Android mobile users in improving user-friendly has been successful. Black Box test results show that this application runs smoothly and without errors. The results of the questionnaire test showed that the implementation of MVVM was matching the scenario and received positive responses from students, lecturers and staff of BAAK UNIDA Gontor. This Android mobile application is ready to be applied from the 2019/2020 school year even semester onwards (https://play.google.com/store/apps/details?id=com.amoled.adago). Further development can be directed to the addition of new features. Also, it can be developed for another mobile operating system, such as iOS, Tizen, Harmony OS, etc

ACKNOWLEDGEMENTS

Our gratitude to the University of Darussalam Gontor Ponorogo for funding this research.

REFERENCE

Katadata. (2016). Pengguna Smartphone di Indonesia 2016-2019. Retrieved March 2, 2019, from https://databoks.katadata.co.id/datapublish/2016/08/08/pengguna-smartphone-di-indonesia-2016-2019

Market share of mobile operating systems in Indonesia from January 2012 to December 2017. (2018). Retrieved December 8, 2018, from https://www.statista.com/statistics/262205/market-share-held-by-mobile-operating-systems-in-indonesia/

Nuari, N. (2014). Perancangan Aplikasi Layanan Mobile Informasi Administrasi Akademik Berbasis Android Menggunankan Webservice (Studi Kasus Reg. B Universitas Tanjungpura). *Jurnal Sistem Dan Teknologi Informasi (JustIN)*, *1*, 1–7.

Popper, B. (2017). Google announces over 2 billion monthly active devices on Android. Retrieved December 8, 2018, from https://www.theverge.com/2017/5/17/15654454/android-reaches-2-billion-monthly-active-users

Saleh, H. (2017). MVVM architecture, ViewModel and LiveData Part 1. Retrieved December 6, 2018, from https://proandroiddev.com/mvvm-architecture-viewmodel-and-livedata-part-1-604f50cda1

Sommerville, I. (2011). *Software Engineering* (9th ed.). Massachusetts: Pearson.

Tiari, P. K. (2015). What's the difference between design patterns and architectural patterns? Retrieved December 9, 2018, from Stack Overflow website: https://stackoverflow.com/a/33757364

Wikipedia contributors. (n.d.). Model–view–viewmodel. Retrieved December 8, 2018, from https://en.wikipedia.org/w/index.php?title=Model–view–viewmodel&oldid=871113657

MVVM

ORIGINALITY REPORT

13%	9%	6%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.scribd.com Internet Source	3%
2	D Muriyatmoko, N A S Asy'ari, M S Arif. "Android Radio Streaming Apps for Songgolangit FM Ponorogo", Journal of Physics: Conference Series, 2019 Publication	1%
3	files.eric.ed.gov Internet Source	1%
4	Submitted to Informatics Education Limited Student Paper	1%
5	lup.lub.lu.se Internet Source	1%
6	Submitted to Sheffield Hallam University Student Paper	1%
7	Submitted to Curtin University of Technology Student Paper	1%
8	repository.bsi.ac.id Internet Source	1%

9	Submitted to University of Sheffield Student Paper	1%
10	www.statista.com Internet Source	<1%
11	docplayer.net Internet Source	<1%
12	en.wikipedia.org Internet Source	<1%
13	Submitted to University of Florida Student Paper	<1%
14	Submitted to Regis University Student Paper	<1%
15	Submitted to Softwarica College Of IT & E-Commerce Student Paper	<1%
16	jurnal.ubharajaya.ac.id Internet Source	<1%
17	Submitted to Aston University Student Paper	<1%
18	Submitted to Kingston University Student Paper	<1%
19	Submitted to CVC Nigeria Consortium Student Paper	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		