Tribhuvan University

Faculty of Humanities and Social Sciences

Mayalu

A PROJECT REPORT

Submitted to

Department of Computer Application

Aadim National College

In the partial fulfillment of the requirements for the Bachelors in Computer Application

Submitted By

Suprim Poudel (10591189)

18th June 2022

Under the Supervision of

Mr. Yuba Raj Kalathoki

Tribhuvan University

Faculty of Humanities and Social Sciences

Aadim National College

Supervisor's Recommendation

This is to certify that this project entitled, **Mayalu** prepared and submitted by Suprim Poudel in partial fulfilment of the requirements of the degree of Bachelor of Computer Application (BCA) awarded by Tribhuvan University, has been completed under my supervision. I recommend the same for acceptance by Tribhuvan University.

…….

SIGNATURE

Mr. Yuba Raj Kalathoki

SUPERVISOR

Chabahil, Chuchepati, Kathmandu 44600

Tribhuvan University

Faculty of Humanities and Social Sciences

Aadim National College

LETTER OF APPROVAL

This is to certify that this project is prepared by Suprim Poudel entitled "Mayalu" in partial fulfilment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

…… Mr. Yuba Raj Kalathoki Supervisor Chabahil, Chuchepati, Kathmandu 44600	…… Mr. Yuba Raj Kalathoki HOD Chabahil, Chuchepati, Kathmandu 44600
…… **Internal Examiner**	…… **External Examiner**

Abstract

Online Dating apps have become a popular means to meet potential partners. Today, there are numerous dating apps. In most of these, they present its user with pictures of people whom they can either like or dislike based on first impression. If two users like each other they are allowed to initiate a conversation via the chat feature.

I thought of integrating similar functionality Mayalu. This system covers most of the features that are available in modern day's dating apps such as Tinder.

This report includes the design and implementation of dating app system where different techniques are explored and compared alongside all the activities performed during the development of Mayalu during the extent of the project.

Acknowledgements

Firstly, we would like to give my gratitude towards my supervisor Mr. Yuba Raj Kalathoki for assisting and inspiring us on our project. Without his support and encouragement, it would have been difficult to work on.

I would also like to thank the staff members of Aadim National College, for their support and vision in making the project a systematic one.

I would also be thankful to our colleagues for their support and encouragement in the project, who have supported me through their guidelines and experience. I would also be thankful to all the faculty members, for their intense support in fulfilling my project requirements and also thankful to the library staff of Aadim National College for providing me with the necessary reference materials.

At the end, I would like to express our sincere thanks to all my friends and others who helped us directly or indirectly during this project period.

Table Of Contents

Abbreviations and Acronyms

API	Application Programming Interface
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
JVM	Java Virtual Machine
TDD	Test Driven Development
XML	Extensible Markup Language
MVC	Model View Controller
MVP	Model View Presenter
MVVM	Model View ViewModel
OMT	Object Modelling
RX	ReactiveX
SSD	System Sequence Diagram
UI	User Interaction
UX	User Experience

List of Figures

List of Tables

Chapter 1: Introduction

1. 1 Introduction

The Internet has brought the world into our desks and pockets via our laptops and mobiles. From shopping to dating almost everything can be done online now. Imagine meeting your imperfectly perfect match just by swiping right. Well, thanks to the internet, this is the reality now. Dating apps are now helping users to find their new romantic partners, renew older romances, strengthen social connections, and reinforce friendship bonds as well.

Generation Z has taken a huge interest in online dating because now, it is easier to find a person online whose interest matches with them. People don't have time to even have proper food in this fast world, so sometimes dating can be out of the question. Online dating on the other hand solves this problem, it is not time-consuming plus it also saves money and energy.

The craze for online dating has increased in such a way that reports claim 38% of relationships happened through online dating. In recent years it has become a widely accepted phenomenon. The free access to various features of dating apps to find potential partners is what makes it more appealing.

Plus, there is no boundary, one can be in the South Pole and will be able to find a partner who is residing in the North Pole, basically from any corner of the world.

1.2 Problem Statement

Finding a person that matches your interest can be quite difficult. Also, for shy and introverts, talking to people in person can be uncomfortable as compared to chatting. Another problem that peoples face while using dating apps today is the subscription fee. In order to even view the person that has matched your profile, we have to pay subscription fee for it. This is one of the major problems that people face while using top rated dating apps.

1.3 Objectives

- To create a mobile application which displays people based on your interest,
- To allow matched (similar interest) persons to connect and chat with each other,
- To let people, chat with each other without the need to pay a subscription fee,
- To alert people when they get a match

1.4 Scope and Limitations

1.4.1 Scope

- Allow similar interest people to connect with each other,
- Allow matched users to chat with each other.

1.4.2 Limitations

- Having to fetch all its data during real-time, the app requires an internet connection to run,
- The data and the photos that are entered by user during signup might be misleading and fake and there is no system made now in order to verify those data.

1.5 Development Methodology

Software development methodology is a process or series of processes used in software development. Again, quite broad but that it is things like a design phase, a development phase. It is ways of thinking about things like waterfall being a non-iterative kind of process. Generally, it takes the form of defined phases. It is designed to describe the how the life cycle of a piece of software. [1]

Being a simple and compact application, I have used Waterfall Model to develop my application Mayalu because it seemed the best model for my project. Waterfall model is a classical model used in SDLC which follows linear and sequential approach. Meaning one only moves forward onto the next phase once the completion of previous phase during SDLC.

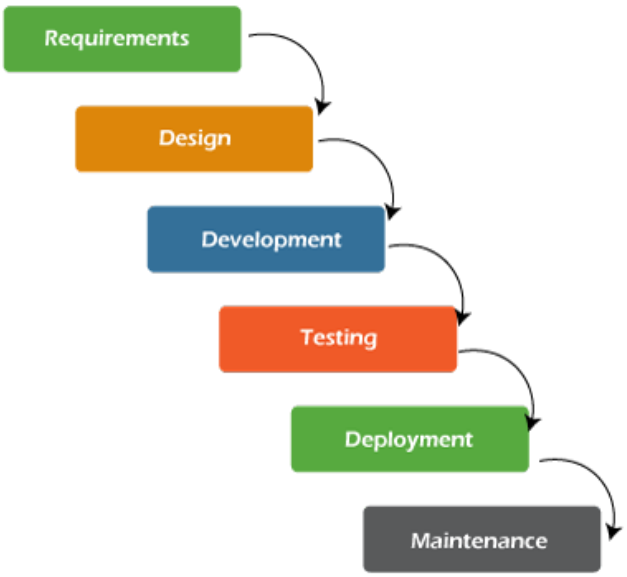

Figure 1 Waterfall Model [2]

1.6 Report Organization

This project report contains five chapters in total. Chapter 1 being Introduction which gives us a brief summary of the nature of project, its objectives, limitations, design methodology, etc. Chapter 2 will be explaining about Background Study and Literature Review, 3 explaining System Analysis and Design. Similarly, chapter 4 will be for Implementation and Testing and chapter 5 being Conclusion and Future Recommendation.

Chapter 2: Background Study and Literature Review

2.1 Background Study

In the last decade, the popularity of the Internet has increased a lot. Along with it, the people using smartphones has also increased extensively. But youths and adult using smartphones mostly, there has been an emergence of dating apps be it a real-time location-based or not. Dating apps like Tinder, Grindr have transformed traditional pathways of socialization and promoted new ways of meeting and relating to potential romantic or sexual partners.

For example, the Statista Market Forecast portal estimated that by the end of 2019, there were more than 200 million active users of dating apps worldwide. It has been noted that more than ten million people use Tinder daily, which has been downloaded more than a hundred million times worldwide. In addition, studies conducted in different geographical and cultural contexts have shown that around 40% of single adults are looking for an online partner, or that around 25% of new couples met through this means.

2.2 Literature Review

2.2.1 Study of existing system

Many dating applications already exists in today's world. But some of the most popular are Tinder, Tan Tan, Grindr, etc. Mayalu will somehow be similar to them. The main goal of Mayalu is to provide the best and user-friendly environment to the users than other application. To get a rough idea about how I want to build my application, I reviewed some of the well renowned applications

I. **Tinder**:

   Tinder is commonly referred to as the "hookup app," but at its core is a dating app that, like competitors, aims to offer a gateway to relationships, and even marriage, for a more tech-savvy generation.

   It upends traditional dating culture, which typically requires you to go out and interact with strangers in physical spaces. Instead, it brings that diverse dating pool that you may — or may not — have had access to at a bar or club straight to you. [3]

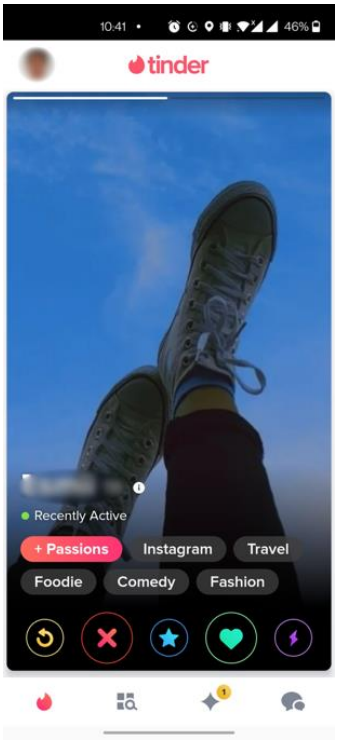

Figure 2 Swipe Page of Tinder

The above picture shows the landing page of Tinder after the user sets up his/ her profile.

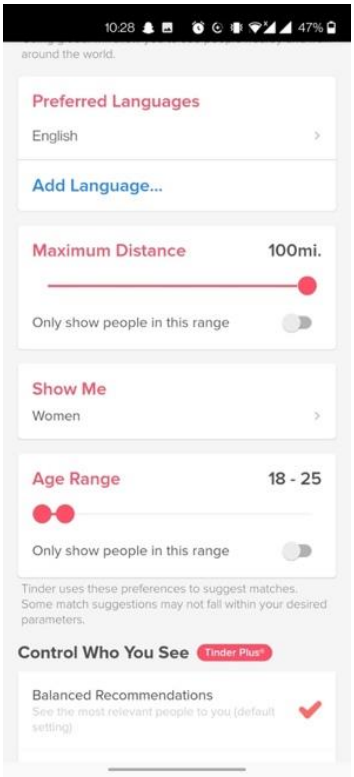

Figure 3 Edit Profile Page of Tinder

The above picture shows the settings page of Tinder where user can update his/ her profile

II. Hinge:

Hinge is a dating app which bills itself as the only dating app that emphasizes long-term connections between users over superficiality and building relationships. It seeks to attract a younger demographic than Match.com and eHarmony, such as the demographic using Tinder. [4]

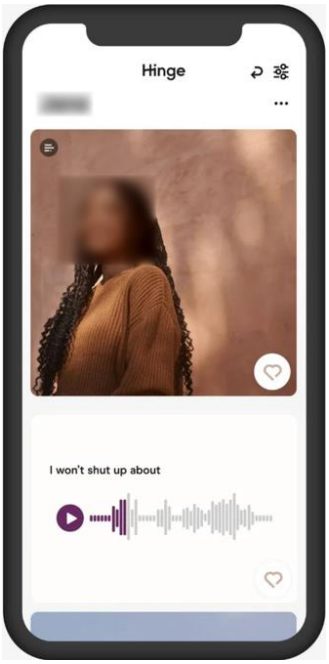

Figure 4 Home Page of Hinge

Chapter 3: System Analysis and Design

3.1 System Analysis

Before starting any new system/ software development, it is important to plan how it will be developed, tested and maintained. It is a key to success for any project. For the success of the project, I along with my supervisor were involved in the discussions regarding the development and work flow of the project.

3.1.1 Requirement Analysis

For any system, there are functional and non-functional requirements to be considered while determining the requirements of the system. The functional requirements are user's visible features that are typically initiated by stakeholders of the system, On the other hand, non-functional requirements are requirements that describe how the system will do what it is supposed to do.

I. Functional Requirement:

- User
  - Can login/ register
  - Can view user's profile
  - Can like/ unlike a user
  - Can match with a user
  - Can chat with matched user
- Admin
  - Can view user's profile
  - Can manage all users
  - Can send promotional notifications to all users

The below picture shows us the use case diagram of our application. The two main users of the application are User, and Admin. A user can login/ register into the application, view a user's profile, like and unlike a user and much more. Whereas Admin can manage all users, send notifications to all other users, etc.

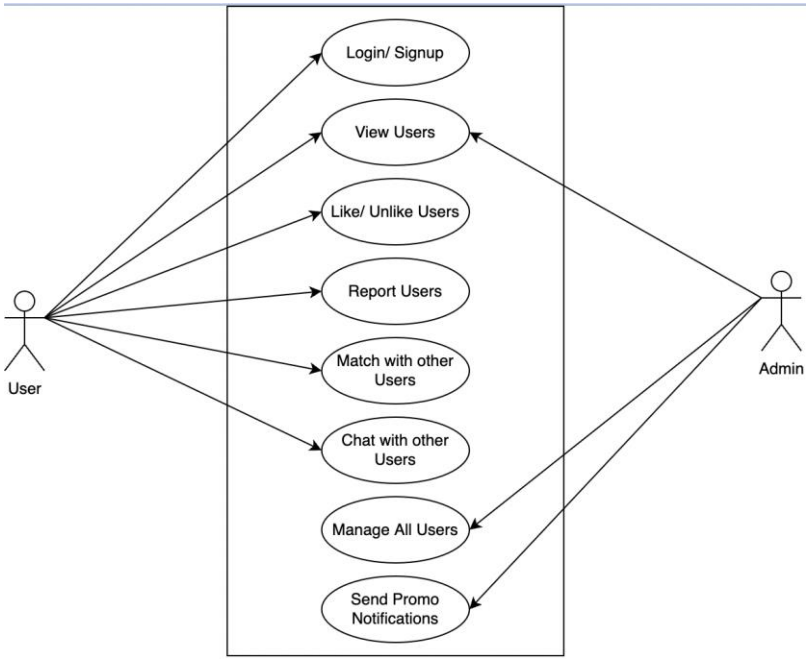

Figure 5 Use Case Diagram

II. Non-Functional Requirements:

- Security: The application can only be accessed by registered and validated user using their email address and password. Also, the system uses Firebase Authentication for better security.
- Reliability: The application is reliable as it has considered the damages that can be caused by incomplete data.
- Usability: Mayalu provides an appropriate output, when necessary, features in the correct format are given as input
- Maintainability: The application is trained and tested according to the correct set of features provided and the data is maintained in appropriate format.
- Availability: The application is made accessible and available anywhere and anytime. All you need is an android mobile device and a good internet connection.

3.1.2 Feasibility Analysis

As the name implies, a feasibility analysis is used to determine the viability of an idea, such as ensuring a project is legally and technically feasible as well as economically justifiable. It tells us

whether a project is worth the investment. Following feasibility analysis were performed prior to working on the project

I. **Technical Feasibility**: All the software development tools required are readily available so it is technically feasible.

II. **Operational Feasibility**: All the required operations such as internet service are available and there isn't any legal issue. So, this project can be considered operationally feasible.

III. **Economic Feasibility**: This project is developed using software selling kits of which all re open source, also the database used for this project i.e., Firebase also provides free plan so this project is economically feasible and can be implemented easily.

IV. **Schedule Feasibility**: In scheduling feasibility, an organization estimated how much time the project will take to complete. To calculate and continually re-examine whether it is possible to complete all the amount and scope of work lying ahead, utilizing the given number of resources, within the required period of time.

3.1.3 Object Modelling

Object modelling (OMT) is an approach for software modelling and designing, it is used to test physical entities before building them. OMT is used for visualization and reduction of complexity. OMT can be categorized into Object Model, Dynamic Model, and Functional Model.

I. **Class Diagram**: Class diagram is also called a static diagram because it represents the static view of an application. In other words, class diagram is used for visualizing, describing, and documenting different aspects of a system. It contains of attributes and operations of a class and also the constraints imposed on the system.

The below diagram shows some of the classes and their interrelationship with each other. Whenever we create a model of a class, its divided into three sections; Class Name, Attributes, and their Methods. In order to represent a class's attributes and method, we follow following approach. + for Public Access Modifier, - for Private Access Modifier and # for Protected Access Modifier The relationship between classes is represented by crowfoot's notation.

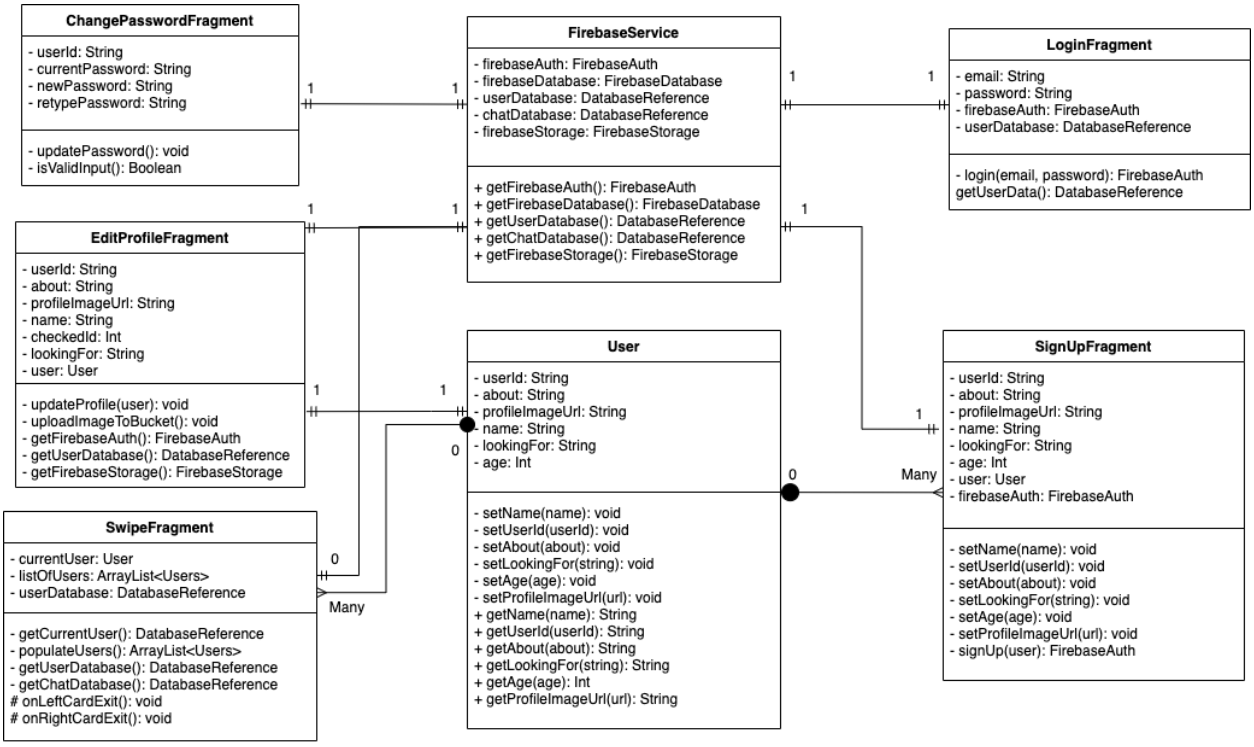

Figure 6 Class Diagram

II. Object Diagram: Object diagram are dependent on class diagram because they are derived from it. It represents an instance of class diagram. Object diagram represents a snapshot of the system at a particular moment.

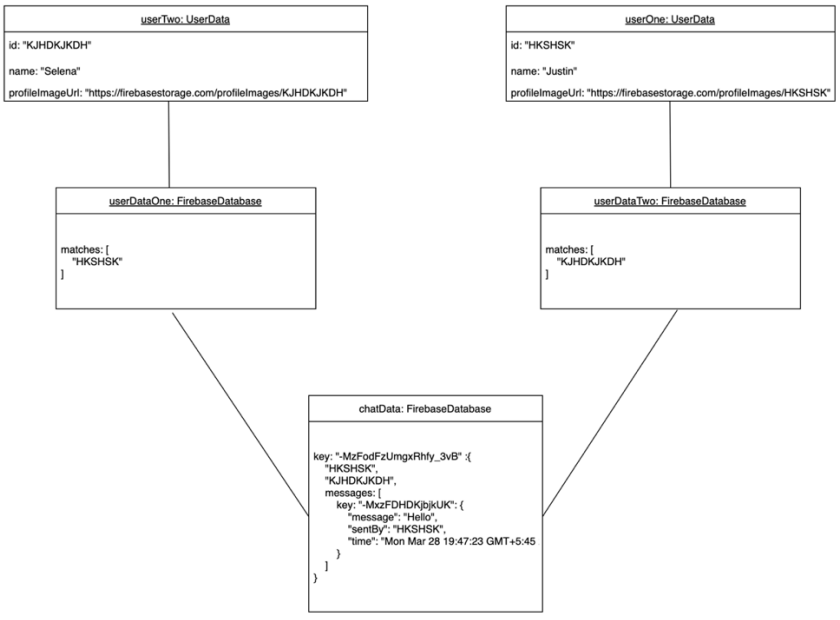

Figure 7 Object Diagram

The above diagram shows how the state of the system flows when any two users are matched and how are they then stored and creates a new object for Chat is created at a point in time.

3.1.4 Dynamic Modelling

Dynamic Modelling describes those aspects of the system that are concerned with time and sequencing of the operations. It is used to specify and implement the control aspect of the system [5].Dynamic model is represented graphically with the help of state and sequence diagram.

I. **State Diagram**: State diagram is used to visualize the entire life cycle of object and provides a user better understanding of state-based diagram. State diagram can be defined as the graphical representation of behavioral model which consist of states, state transitions and actions.

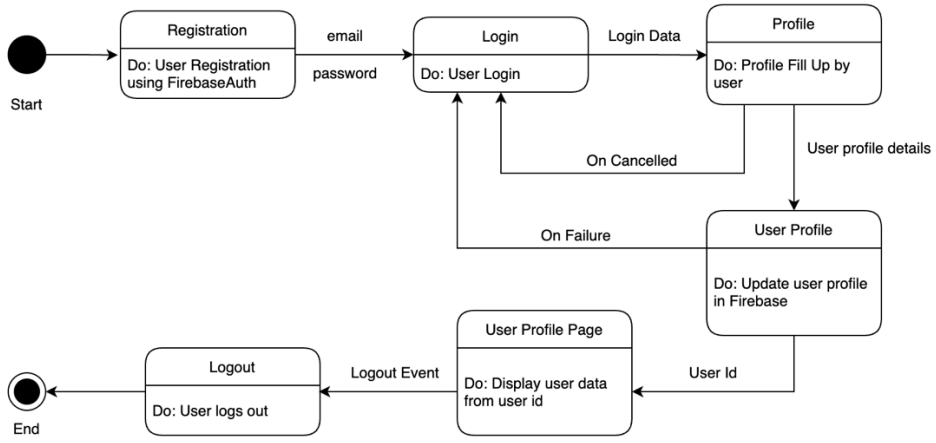

Figure 8 State Diagram

The above state diagram, explains the transition from when a user logs in and until he/ she logs out of the system. It also shows how a state is being transitioned to another state with the help of events such as Do Login, Do Logout, etc.

II. **Sequence Diagram**: Sequence diagram is an interaction diagram that shows how operations are carried out and what messages are received and sent. It is also called as System Sequence Diagram (SSD). SSD focus on lifelines, or the processes and the object that lives simultaneously.

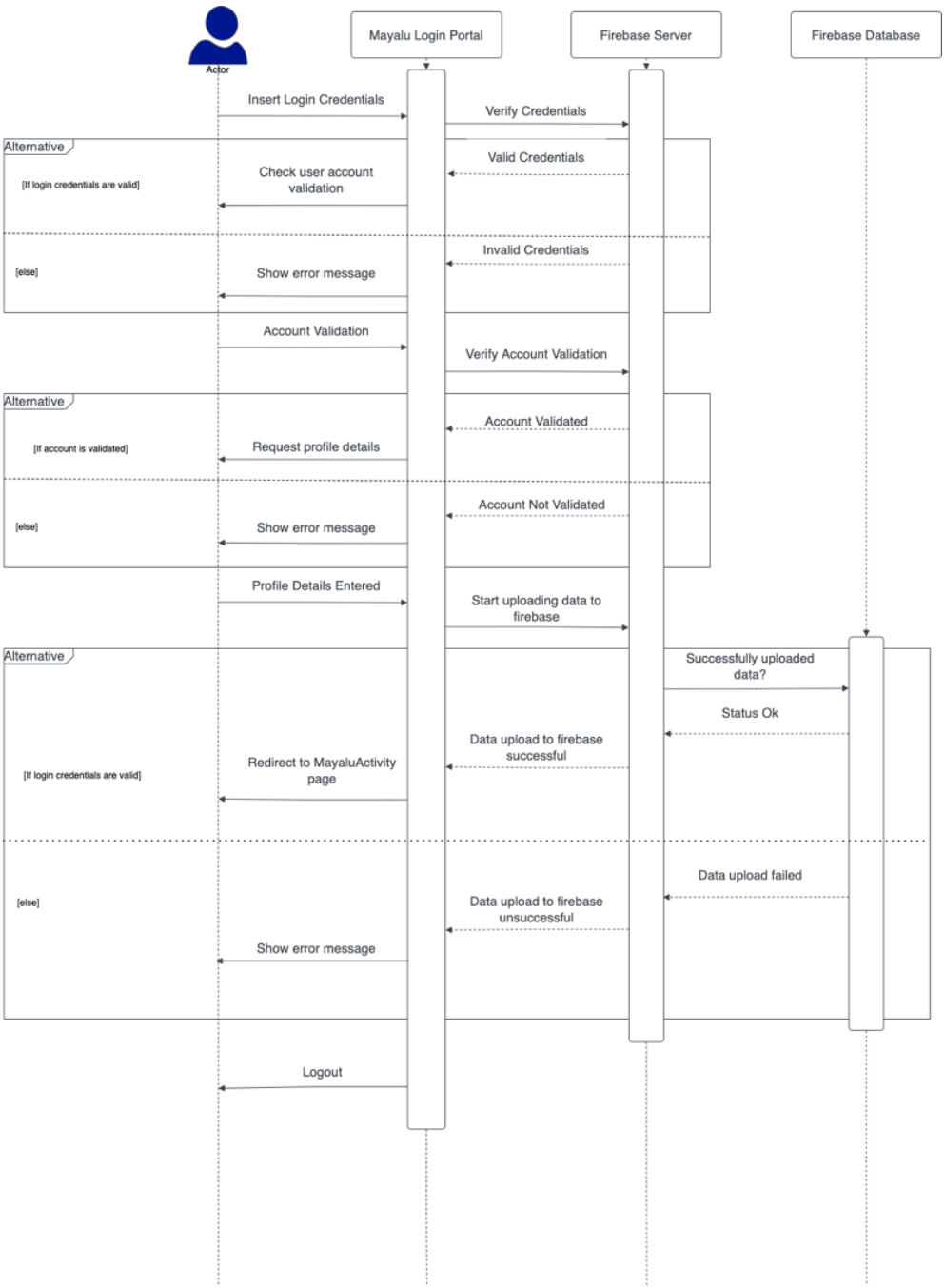

Figure 9 Sequence Diagram

The above sequence diagram explains about the login and logout process of user. It explains the order of those process. Once the user logs in, his/ her data is sent to the firebase database to check if he/ she has uploaded his/ her personal details. After that is check actions are done accordingly.

3.1.5 Process Modelling

Process modeling is the graphical representation of business processes or workflows. Like a flow chart, individual steps of the process are drawn out so there is an end-to-end overview of the tasks in the process within the context of the business environment. [6]

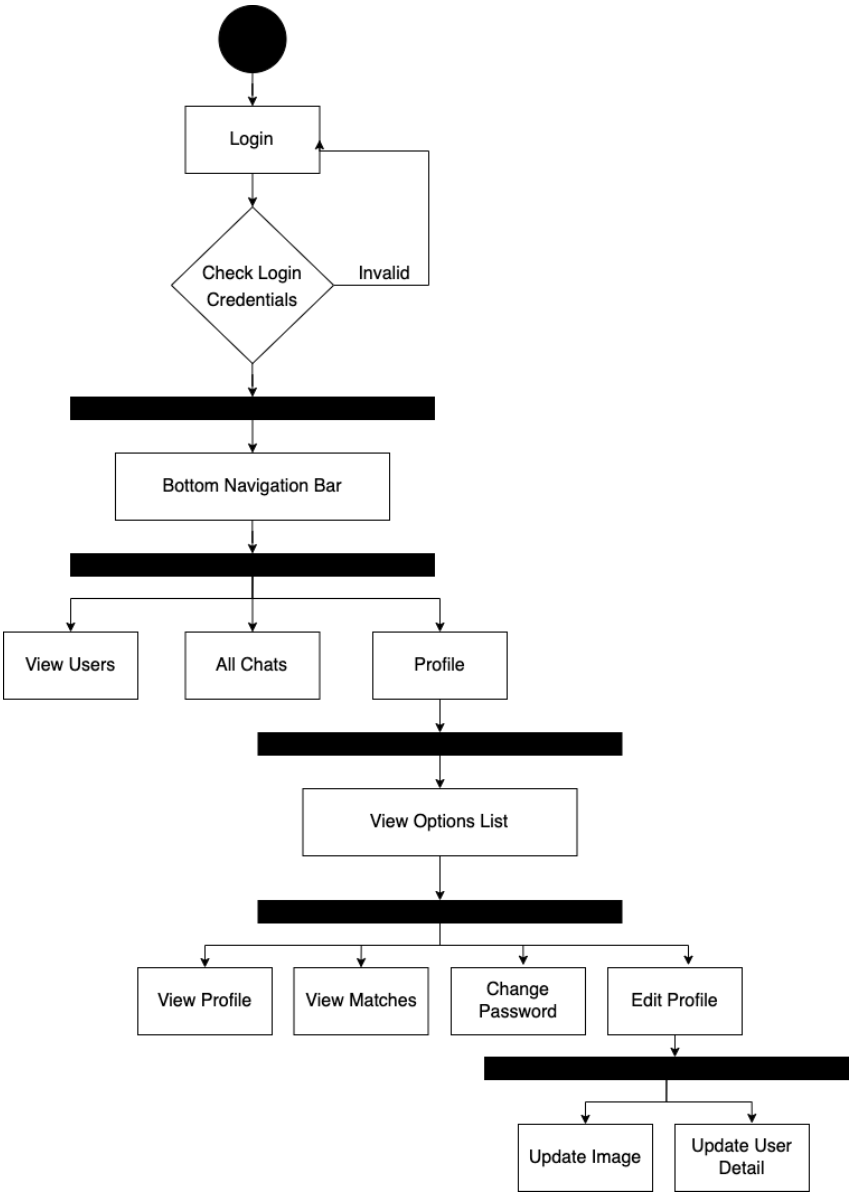

Figure 10 Activity Diagram

The above figure shows the series and action of flow one can perform in a system once the user logs in. It visualizes what process can be done in the system and also concurrent processes that can be done in the system, e.g., View Profile, View Matches, etc.

3.2 System Design

3.2.1 Component Diagram

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development. [7]

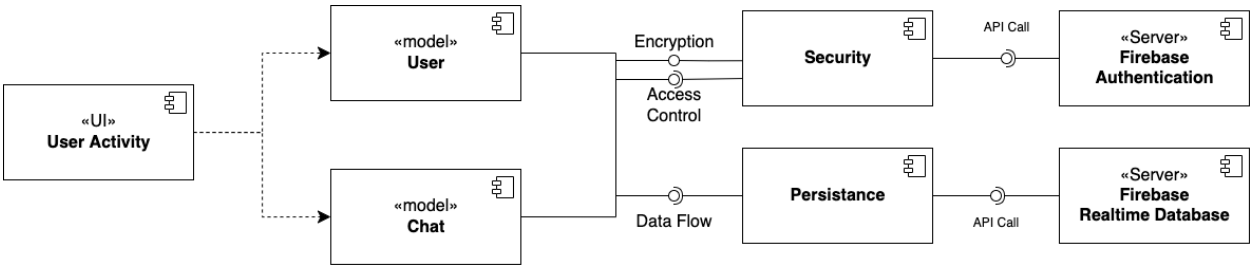

Figure 11 Component Diagram

The above figure shows what component are required for User Activity. It visualizes how those components are used to make the functionality and how components are helping to make the functionality of the system using the help of interfaces.

3.2.2 Deployment Diagram

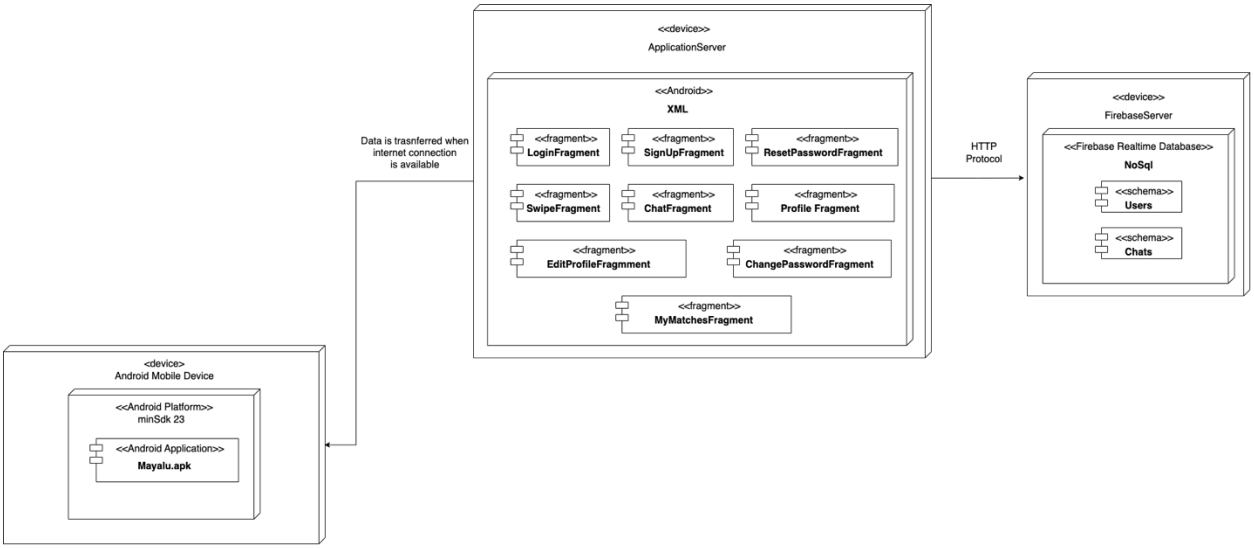

Figure 12 Deployment Diagram

The above figure assumes that the application is an android application, which is deployed in an environment using FirebaseServer. The application connects to the FirebaseServer using Application server using HTTP protocol.

Chapter 4: Implementation and Testing

4.1 Implementation

After designing the app, the only thing that needs to be done is implement it **s**o that we can release it as per the user satisfaction. Implementing the system requires a lot of resources and explanation which will not be completely explained in this report**;** however**,** some major aspects of the system are described below.

4.1.1 Tools Used

Different tools were used during the development of Mayalu. Below are the tools that were used for the development of the project.

I. **UI**: The project is done using Native Android Technique hence only XML is used to implement UI from the design made using Figma. XML is a markup language similar to HTML, but without predefined tags to use. In android, XML is used to implement UI-related data, and it's a lightweight markup language that doesn't make layout heavy,

II. **Design:** Before creating any system, its better approach to design the UI of the app. For Mayalu, a design tool called as AdobeXD was used. AdobeXD is a vector graphics editor and prototyping tool which is primarily web-based, with additional offline feature enabled by desktop application for macOS and Windows.

III. **IDE:** The IDE which was used during the development of the project is Android Studio. Android Studio is an IDE based on IntelliJ IDEA which provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV and Android Auto.

IV. **Programming Language:** Kotlin being recommended by Google itself to create Android application because of various features such as maintaining platform independence and cross-compilation support. So, I used Kotlin during the development of the application.

V. **Database:** For database, I used Firebase Realtime Database which is provided by Firebase. It is a cloud-hosted database in which data is stored as JSON and the data is synchronized in real-time to every connected client. Similarly for storing images Firebase Storage was used which is a service that developers can use to store and download files generated directly by client without the need of server-side code.

4.1.2 Implementation details of modules

I. **UI and UX of the app:** After the design that was made using Figma, it was time to start implementing the design into real application. The whole UI of the app is made using XML as the app is not made using any frameworks such as Flutter nor is based on Jetpack Compose. The whole application only contains three activities; SplashActivity::class, HelperActivity::class and UserActivity::class making the app lightweight. All other necessary components were made as a Fragment. For the UX part of the app, Navigation component was used. Navigation Component is a part of Android Component released along with Jetpack Compose but also supports Native Application. It is a new way for managing fragments without having to deal with fragment managers, transactions or back stack. It provides something known as NavGraph which was used to plot the whole UX of the app. Also, navigation component provides us with Nav Controller which is used to manage Fragments in the application.

II. **Email Verification:** The email verification provided by Firebase has many restrictions so for mail service used during the verification of the user when creating an account SendGrid API was used. SendGrid provided a cloud-based service that assists businesses with email delivery.

III. **Dependency Injection:** Hilt was used for dependency injection in the application which made the use it easier to inject frequently used components into fragments or activities easier. Hilt also made it easier to perform Instrumented Test and UI Test.

IV. **Swipe Cards:** One of the crucial parts of the application is the ability to allow user to like or dislike any user on swipe. For making it possible, a dependency used for this was “lorentzos swipecards”. After this dependency was added on to the projects the data required was set into the Swipe Card was set using a Custom Adapter made for it.

V. **Displaying Network Image:** Android doesn’t provide us any widgets to display network image into the app directly as it cannot be done in main thread. So, a third-party dependency called as Glide was used in the project.

4.2 Testing

After the completion of the implementation phase and even during the development phase, the app was tested. Some parts of the app followed TDD approach while many parts of the app were tested

afterwards. Many types of tests were carried out into the system. From UI testing to Instrumented test and also Local Unit Test, all these tests were performed to check the app's performance, functionality, time handling, user's action and much more.

4.2.1 UI Testing

UI Testing is to verify whether the results from the actions performed by user are as expected or not. From opening an activity to check whether a button is clickable or not, all these test fall under UI test. For UI Test in the application Espresso was used. Some of the UI test done in the app are displayed below.

Table 1 UI Test Case for Fragment and Activity Isolation

Test Case ID	Test Description	Test Functions	Expected Result	Actual Result	Remarks
TC_01	Check if splash activity launches	@Test fun test_isSplashActivityBeingLaunched() { ActivityScenario.launch(SplashActivity::class.java)	Splash Activity should launch	Splash Activity Launched	Pass

TC_02	Check if user activity launches	@Test
		fun test_isUserActivityBeingLaunched() {
		ActivityScenario.launch(UserActivity::class.java)
		}
			User Activity should launch	User Activity is launched	Pass

TC_03	Check if login fragment launches	@Test fun test_isLoginFragmentLaunched() { launchFragmentInHiltContainer<LoginFragment>(	Login Fragment should launch	Login Fragment launches	Pass

		factory = helperOnBoardingFactory) onView(withId(R.id.loginFragment)).check(matches(isDisplayed())) }			

TC_04	Check if signup fragment launches	@Test
		fun
		test_isSignUpFragmentLaunched()
		{
		launchFragmentInHiltContainer<SignUpFragment>(
		factory = helperOnBoardingFactory
		)
		onView(withId(R.id.signUpFragment)).check(matches(isDisplayed()))
		
		}
			SignUp Fragment should launch	SignUp Fragment launches	Pass

Table 2 UI Test Cases for Activity and Fragment Navigation

Test Case ID	Test Description	Test Functions	Expected Result	Actual Result	Remarks
TC_05	Check if SignUp button redirects from login fragment	@Test Fun test_ifIsRedirectedFromLoginToSignUpFragmentOnButtonClicked() { launchFragmentInHiltContainer<LoginFragment>(	Should be redirected to signup fragment	Is Redirected to signup fragment	Pass

	to signup fragment	factory = helperOnBoardingFragmentFactory) onView(withId(R.id.txtSignUpBtn)).perform(click()) onView(withId(R.id.fragmentSignUp)).check(matches(isDisplayed())) }			

TC_06	Check if back button navigates back to login fragment	@Test
		Fun test_ifIsRedirectedBackFromSignUpToLoginFragmentOnButtonClicked() {
		launchFragmentInHiltContainer<LoginFragment>(
		factory = helperOnBoardingFragmentFactory)
		onView(withId(R.id.txtSignUpBtn)).perform(click())
		onView(withId(R.id.fragmentSignUp)).perform(pressBack())
		
		onView(withId(R.id.fragmentLogin)).check(matches(isDisplayed()))
		}
			Should be redirected back to login fragment	Is redirected back to login fragment	Pass

TC_07	Check if pressing back	@Test	App Should Close	App Closes	Pass

	from an activity closes app	fun test_ifPressingBackFromActivityClosesApp() {
		ActivityScenario.launch(SplashActivity::class.java)
		ViewActions.pressBack()
		onView(withId(R.id.activitySplash)).noActivity()
		}			

4.1.2 Local Unit Test

A local test runs directly on your own workstation, rather than an Android device or emulator. As such, it uses your local Java Virtual Machine (JVM), rather than an Android device to run tests. Local tests enable you to evaluate your app's logic more quickly. However, not being able to interact with the Android framework creates a limitation in the types of tests you can run. [8]

Table 3 Test cases for ChangePasswordFragment

Test Case ID	Test Description	Test Functions	Expected Result	Actual Result	Remarks

TC_08	Check the system behavior when new password and retype password are different	@Test
		fun `different new password and retype password returns false`() {
		val currentPassword = “111111”
		val newPassword = “123456”
		val retypePassword = “12345s”
		assertThat(changePasswordFragment.isValudInput(currentPassword, newPassword, retypePassword)).isFalse()
		}
			Should expect a false boolean	Is returned a false boolean	Pass

TC_09	Check the system behavior when new password and retype password are different	@Test
		fun `different new password and retype password returns false`() {
		val currentPassword = "111111"
		val newPassword = "123456"
		val retypePassword = "123456"
		assertThat(changePasswordFragment.isValudInput(currentPassword, newPassword, retypePassword)).isFalse()
		}
			Should expect a true boolean	Is returned a true boolean	Pass

TC_10	Check the system behavior when empty password field is passed to ChangePasswordFragment	@Test
		fun `any empty password returns false` () {
		val currentPassword = ""
		val newPassword = "123456"
		val retypePassword = "123456"
		assertThat(changePasswordFragment.isValudInput(currentPassword, newPassword, retypePassword)).isFalse()
		
		}
			Should expect a false boolean	Is returned a false boolean	Pass

Table 4 Local Unit test for Functions

Test Case ID	Test Description	Test Functions	Expected Result	Actual Result	Remarks
TC_11	Check the system behavior	@Test fun `invalid email returns false`(){ val email = "abc.com"	Should expect a	Is returned a false Boolean	Pass

	when provided with invalid email	val checker = PatternsCompat.EMAIL_ADDRESS.matcher(email).matches() }	false Boolean		

TC_12	Check the system behavior when provided with valid email	@Test
		fun `valid email returns true`(){
		val email = abc@abc.com
		val checker = PatternsCompat.EMAIL_ADDRESS.matcher(email).matches()
		assertThat(checker).isTrue()
		}
			Should expect a true Boolean	Is returned a true Boolean	Pass

TC_13	Check system behavior when age provided is less than 18	@Test
		fun `less than 18 years old returns false`(){
		val age = SignUpPersonalDetailFragment().getAge(2022, 1, 1)
		assertThat(age).isLessThan(18)
		}
			Should expect a false boolean	Is returned a false boolean	Pass

TC_014	Check system behavior when age provided is more than 18	@Test
		fun `more than 18 years old returns true`(){
		val age = SignUpPersonalDetailFragment().getAge(2000, 1, 1)
		assertThat(age).isGreaterThan(18)
		}
			Should expect a true boolean	Is returned a true boolean	Pass

Chapter 5: Conclusion and Future Recommendations

5.1 Conclusion

My goal was to create an application where people can meet and connect each other without having the need to know them. The current application has fulfilled these goals. I followed the specifications strictly but enhanced some of the features when there was need for it to be done. With the goals achieved the basis of the application and this project has been achieved. Building this project has been both fun and challenging for me since I got to learn many new things along the way while I enjoyed building it.

As I came to the end of the project, I realized that there are many enhancements that can be made on the application. Many suggestions and new ideas came down the way when I gave my friends to test the application. But for now, I decided to follow the specification because there were realistic to achieve in this given amount of time. Any other enhancements to the application can be done in future development of the application.

5.2 Lesson Learnt/ Outcome

After the completion of this project, a person can connect and chat with many other users over the world. It will also allow similar interest people to find and connect with each other much faster.

5.3 Future Recommendations

Some of the things that can be added into this project are:

I. Only being available for Android devices right now, I believe that this system can be made for iOS and even web platform.

II. Firebase doesn't allow or provides many crucial features; hence the app's own API can be made to tackle this problem.

III. Notification service for particular device when receiving is not yet available in this app which can be further included in the app.

IV. Currently, the project is made using MVC pattern which is outdated and not recommended. The app can be converted to current trending architecture components such as MVP or MVVM.

V. Almost all operations on this app runs on Main thread which can be improvised by making intensive tasks in the app run on background thread by applying the concepts of multithreading using RXJava and Coroutines.

References

[1] "What are Software Development Methodologies? | Alliance Software," Allience Software, [Online]. Available: https://www.alliancesoftware.com.au/introduction-software-development-methodologies/#:~:text=Software%20development%20methodology%20is%20a,the%20form%20of%20defined%20phases..

[2] P. Smith, Artist, *Waterfall Model.* [Art]. Wikipedia, 2009.

[3] R. Lyons, "What Is Tinder? What You Should Know About the Dating App," [Online]. Available: https://www.businessinsider.com/what-is-tinder. [Accessed 27 March 2022].

[4] "Hinge (app)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Hinge_(app).

[5] "Dynamic modelling in object oriented analysis and design," Geeks For Geeks, 22 April 2020. [Online]. Available: https://www.geeksforgeeks.org/dynamic-modelling-in-object-oriented-analysis-and-design/. [Accessed 28 March 2022].

[6] C. Vanner, "bizagi," [Online]. Available: https://www.bizagi.com/en/blog/process-modeling-and-mapping/what-is-process-modeling-6-essential-questions-answered#:~:text=Process%20modeling%20is%20the%20graphical,context%20of%20the%20business%20environment..

[7] "Component Diagrams - See Examples, Learn What They Are," smartdraw, [Online]. Available: https://www.smartdraw.com/component-diagram/#:~:text=A%20component%20diagram%2C%20also%20known,is%20covered%20by%20planned%20development..

[8] "Build local unit tests," Google, [Online]. Available: https://developer.android.com/training/testing/local-tests.

[9] A. Wong, "The Best Hinge Conversation Starters (For Guys and Girls)," [Online]. Available: https://alexwongcopywriting.com/best-hinge-conversation-starters/.

Appendices

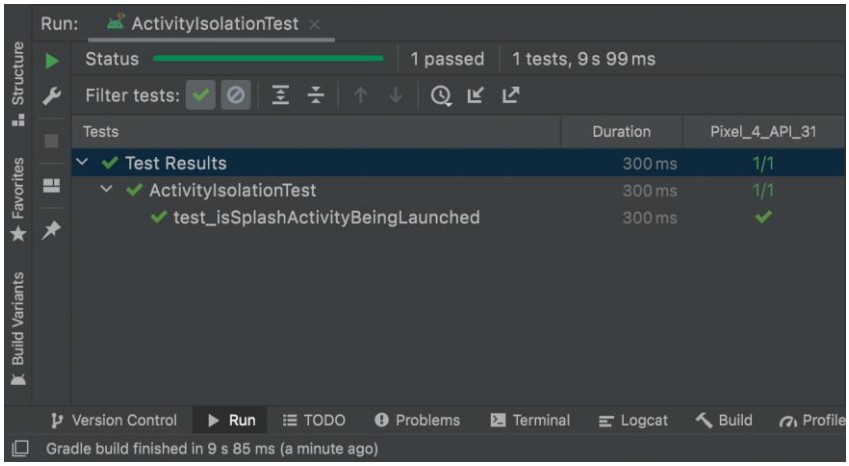

Figure TC_01

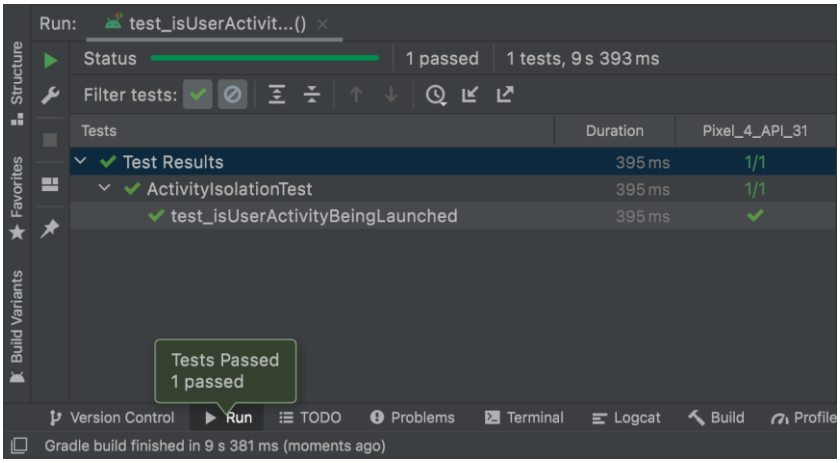

Figure TC_02

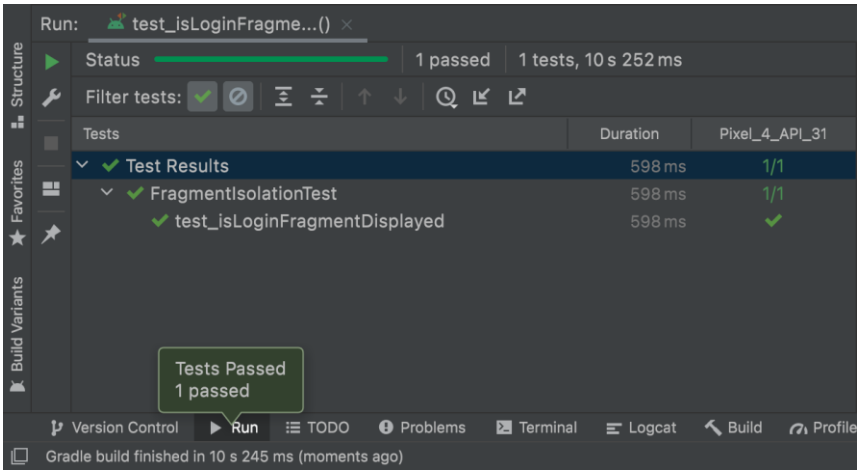

Figure TC_03

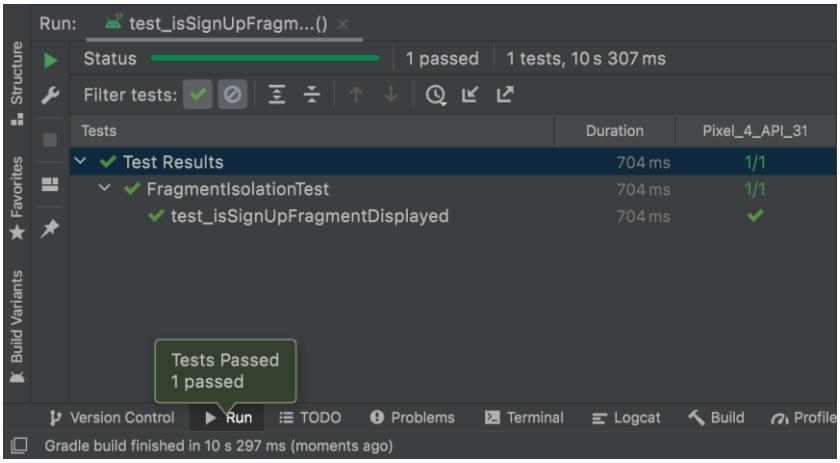

Figure TC_04

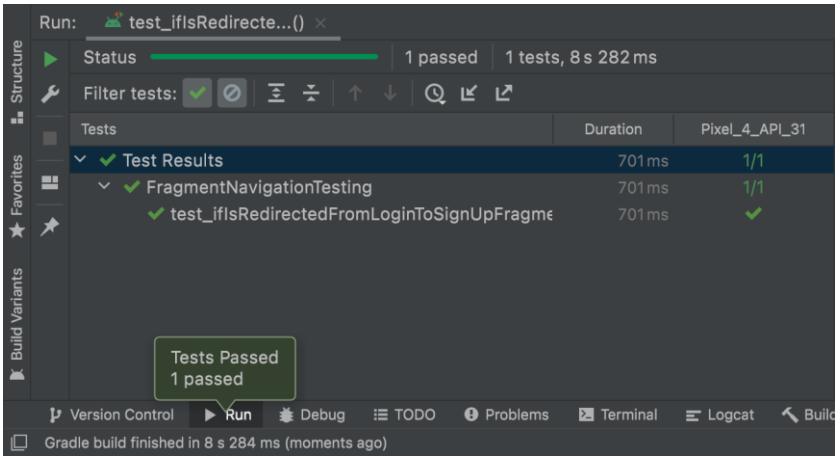

Figure TC_05

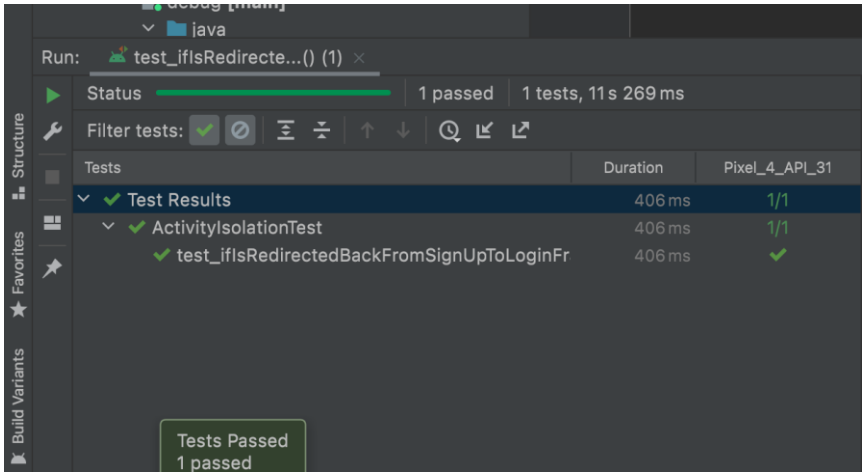

Figure TC_06

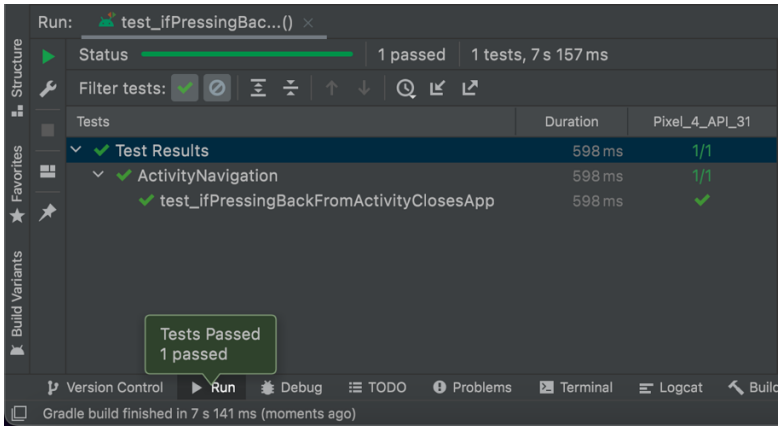

Figure TC_07

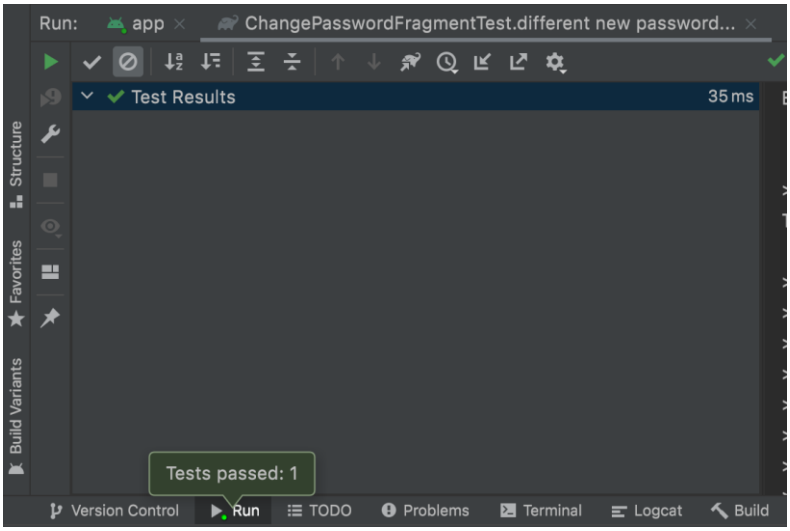

Figure TC_08

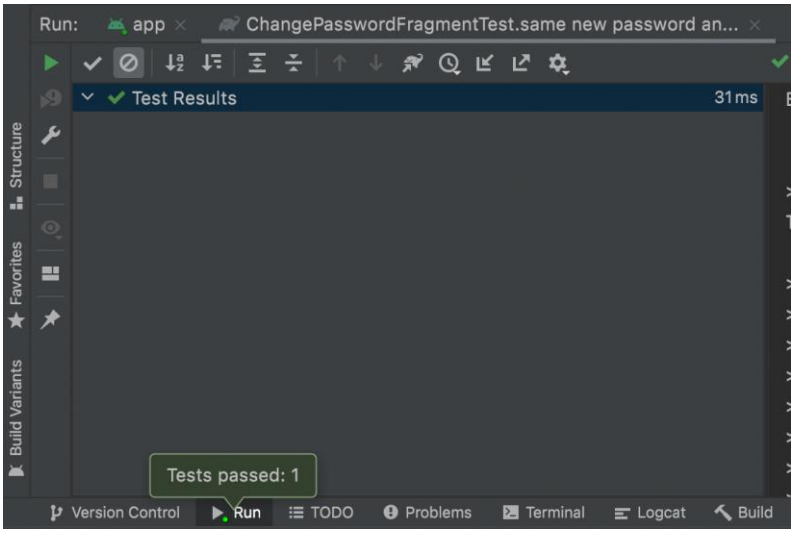

Figure TC_09

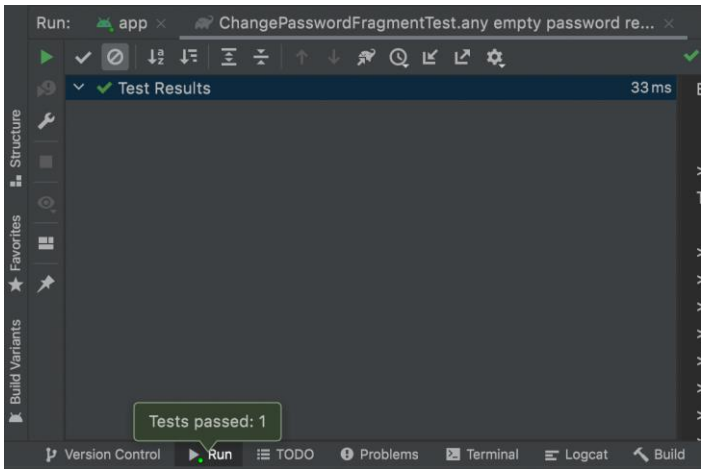

Figure TC_10

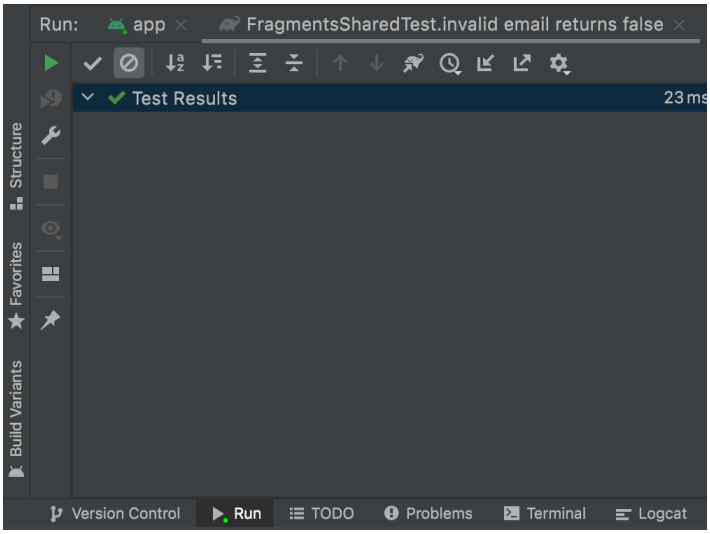

Figure TC_11

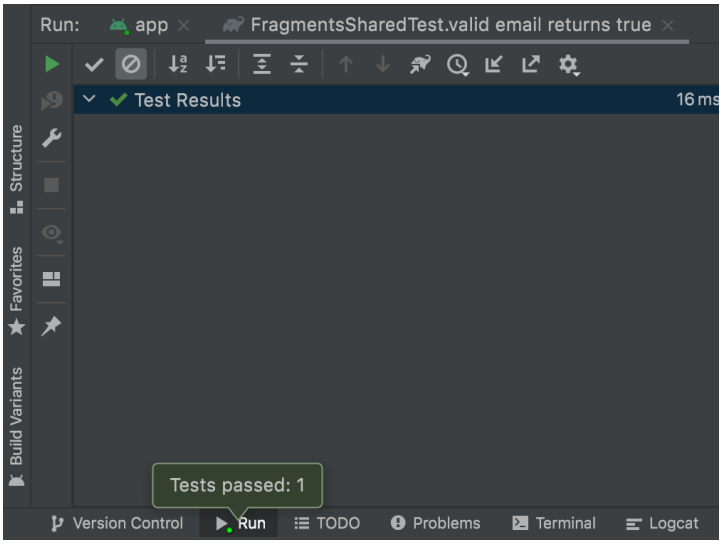

Figure TC_12

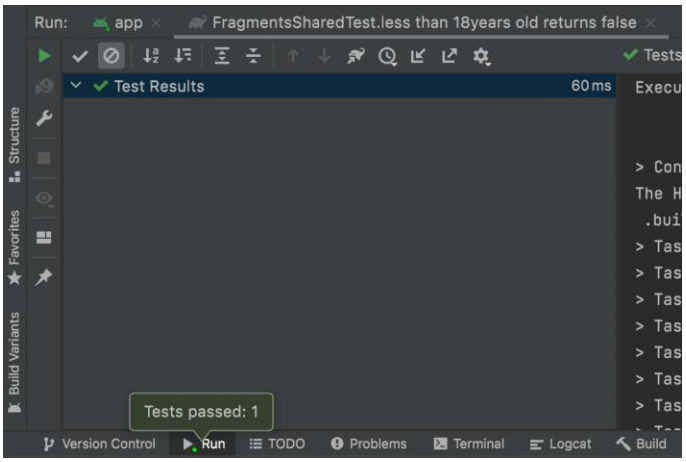

Figure TC_13

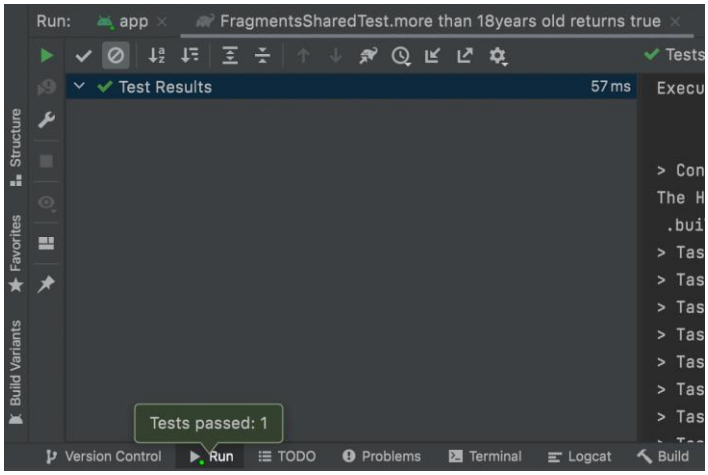

Figure TC_14