ICT2105 Mobile Application Development

Project Design Report

AY 2021/2022, Trimester 2

HR Android Application

Lab Group 18

Member: Mohamed Asif S/O K A Amanullah (2000582)

Member: Danial Ashidiq (2000867)

Member: Muhammad Muhaimin (2002950)

Member: Hun Yew Kwang Brian (2002255)

1. Introduction

During these times of pandemic, employees are required by law to work from home, and possibly in the future, working from home might be the norm in the industry. Everyone's livelihood has changed dramatically as a result of this severe upheaval. A survey conducted by National University Health System's (NUHS) Mind Science Center, found that 61 percent of employees working from home reported feeling stressed, compared to front liners. (NUHS)

A good Human Resource Management (HRM) is the coordination of an organization's people to achieve specific business objectives, fulfilling staff needs, and maintaining employee satisfaction and wellbeing. Human Resource departments around the country are grappling with a new problem as a result of the increased stress experienced by workers who work from home.

Thus, developing a robust human resource (HR) application would be essential to streamline HR business processes while taking care of the well-being of their employees. Through an Open-Ended Survey, Interview, and Competitive Product Review, the team would get a better understanding of the various HR business processes. The team has chosen to target Attendance Taking, Claims Management, and Machine Learning-enabled Chatbot.

The team targeted these three avenues as per the client's need to have an attendance-taking module to ensure the employee is within his required premises when starting to work and each employee's working hours tracked. A Machine Learning-enabled Chatbot to help care for the mental wellbeing of their employees. And lastly, submitting Claims was selected due to the survey results where the team sent the survey to multiple working adults and gathered that submitting claims was the process where that took the longest to resolve. The team intends to meet these criterias with the aid of the following mobile features listed: Firebase, GPS, Biometric (Fingerprint), Camera, Chatbot, Network Service, Notification.

2. Requirements & User Stories

2.1.1 Survey

The team conducted surveys over a period of a week and received a total of 17 responses. Through surveying, it enables the team to outreach a wide audience,gauging their satisfaction

level with the current processes, and have a better understanding on how the current HR processes are being carried out.

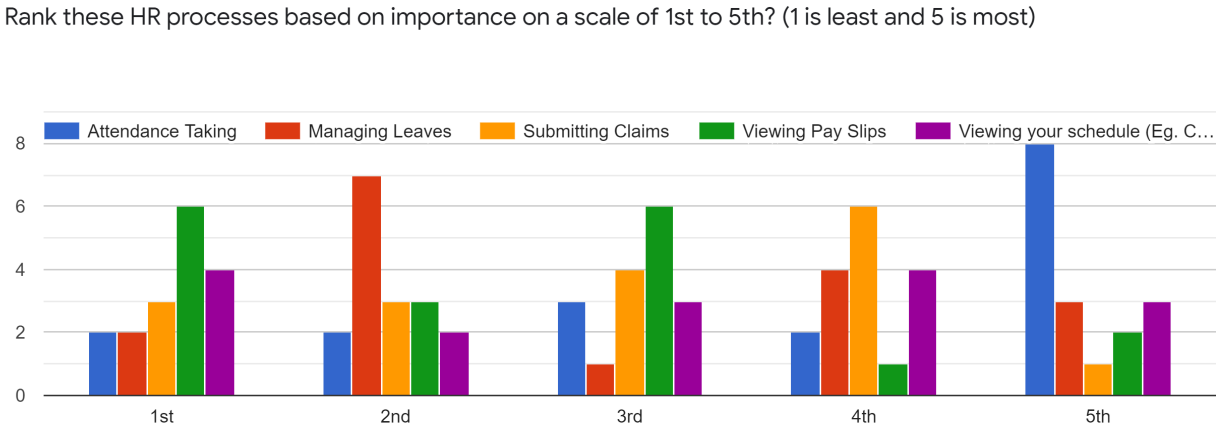


Figure 2.1.1 Gathered data on the ranking of HR processes based on their importance.

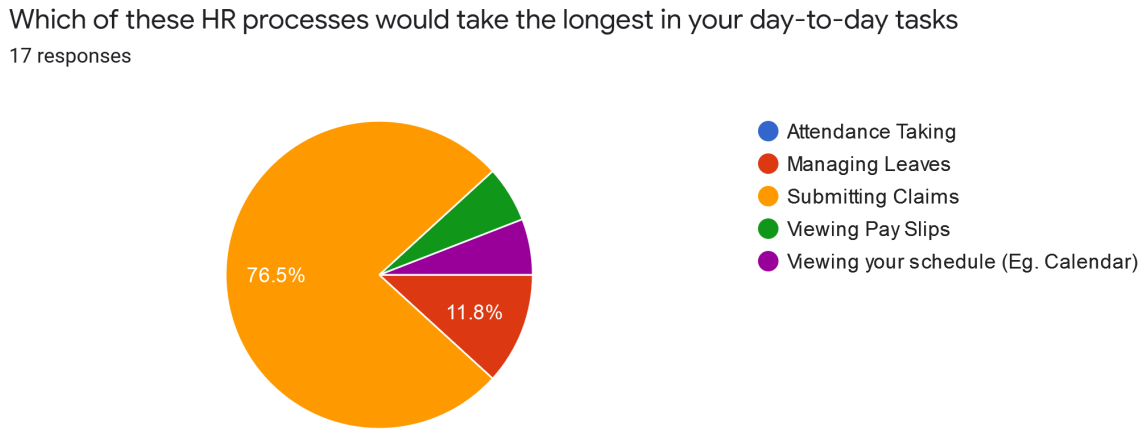


Figure 2.1.2 Gathered data which of the following HR processes would take the longest to complete.

Through these data collected, the team was able to better understand the user's needs and requirements on a larger scale with the inputs of many people.

2.1.2 Interview

The team has chosen to use an interview and decided to conduct an interview with the client in order for the team to gather accurate data. We had 2 scheduled sessions of interviews with the client ,Mr Madhu (from Precursor), through a zoom meeting.This allows the team to gather data on requirements/needs in greater depth and details that the survey could not cover. The findings of the interview are on the table below.

Module	Finding
Overall	To be a mobile application which compliments their existing Web Application.
Attendance Taking	Check Location when “Checked-In” & to store hours worked onto DB for their web portal to retrieve the information.
Machine Learning-enabled Chatbot	Wishes for an Artificial Intelligence Chatbot using deep learning and which the Chatbot would be trained using a model periodically over time.

Figure 2.2.3 Interview Findings Table

2.1.3 Competitive Product Review

The team has decided to review a competitor called Woebot Health to evaluate and gain insight on the industry standard of delivering mental health care. The team has chosen Woebot as products to review as they are the current leader in delivering mental care through various technologies.

	Things Observed that could be used within our Chatbot
1	Allow simple conversations such as “Hi” and etc
2	Provide Therapeutic Exercising to help calm mood
3	Provide appropriate links for user depending on how the conversations go

Figure 2.3.1 Findings Table

2.1.4 Conclusion

The team has decided to move forward with Attendance-Taking, Claims Management, and finally a Machine Learning-enabled Chatbot integrated into the Human Resource Application for Mr Madhu. Other features such as Viewing Payslip, Leaves management, etc would be up for future implementation due to the lack of available personnel in the team.

2.2 User Stories

The user stories are not organized in order as this section highlights the key user stories. (Refer to Appendix for more information)

User Story 2

Title: Login using Fingerprint Sensor	Priority
User Story: As an employee of the company, I want to log in to the company's HR application using my fingerprint to access the services in it.	HIGH
Acceptance Criteria: 1. The user must be shown the splash screen for 2 seconds once he/she launches the application from the phone's home screen. 2. Once the splash screen is displayed, the user must be directed to the login screen. 3. Once the user clicks on the fingerprint icon on the login screen, the user will be prompted to scan his fingerprint. 4. If the user's fingerprint matches with the fingerprint registered on his/her phone, the user will be directed to the home screen of the application. 5. If the user's fingerprint does not match with the fingerprint registered on his/her phone, the user will remain on the login screen and an error message will be displayed.	

Table 3.1.2 User Story 2 Login using Fingerprint Sensor

User Story 3

Title: Check In from Home Screen	Priority
User Story: As an employee of the company, I want to check in on the application's home screen so that I can mark my attendance for the day.	HIGH
Acceptance Criteria: 1. The user presses the "Check In" button that is displayed on the home screen of the application. 2. The user will be displayed a "Check In Success" message if the user's location is within the determined range. 3. Else, the user will be displayed a "Check In Unsuccessful" message if the user's location is not within the determined range.	

Table 4.3 User Story 3 Check In from Home Screen

User Story 6

Title: Submit Claim Expenses from Home Screen	Priority
User Story: As an employee of the company, I want to claim my expenses made during my trip out with the client.	HIGH
Acceptance Criteria: 1. The User's camera function will be opened to allow him/her to take a picture of the claim receipt. 2. Once the user takes the picture, the user will be directed to the new claims screen to enter other required details. 3. When the User presses the submit button on the new claims screen, a confirmation popup will be shown that claims have been applied for. 4. When the User clicks "Ok" on the confirmation popup, he/she will be directed to the claims screen.	

Table 1.6 User Story 6 Submit Claim Expenses

User Story 8

Title: Converse with Wickie	Priority
User Story: As an employee of the company, I want to seek assistance and guidance for any issues that I am facing so that I can improve in my mental well-being as a worker of the company	HIGH
Acceptance Criteria: 1. The User will click on Wickie and it will redirect the user to the Wickie Screen 2. The User will send text messages with Wickie and Wickie will appropriate responses depending on the text message	

Table 1.8 User Story 8 Converse with Wickie

3. Features & App Design

3.1.1 User Feature Description

User Features	Description
Claim Management	Claims management is a necessary function as it

	supports the employees' capability to apply for claims. Companies generally entitle their employees with a limited amount of claims that can be redeemed. This feature eases the employee's to monitor their eligibility to apply for claims by displaying the relevant data like the total claims eligible, claims taken, as well as available claims. It also shows a history of applied claims along with their details and status. Employees can also apply for claims and update the details if they are still pending.
Attendance Management	Attendance Management allows companies to keep track of employee's presence in the office. Most importantly, companies can check if any employee requires overtime wages.
Chatbot	The chatbot feature allows employees to have a conversation during work as a means to receive therapy and support from the company and also a way to manage productivity, and their mental health. They can seek advice and the chatbot will provide links specific to their problems. The chatbot uses the idea of Deep Learning with Neural Networks and Machine Learning to achieve an AI-based chatbot that is aware of the employee's condition.

3.1.2 Mobile Feature Description

Mobile Features	**Description**
Camera Function (Claim Management)	This camera feature is used in allowing users to submit evidence of their transactions. Users are provided the option to submit the image through the device's camera or gallery.
Fingerprint (Login)	If enabled, allows employees to login by scanning their fingerprint.
Real-Time Firebase (Database of the Application)	Firebase is used as the database of the application. It helps to facilitate Create, Read,Update and Delete (CRUD) functionalities.
Firebase Storage (Storing of Images)	Firebase Storage is being used to store images that are uploaded by the employees

	while applying for claims.
Machine Learning Chatbot	This Chatbot is developed with the use of python deep learning libraries, tensorflow, numpy, nltk and keras. The model is trained with a bag of words from a json file that contains any type of reply from a user. This is where the chatbot learns how to reply when reading a text from the user.
API Service call to chatbot	The chatbot model is deployed in the Heroku web server using python flask so the mobile application can call the web service using an API call to collect the response of the chatbot and receive the replies from the user.
Network (Service)	The network service is constantly running behind the application to check for network connectivity. If the user is disconnected, the user will be prompted to connect to a network.
Notification	This Notification feature is used to notify the users of the mood they selected. Users will then be redirected to the Wickie chatbot based on the notification.
GPS	Current Location of the Employee is obtained using the GPS functionality on the phone. The latitude and longitude is used to cross check if the employee is within the required location when “Checking-In” their attendance.

3.2 Storyboards

Fig 1.1 Storyboard for Claim Management

In Figure 1.1, Jake has to work overtime (OT) and thinks to himself he should go get his dinner now while doing his work. Jake goes to make an order for his dinner and pays for the food to get a receipt from the cashier. Now that he has the receipt, he can make a claim for the dinner due to working overtime. Jake has a lot of free time waiting to collect his dinner so he decides to make the claim for his dinner to HR with the HR app on his mobile phone.

Fig 1.2 Storyboard for Attendance Management

In Figure 1.2, Jake arrives to work at 8:30am and checks in with the HR app to verify his attendance. Unfortunately, his manager informs him that he needs to do OT today. Jake finishes his work at 9:30pm and checks out with the HR app so that he can get his OT pay without needing to notify anyone else.

Fig 1.3 Storyboard for Chatbot

In Figure 1.3, Jake feels tired and sad during work. He decides to chat with Wickie chatbot to lift up his mood. Wickie chatbot talks to him and gives him advice on how to feel better. Jake feels much better talking to Wickie chatbot and follows its advice while continuing his work.

3.3 Final UI App Design

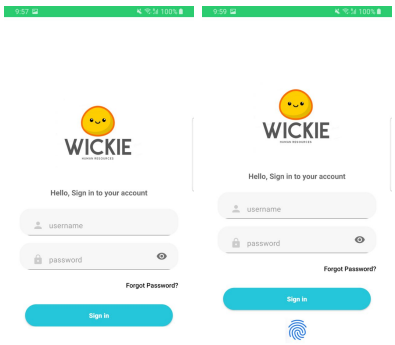


Fig 2.1 Login UI Screen with Fingerprint

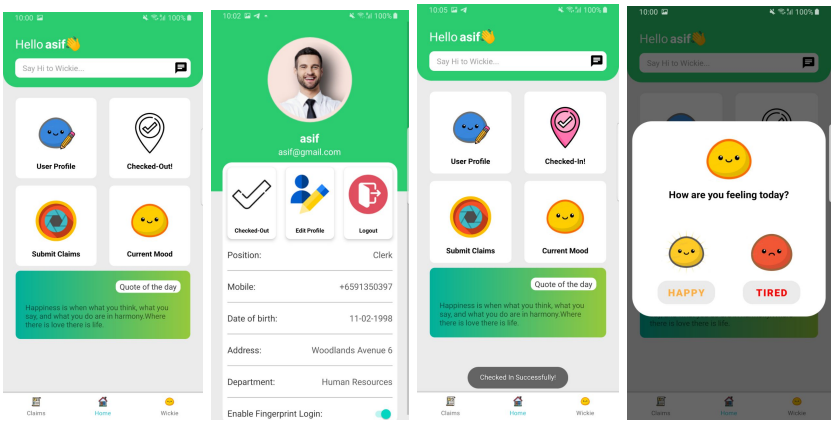


Fig 2.2 Home UI Screen

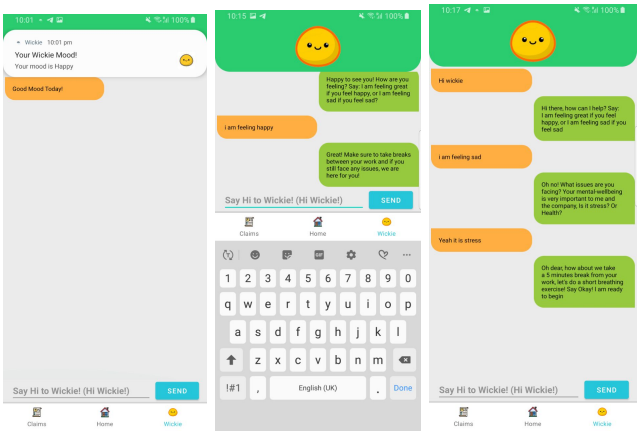
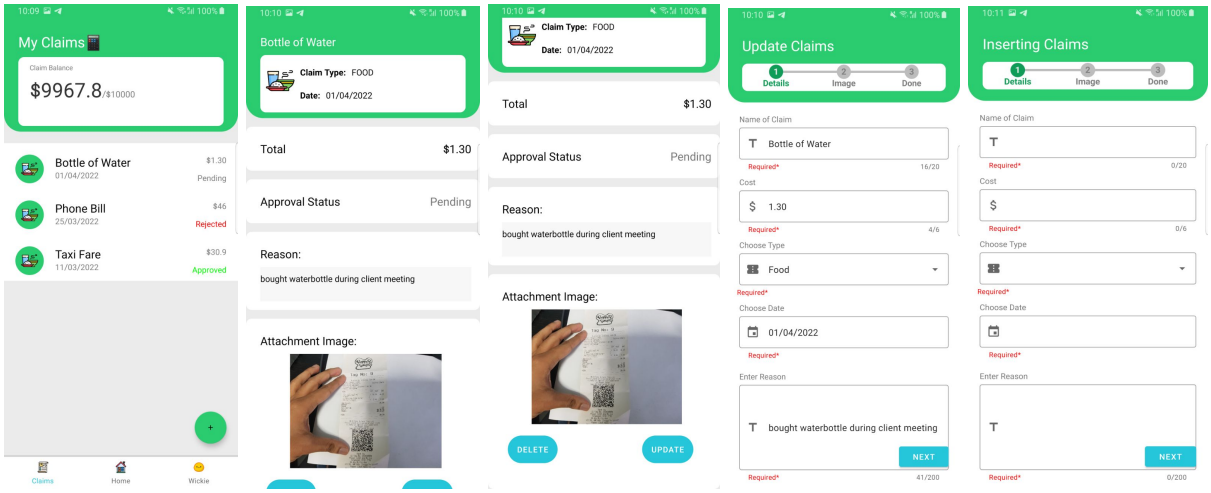


Fig 2.3 Chatbot UI messages



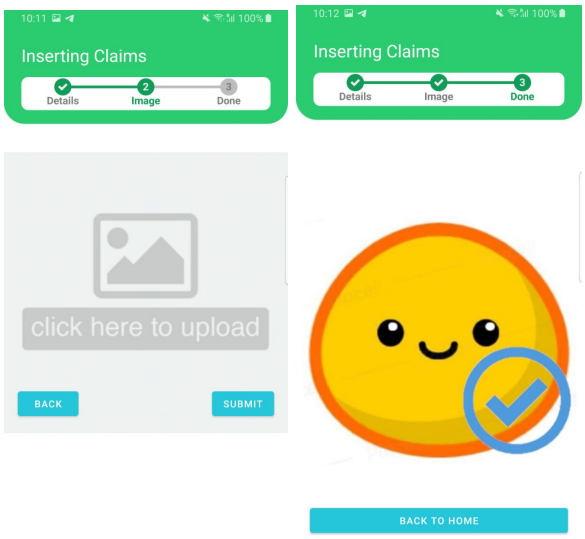


Fig 2.4 Claims and Claims Form UI

4. System Architecture & Implementation

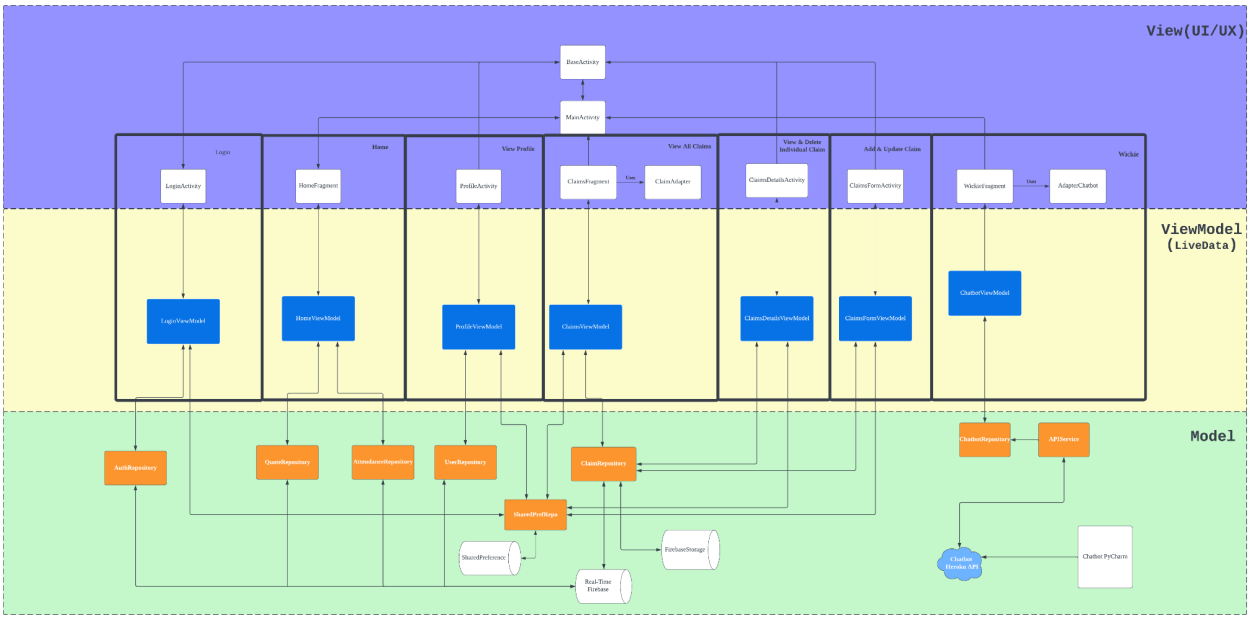


Fig 4.1 Final Software Architecture

4.1 MVVM Architecture

The team has implemented the application using the Model-View-ViewModel (MVVM) architecture. The following describes the layers in further detail:

- **4.1.1 Model**

  The app's data and business logic are represented by the model. One of the layer's recommended implementation options is to expose its data via observables, which allows it to be totally detached from ViewModel or any other observer/consumer.

- **4.1.2 View**

  The view's duty is to obtain data by observing (or subscribing to) a ViewModel observable and updating UI elements accordingly.

- **4.1.3 ViewModel**

  ViewModel interacts with the model and creates observable(s) that a View can see. The ViewModel can provide hooks for the view to pass events to the model if desired.
  One of the most significant implementation strategies for this layer is to decouple it from the View, which means that the ViewModel should be unaware of the view with which it is communicating.

The team chose MVVM architecture as it provides a structure, making code easier to navigate. The code is further decoupled and easier to test. MVVM also makes the code easier to maintain and add features.

4.2 ViewModelFactory (Dependency Injection)

Custom ViewModelFactory was implemented for every ViewModel that has dependencies on various services such as repositories.

```
class ProfileViewModelFactory(private val userRepository : UserRepository, private val prefRepo: SharedPrefRepo) : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(ProfileViewModel::class.java)) {
            @Suppress( ...names: "UNCHECKED_CAST")
            return ProfileViewModel(userRepository, prefRepo) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

4.3 Inheritance (Base Activity)

The application has implemented Inheritance where all the activities on the application inherits from BaseActivity. The BaseActivity consists of common attributes and functions that all activities require. The figure below shows an example of attributes and functions that can be found inside BaseActivity.

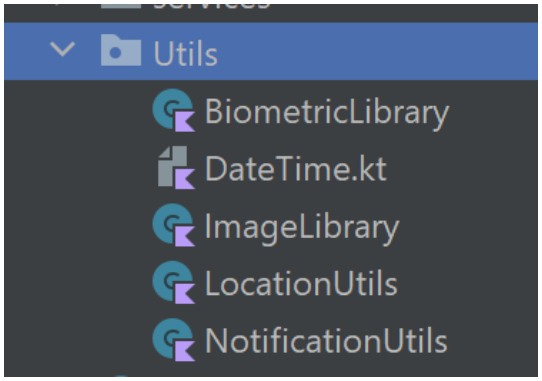
4.4 Singleton

The repositories are implemented as a singleton pattern to control object creation, limiting instances to only one. This is done to prevent multiple instances of the same repository object accessing the Firebase.

```
companion object {
    // The usual for debugging
    private val TAG: String = "ClaimRepository"

    // Boilerplate-y code for singleton: the private reference to this self
    @Volatile
    private var INSTANCE: ClaimRepository? = null

    fun getInstance(context: Context): ClaimRepository {
        return INSTANCE ?: synchronized( lock: this) {
            INSTANCE?.let {  it: ClaimRepository
                return it
            }
            val instance = ClaimRepository()
            INSTANCE = instance
            instance  ^synchronized
        }
    }
}
```

4.5 Utilities

The Utilities are modularised to enable reusability in other features. Each utility has its own class that will act as a library to be used in the relevant activity classes. This reduces the need to implement common code across activity classes that use these utilities. Activity classes just need to initialize an instance of the utility.



4.6 RequestCallObject

Each function inside the repository returns a RequestCallObject which is a MutableLiveData. Inside this Request Call Object returns the status, message and the respective data from each repository call.

The example below shows the RequestClaimCall. This object is used as return type for all Claim calls made inside ClaimRepository. This was developed to standardize return output for all Firebase requests made. Where the status field refers to the status of the request made and the message field returns comments such as “DATA FOUND” or “NO DATA FOUND”.

```
class RequestClaimCall {
    var status = 0
    var message : String = "No Message"
    var claimArray : ArrayList<Claim> = ArrayList<Claim>()
    var claimTotal : Double? = null
}
```

4.7 Chatbot API and Machine Learning

(Refer to Appendix for code and description)

The chatbot is trained using tensorflow by opening up a json file first and compiles the list of words a typical user would message in a chat. The words will then be labeled and tagged so that the chatbot can now understand patterns of the text and identify what the user means when typing. This is then conducted by training and generating a model. The chatbot model named chatbotmodel.h5 is generated and it will then be deployed in the heroku server.

The APIService.kt class in the android application will then call the heroku website hostname to exchange data. This is where the user is able to send messages through the APIService, then through the website. The website will then respond to the message and the APIService retrieves the responses to be displayed in the application.

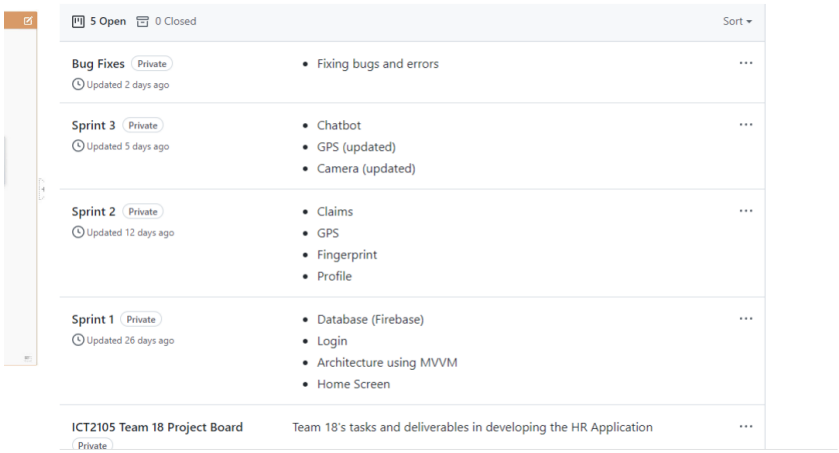
5. Project Management and Planning

During the first few sprints, we decided to use the excel sheets to keep track of our progress in the sprint and product backlog. However, after the first client meeting, we discovered some changes and decided to use the github project board to continue our progress. Hence, the excel sheets are not the latest updates. (Please refer to github project board team 18 for more details).

Product Backlog Highlights	Description
Sprint 1 (Completed)	By the end of the first sprint, the team is expected to complete the main base of the application and the skeleton codes that are necessary for HR application
Sprint 2 (Completed)	During the second sprint, the Fingerprint scanner will be implemented first followed by CRUD functions using Firebase as our data store. These two features must be completed by the end of sprint 2 and to have a finalized minimal working version of the application
Sprint 3 (Completed)	In sprint 3, the team is expected to deliver a fully working version of the application. The design of the camera, GPS and Chatbot will be completed by the end of the 3rd sprint and present to the client

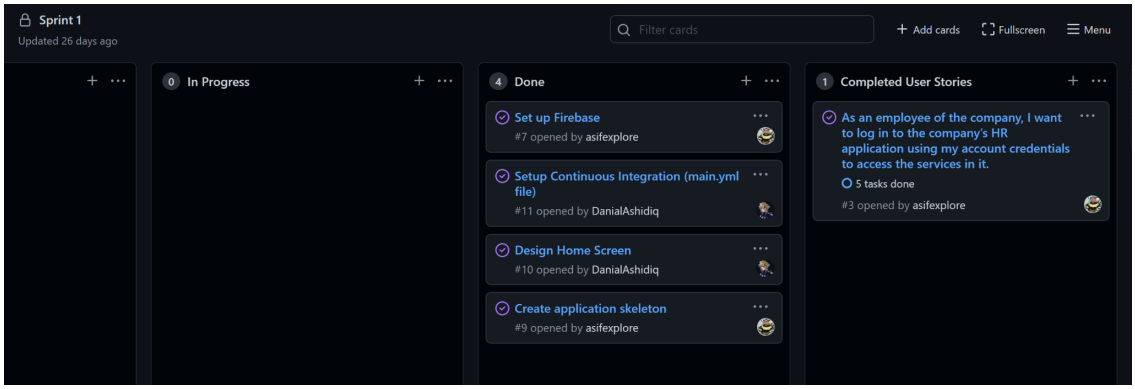
Sprint Backlog	Description
US001 - US0015	Can be seen in github for more updates, the appendix contains the non-updated as we proceed to use the project board from week 8.

5.1 Team Workflow

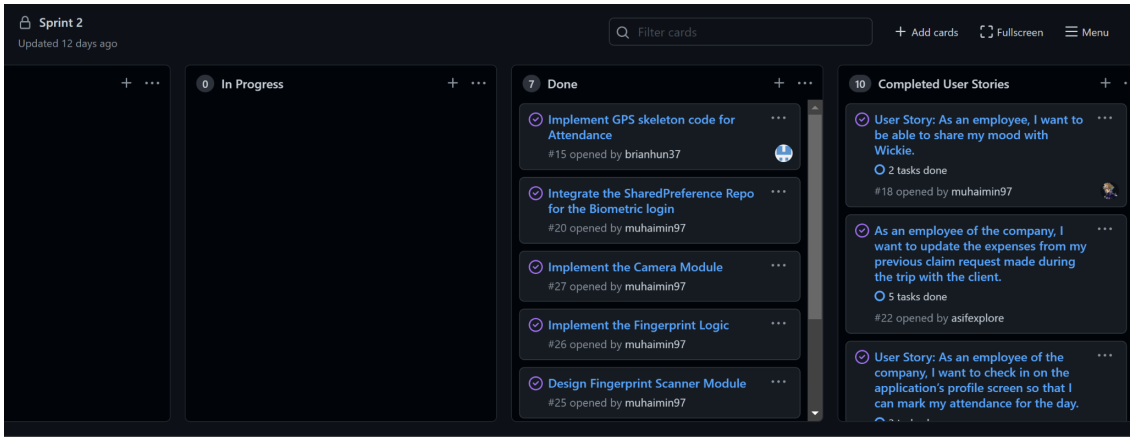


5.2 Sprint's Progress, Review & Retrospective

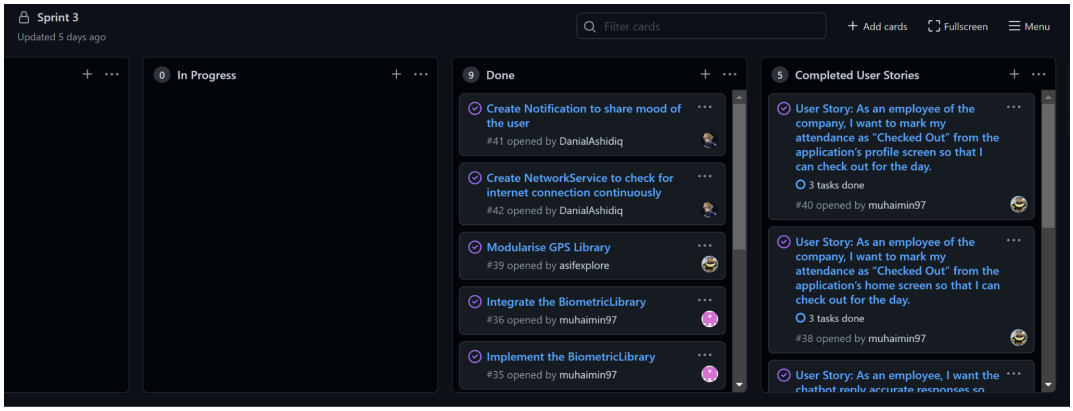
5.2.1 Sprint 1



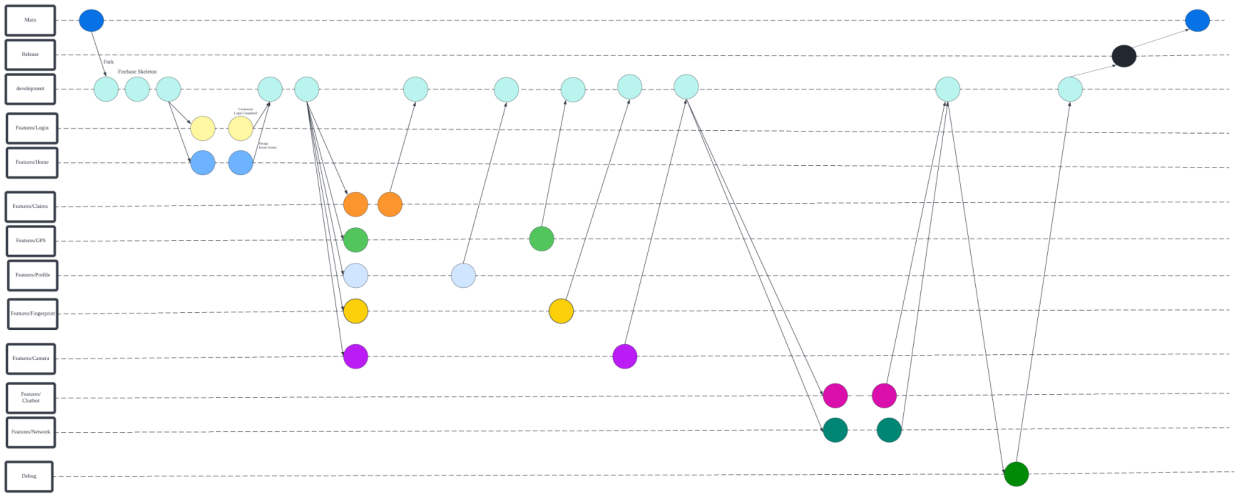
5.2.2 Sprint 2



5.2.3 Sprint 3



5.3 Description of the team's branching model and merging model

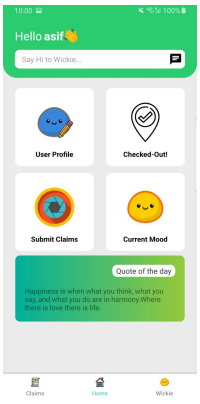


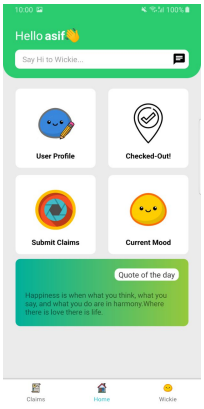
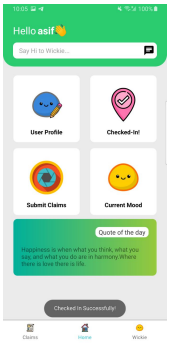
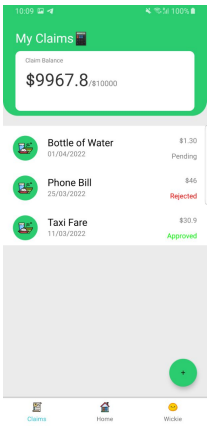
The team has created a main branch and a development branch. Initially, the team created branches based on the names of the team members to indicate roles of each team member. However, upon further feedback, the team decided to make branches based on the features instead. These feature branches originate from the development branch. The feature branches are where everyone would implement and test. After everyone tests that the code in the feature branch is working as expected, the feature branch will then be merged into the development branch. After all the feature branches are merged into the development, the team will conduct tests on the development branch to see if all the features are working. The team will then create debug branches based on the features to rectify bugs and make the necessary comments. For clearer understanding, please refer to the link in the references.

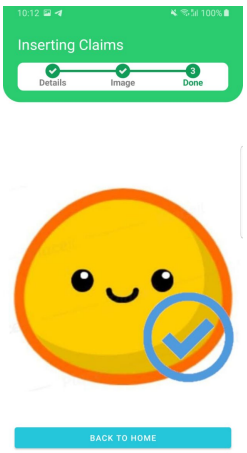
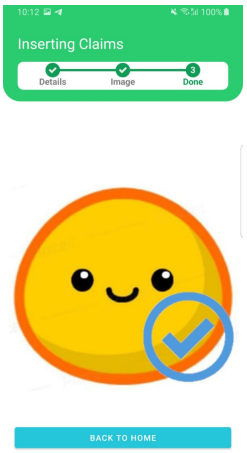
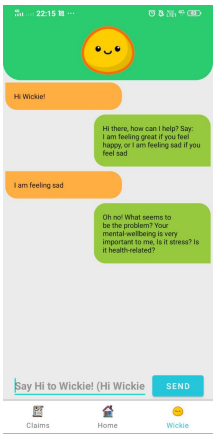
6. Application Testing

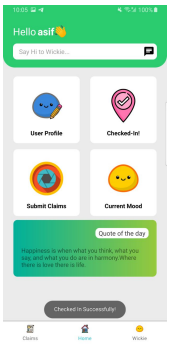
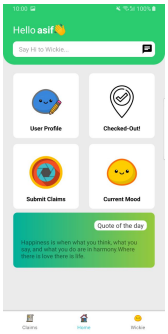
6.1 UAT Testing (Refer to Appendix for code and description)

The test cases in this section highlight the test cases for the key features.

Test ID	US ID	Case Description	Precondition	Steps	Screenshot	Status (Pass / Fail)
1	1	Login using Account Credentials	Login Page	1. Enter username: “asif” 2. Enter password: “123” 3. Click the Login Button 4. System validate credentials 5. Redirected to Home Page		Pass

2	2	Login using Fingerprint Sensor	Login Page Enable Fingerprint on Profile Screen	1. Click on the Fingerprint Image 2. Place finger below fingerprint scanner 3. Device validates fingerprint 4.		Pass
3	3	Check-in from Home Page	User checked-out Home Page	1. User clicks on “Checked-Out” button		Pass
4	5	View the claims	Logged in to Asif account	1.Click on Claims icon 2. User will be shown to his applied claims		Pass

5	6	Submit Claims from Home Page	Home Page	1.User clicks on Submit Claims button 2. System starts the camera 3. User takes a picture of the receipt 3. User accepts the image taken 4. User fills in credentials 5. User clicks “Next” 6. User checks the image 7. User clicks “Submit” 8. System notifies success		Pass
6	7	Update Claim Expenses from Update Claims Screen	Claims Page	1.User clicks on a “Bottle of Water” pending claim image 2. User redirected to its details page 3. User clicks “Update” 4. User changes the amount from 1.30 to 1.40 5. User clicks “Submit” 6. System notifies success		Pass
7	8	Converse with Wickie	Wickie Page	1. User clicks on the Wickie icon 2. User redirected to Wickie page 3. User types “Hello Wickie” 4. User clicks “send” button 5. Wickie responds accordingly 5. User types “I am feeling sad”		Pass

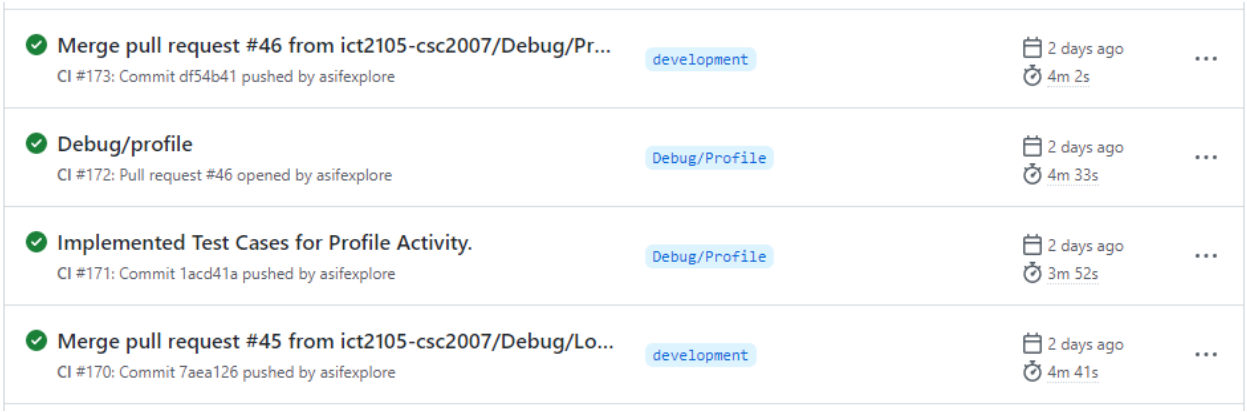
				6. Wickie responds accordingly		
8	3	Check-in from Home Page	User checked-out Home Page	1. User clicks on “Checked-Out” button		Pass
9	12	Check-out from Home Page	User checked-in	1. User clicks on the “Checked-In” button in the Home Page		Pass

In conclusion, the system passed the UAT for the key features as the system is able to receive the expected output for the features.

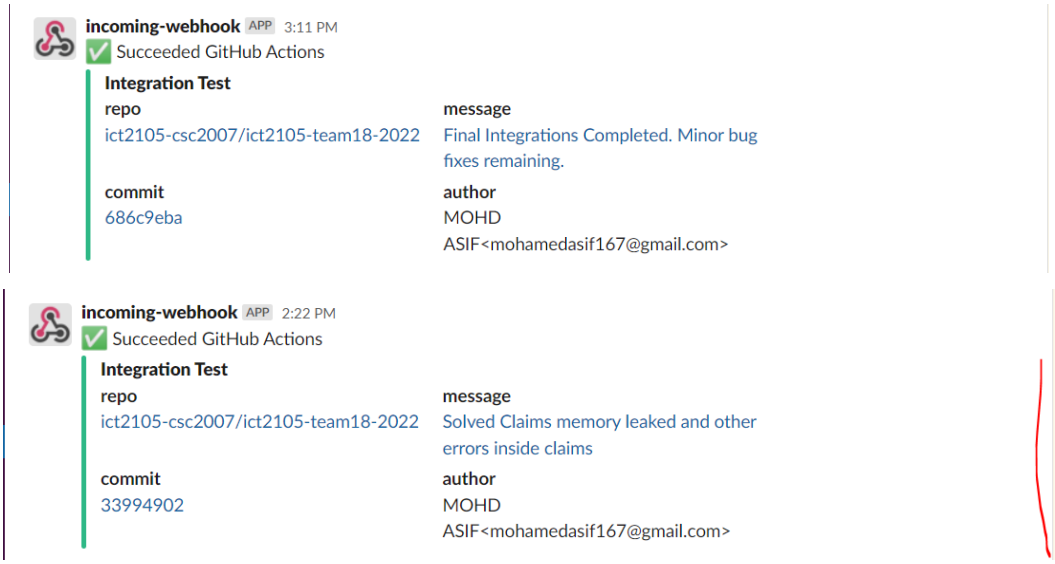
6.2 Integration Testing

Throughout the sprint phases, the team had continuous integration testing with reference to the Github Actions and incoming-webhook notifications in the Slack channel.

The team has made use of the Github Actions to determine if the corresponding commits or merge is successful. Based on the figure above, the older actions failed to be pushed into the development branch. However, after resolving the issues, the team has managed to successfully merge the development branch.



The figure above indicates the more recent actions made by the team, with the most recent on the list being a pull request. The team has successfully committed the following actions.



The team also made use of the slack notification channel to determine the outcome of the actions made by the team. Based on the figure above, the actions made had successfully integrated into the repository.

6.3 Unit Testing

The team implemented Unit Testing to test the Repository calls from ViewModels. As the application gets most of its data from Firebase Calls, the team prioritized developing Unit Testing them for them. With this, we were always able to check if data is being properly received from our Firebase Calls. The Unit Test Cases below are just some

examples of all our implementations. Go to AndroidTest inside the project structure to view the other test cases.

6.3.1 Login Testing

```
@Test
fun testSuccessLogin()
{
    var msg = ""
    val result = viewModel.login( username: "afiq",  password: "123").getOrAwaitValue().let {  it: RequestAuthCall
        msg = it.message
    }
    assertEquals( expected: "DATA FOUND", msg)
}

@Test
fun testFailLogin()
{
    var msg = ""
    val result = viewModel.login( username: "NoSuchUser",  password: "123").getOrAwaitValue().let {  it: RequestAuthCall
        msg = it.message
    }
    assertEquals( expected: "NO DATA FOUND", msg)
}
```

The functions above are located in the LoginActivityUnitTest, which are used to test the possible scenarios an employee might encounter during the Login process. The testSuccessLogin is used to test whether the system is able to receive data if the employee enters a valid username and password. The testFailLogin is used to test that the system is to indicate that it does not receive data if the employee input an invalid login credential.

Tests	Duration	vivo vivo 1902
✔ Test Results	787 ms	2/2
✔ LoginActivityUnitTest	787 ms	2/2
✔ testFailLogin	736 ms	✔
✔ testSuccessLogin	51 ms	✔

Upon running the tests, the system passed the unit test cases for the LoginActivityUnitTest as the functions receive the expected outputs.

6.3.2 Claims Testing

To test the functionality of the Claims feature, the team has implemented three test classes, the ClaimViewModelTest, the ClaimDetailsViewModelTest and the ClaimsFormActivityTest.

```
@Test
fun retrieveClaims()
{
    var msg = ""
    var claimTotal = 0.0
    val result = viewModel.retrieve( username: "asif").getOrAwaitValue().let {  it: RequestClaimCall
        System.out.println(it.claimTotal.toString())
        msg = it.message
        claimTotal = it.claimTotal!!
    }
    System.out.println(claimTotal.toString())
    System.out.println("Hello StackOverflow")
    assertEquals( expected: "DATA FOUND", msg)
}
```

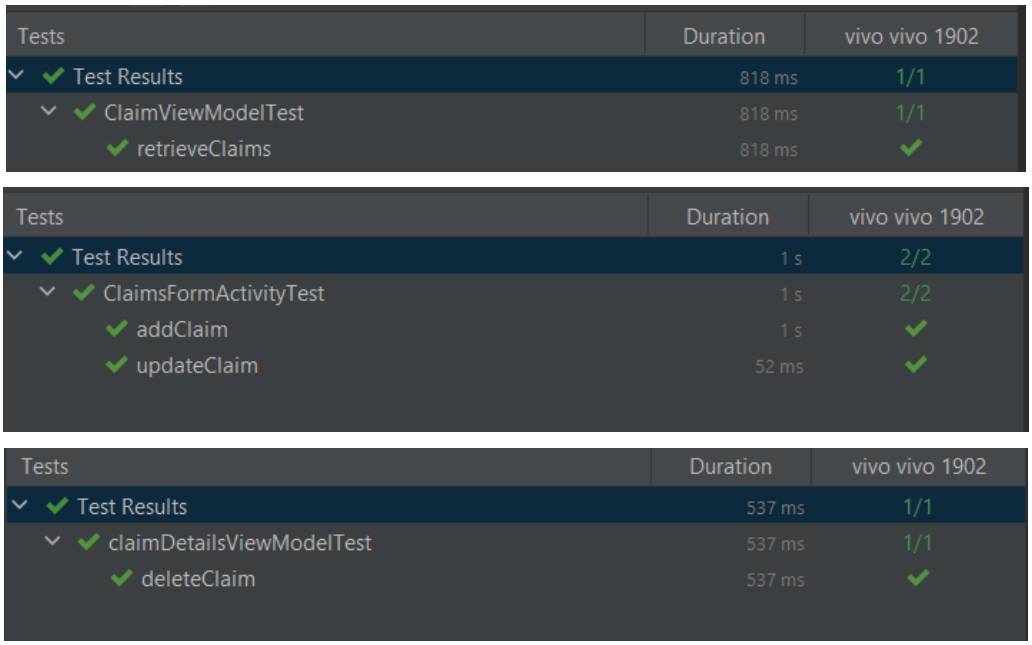
The ClaimsViewModelTest Class contains the test functions that are required for both the ClaimsFormActivity and ClaimsDetailViewModel. It contains the retrieveClaims test function to test that system receives data once if a valid username is passed in.

```
@Test
fun addClaim()
{
    var msg = ""
    val result = viewModel.create( username: "afiq", claimObj).getOrAwaitValue().let {  it: RequestClaimCall
        msg = it.message
    }
    assertEquals( expected: "Add Success", msg)
}
/*
Testing Update Function inside ClaimsFormViewModel
Run this function after running "addClaim". As of now data is added to test account called "Afiq"
*/
@Test
fun updateClaim()
{
    var msg = ""
    claimObj.title = "Updating Claim Title"
    val result = viewModel.update( username: "afiq",claimObj).getOrAwaitValue().let {  it: RequestClaimCall
        msg = it.message
    }
    assertEquals( expected: "Update Success", msg)
}
```

The ClaimsFormActivityTest tests the functions that are relevant only to the ClaimsFormActivity. It consists of the addClaim test function that will receive an “Add Success” message after taking in the username and the claim created by the employee. It also contains the updateClaim test function that will receive an “Update Success” message after the system updates the claim object tied to the employee.

```
@Test
fun deleteClaim()
{
    var msg = ""
    viewModel.deleteClaims(claimObj,  username: "afiq").getOrAwaitValue().let{  it: RequestClaimCall
        msg = it.message
    }
    assertEquals( expected: "Delete Success", msg)
}
```

The ClaimDetailsViewModelTest consists of functions that are only relevant to the ClaimsDetailViewModel. It consists of the deleteClaim test function that will receive a “Delete Success” after the system deletes a claim object tied to the employee.



Upon running the test functions for the Claims feature, the system was able to receive the expected outputs for all the tested functions. Hence, the team is able to conclude that the system passed the test cases covered in the Claims Unit Testing.

7. Conclusion

In conclusion, the team has achieved and met the following project objectives and managed to pull through together although severely short-handed. The team has understood MVVM very well and has applied in the project smoothly with enough commits. Furthermore, the application can be extended with the inclusion of an update profile page, biometric face ID using the logic from the fingerprint scanner, enhancing the model training of the chatbot so that it can better reply to any type of response by the user and extension of CRUD for leaves using the logic from claims.

8. Appendices (total 25 pages)

8.1 End of Sprint 1

8.1.1 Added User Stories:

US11 (Added for Sprint 2)	**As an employee I want the chatbot to be aware of my mental-well being. The chatbot is an AI that can identify issues of the employee, so that it can help me in my mental-well being**	- Research on Neural networks with chatbot implementation - Understand Deep Learning and Model Training for chatbot - Find a way to use database for model training - Find source codes with similar examples of chatbot - Automate the tests
US12 (Added for Sprint 2)	**As an employee, I want the chatbot reply accurate responses so that it can provide the appropriate assistance to manage my mental-well being**	- Implement chatbot logic with proper replies - Discover other data crawling techniques to increase model development - Design Wickie's Backend implementation - Design Wickie's Backend Responses -Automate the tests

8.1.2 Completed Task

UID	User Story	Task
US001	**As an employee of the company, I want to log in to the company's HR application using my account credentials to access the services in it.**	**- Design base app with splash screen UI** **- Design login screen UI** **- Design tabs for email and password** **- Design submit button** **- Implement Validation for login** **- Implement Error Messages for login** **- Automate the tests**
US002	**As an employee of the company, I want to log in to the company's HR application using my fingerprint to access the services in it.**	**- Design Fingerprint scanner library** **- Implement Fingerprint scanner logic** **- Implement validation for fingerprint** **- Implement error messages for fingerprint** **- Automate the tests**
US007	**As an employee of the company, I want to claim my expenses made during my trip out with the client.**	**- Design icon for the claim expenses** **- Implement Camera Library** **- Design new claim expenses screen** **- Store all entered data from employee to the database.** **- Automate the tests**
US008	**As an employee of the company, I want to update my claim expenses made during my trip out with the client.**	**- Design the icon for update claims** **- Design update claim screen** **- Implement update claim logic** **- Automate the tests** **- Meet Up Client for Sprint 1**

8.2 End of Sprint 2:

8.2.1 Added User Stories [Also added into the github project board]:

Note: Based on feedback for sprint 1 and 2, we have decided to use the sprint backlog from the excel sheet and transfer the items into the github project board. We discovered that it is much easier to keep track of our things for our project and also much easier for grading purposes.

User Story 1

Title: Login using Account Credentials	**Priority**
User Story: As an employee of the company, I want to log in to the company's HR application using my account credentials to access the services in it.	HIGH
Acceptance Criteria: 1. The user must be shown the splash screen for 2 seconds once he/she launches the application from the phone's home screen. 2. Once the splash screen is displayed, the user must be directed to the login screen. 3. The user enters his/her email and password and clicks on the submit button. 4. If the credentials entered are identical to the entry in the database, the user gets routed to the home screen of the application. 5. If the credentials entered are not identical to any entry in the database, the user will remain on the same screen and an error message will be displayed.	

Table 1.1 User Story 1 Login using Account Credentials

User Story 4

Title: View Profile Screen Credentials	**Priority**
User Story: As an employee of the company, I want to view my own credentials on a page so that I can check my credentials.	HIGH
Acceptance Criteria: 1. The user presses the profile button that is displayed on the home menu of the application. 2. The user will be displayed a profile page filled	

with the user's credentials. 3. The user is able to view their profile page, based on name, email, address, date of birth, department.	

Table 1.4 User Story 4 Check In from Profile Screen

User Story 5

Title: View Claims	**Priority**
User Story: As an employee of the company, I want to view the statuses of my claims that I have made, so that I am aware if my claims are successful or not	HIGH
Acceptance Criteria: 1. The User will be directed to the claims screen when he/she presses the claims icon on the bottom navigation bar. 2. The User will be able to view all his/her claims and its statuses on the claims screen.	

Table 1.5 User Story 5 View Claims

User Story 7

Title: Update Claim Expenses from Update Claims Screen	**Priority**
User Story: As an employee of the company, I want to update my claim expenses made during my trip out with the client.	HIGH
Acceptance Criteria: 1. The user will be directed to update the claims screen once he/she presses the update button on the respective claim entry. 2. When the User presses the submit button on the new claims screen after filling in required details to update the claim, a confirmation popup will be shown that claims have been updated. 3. When the User clicks "Ok" on the confirmation popup, he/she will be directed to the claims screen.	

Table 1.7 User Story 7 Update Claim Expenses from Update Claims Screen

User Story 9

Title: Clicks Logout from Logout Screen	**Priority**

User Story: As an employee, I am able to log out from the application so that I can log in from a different phone	HIGH
Acceptance Criteria: 1. The User will click on the logout button and it will send the user to the login screen.	

Table 1.9 User Story 9 Clicks Logout from Logout Screen

User Story 10

Title: Set Current Mood from Home Screen	**Priority**
User Story: As an employee, I want to be able to share my mood with Wickie.	HIGH
Acceptance Criteria: 1. The User will click on large wickie icon on the home screen, the screen will display a pop up dialog 2. The dialog will provide options to the user to set their mood (e.g “Happy” or “Tired”)	

Table 1.10 User Story 10 Set Current Mood from Home Screen

User Story 11

Title: Chatbot responses using Deep Learning	**Priority**
User Story: As an employee, I want the chatbot reply accurate responses so that it can provide the appropriate assistance to manage my mental-well being	HIGH
Acceptance Criteria: 1. User sends a message and the chatbot uses its model to understand the user’s message 2. Chatbot uses deep learning methods to understand and reply appropriate responses 3. User receives a helpful response depending on the situation	

Table 1.11 User Story 11 Chatbot Response using Deep Learning

User Story 12

Title: Clicks checked-out from Home Screen	**Priority**
User Story: As an employee of the company, I want to mark my attendance as “Checked Out” from the application’s home screen so that I can check out for the	HIGH

day.	
Acceptance Criteria: 2. User clicks on checked-in icon from home screen 3. System sends a display message, “Checked out successfully” 4. The checked-in icon changes to the checked-out icon after message “checked out successfully”	

Table 1.12 User Story 12 Clicks Checked-out from Home Screen

User Story 13

Title: Check-in Attendance from Profile Page	**Priority**
User Story: As an employee of the company, I want to check-in from the application’s profile screen so that I can keep track of my attendance.	HIGH
Acceptance Criteria: 5. User clicks on view attendance history icon from home screen 6. Database retrieves the attendance history and displays a list 7. User is able to view his/her own attendance history.	

Table 1.13 User Story 13 Clicks Checked-out from Profile Page

User Story 14

Title: User requires internet connection to use the application	**Priority**
User Story: As a user of the Wickie Application, I have to be connected to the Internet so that I can make full use of the Wickie.	HIGH
Acceptance Criteria: 8. Permissions will be displayed when using the camera module 9. Permissions will be displayed when using the fingerprint module 10. Permissions will be displayed when using the GPS module 11. When there is no internet connection, user is unable to login from the login page 12. When there is no internet connection when the	

user is currently using the application, a message prompt will be displayed to prompt the user to connect to a wireless network.	

Table 1.14 User Story 14 User requires internet connection to use the application

User Story 15

Title: Clicks checked-out from Profile Screen	Priority
User Story: As an employee of the company, I want to mark my attendance as “Checked Out” from the application’s profile screen so that I can check out for the day.	HIGH
Acceptance Criteria: 13. User clicks on checked-out icon from profile screen 14. System sends a display message, “Checked out successfully” 15. The checked-in icon changes to the checked-out icon after message “checked out successfully”	

Table 1.12 User Story 15 Clicks Checked-out from Home Screen

**Completed Tasks will be reflected on the github project board.

End of Sprint 3:
Updates on Product Backlog (Also on Github)

ID	Description	Sprint #	1	2	3
	Effort needed for Release 1 as in the beginning of the sprint		123	110	59
1	**Set up continuous integration system**		5	2	2
2	**Create compilable application skeleton**		5	3	3
3	**Display current UI in a simplest possible way**		5	3	3
4	**Set up Firebase as the database for the application**		7	7	2
5	**Design Log in system and Home Screen based on figma prototype**		0	0	0
Sprint 1	*Implement Firebase into HR mobile (Transfer all data into the app)*				
6	**Graphics Support on Client Side (Design XML skeleton) based on figma**		20	0	3
7	**Design drawable icons and related text on UI (Fingerprint/Login) (1st Feature)**		10	13	0
8	**Complete design of the Home screen with navigation bar**		20	8	3
9	**Complete CRUD functions with Firebase (Database) (2nd Feature)**		8	10	3
10	**Implement Camera Skeleton code for Claims**		2	10	3
11	**Implement GPS Skeleton code for Attendance**		10	20	3
Sprint 2	*Finalized minimal working version*				
12	**Complete design of Camera with full validation/exceptions testing (AR) (3rd Feature)**		10	13	13
13	**Complete design of GPS with full validation/exceptions testing (Attendance) (4th Fea**		8	8	8
14	**Implement Chatbot Skeleton**		5	5	5
15	**Complete design of Chatbot with full validation/exceptions testing (Social) (5th Featu**		8	8	8
16	Fix bugs/errors/error handling		20	20	20
17	Document and Finalize report for sprint 3		20	20	20
18	Document and Finalize presentation for sprint 3		40	40	40
Sprint 3	*Implement fully working version and present to the client*				
		Effort in the whole backlog	203	190	139

Color	Sprints	Description
	Sprint 1 is completed	Have Completed the Tasks of Sprint 1
	Sprint 2 is completed	Have Completed the Tasks of Sprint 2
	Sprint 3 is completed	Have Completed the Tasks of Sprint 3

Updates on Sprint Backlog (Also on Github)

US013 (Added for Sprint 3)	**As an employee of the company, I want to check in my attendance form my profile screen**	**Design Check-in Attendance Function in Profile Screen** **Retrieve List from Database** **Display List on UI** **Link UI to button on profile page**
US014 (Added for Sprint 3)	**As a user of the Wickie Application, I have to be connected to the Internet so that I can make full use of the Wickie.**	**Implement the Internet connection validation (check if connected to the** **Implement the Internet connection logic service (proceed if connected,** **Design the 'Not connected to Internet' UI popup** **Design the 'Connected to Internet' UI popup** **Set the time for popup window to appear**
US10 (Added for Sprint 2)	**As an employee, I want to be able to share my mood with Wickie.**	**- Design button to set current mood** **- Implement Logic to display mood on home screen** **- Send status to chatbot**
US11 (Added for Sprint 2)	**As an employee, I want the chatbot reply accurate responses so that it can provide the appropriate**	**- Research on Neural networks with chatbot implementation** **- Understand Deep Learning and Model Training for chatbot** **- Find a way to use database for model training** **- Find source codes with similar examples of chatbot**

Note: Added Bug fixes category in project board, final diagram architecture will be updated after all bug fixes and all test cases are done.

8.3 Initial Software Architecture

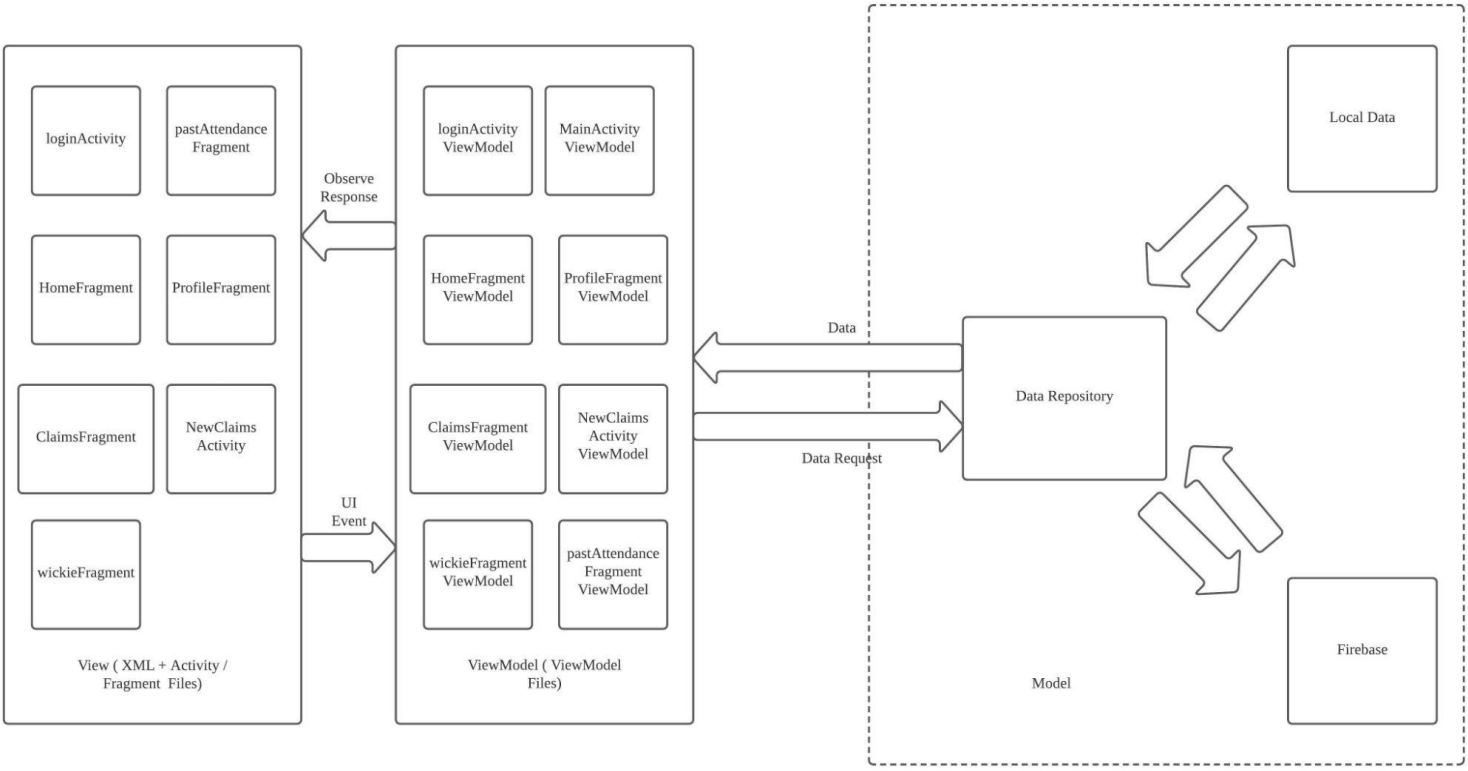


Fig 2 Initial Software Architecture

8.3.1 MVVM Architecture

The diagram above shows the overview of the application's software architecture and the related components needed for the HR app's implementation. The team has decided to deploy the application using the MVVM architecture as it is classified as the most appropriate for android's design in terms of the View-Model. MVVM architecture has a way to separate the View, in essence the Activities and Fragments from the business logic. The architecture is broken down into 3 layers which include:

Layer	Description
Model	This layer is responsible for data storage and manipulation. It contains a data repository which interacts with the data source. It also contains the both the remote and local data sources
View	The purpose of this layer is to inform the ViewModel about the user's action. This layer observes the ViewModel and does not contain any kind of application logic.

ViewModel	It exposes those data streams which are relevant to the View. Moreover, it serves as a link between the Model and the View. Keeps the data from being destroyed if a view doesn't require it anymore but another view is still using it.

Table 3 Architecture Layer Description

8.4 Initial UI Prototype

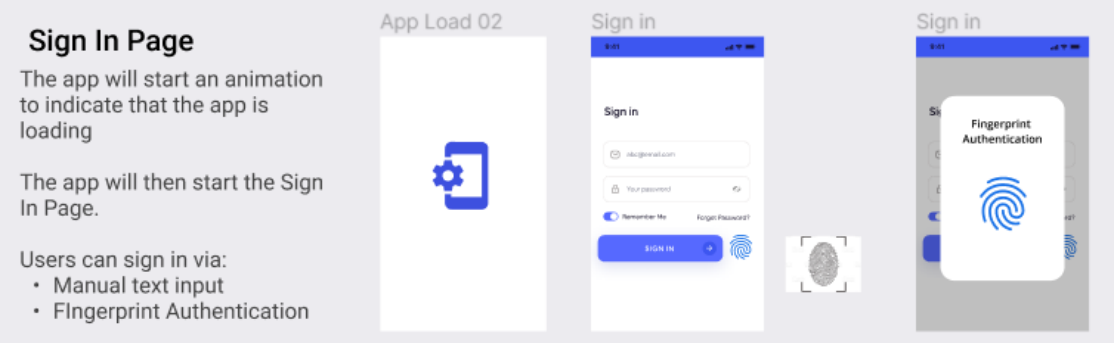


Fig 1.1 UI Prototype Sign In Page

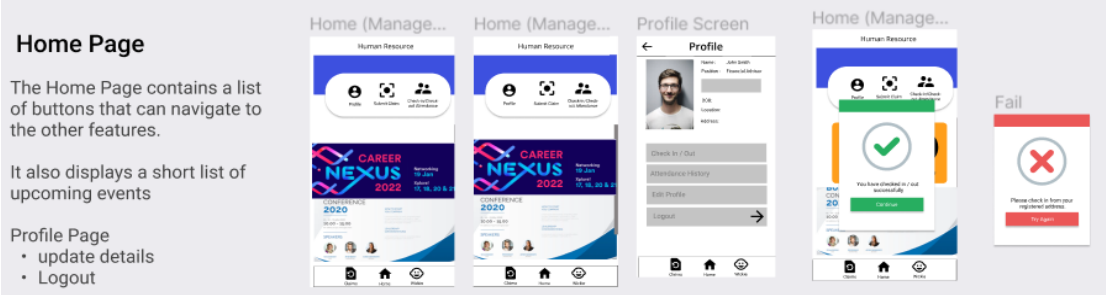


Fig 1.2 UI Prototype Home Page

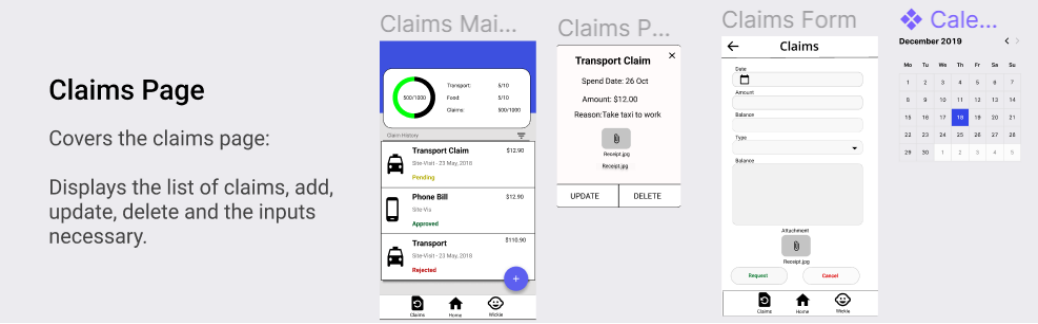


Fig 1.3 UI Prototype Claims Page

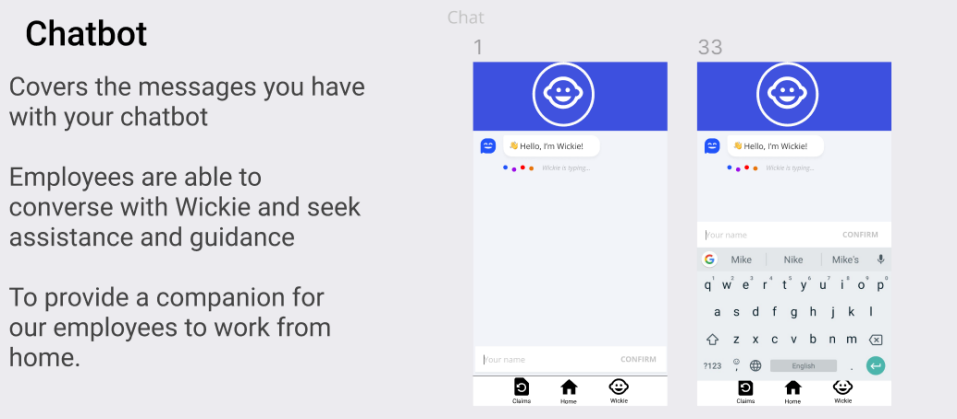


Fig 1.4 UI Prototype Chatbot

8.5 Sprint Backlog

UID	User Story	Task
US001	As an employee of the company, I want to log in to the company's HR application using my account credentials to access the services in it.	- Design base app with splash screen UI
		- Design login screen UI
		- Design tabs for email and password - Design submit button
		- Implement Validation for login
		- Implement Error Messages for login
		- Automate the tests
US002	As an employee of the company, I want to log in to the company's HR application using my fingerprint to access the services in it.	- Design Fingerprint scanner library - Implement Fingerprint scanner logic
		- Implement validation for fingerprint
		- Implement error messages for fingerprint
		- Automate the tests
US003	As an employee of the company, I want to check in on the application's home screen so that i can mark my attendance for the day.	- Design Check-in button UI on home screen
		- Implement GPS library to obtain current employee location.
		- Implement validation for check in / out
		- Store date and time of check in / out into database.
		- Design "Check-in" success message
		- Design "Check-in" unsucessful message
		- Automate the tests

Fig 3.1 Sprint Backlog (Part 1)

US004	**As an employee of the company, I want to check in on the application's profile screen so that I can mark my attendance for the day.**	**- Meet Up Client for Sprint 1**
		- Design Check-in button UI on profile screen
		- Reuse library and logic from User Story 3
		- Reuse success & fail messages from User Story 3
		- Automate the tests
US005	**As an employee of the company, I want to view my past checked in and out timing, so that i can calculate my pay for the month.**	- Design past check in / out screen.
		- Obtain past attendance date and time information from the database and display the information
		- Automate the tests
US006	**As an employee of the company, I want to view the statuses of my claims that I have made, so that I am aware if my claims are successful or not**	- Design statuses screen of the application
		- Obtain past claims information from the database
		- Display past claims information
		- Automate the tests
		- Meet Up Client for Sprint 2

Fig 3.2 Sprint Backlog (Part 2)

US007	**As an employee of the company, I want to claim my expenses made during my trip out with the client.**	- Design icon for the claim expenses
		- Implement Camera Library
		- Design new claim expenses screen
		- Store all entered data from employee to the database.
		- Automate the tests
US008	**As an employee of the company, I want to update my claim expenses made during my trip out with the client.**	- Design the icon for update claims
		- Design update claim screen
		- Implement update claim logic
		- Automate the tests
		- Meet Up Client for Sprint 3
US009	**As an employee of the company, I want to seek assistance and guidance for**	- Design Wickie Screen
		- Implement chatbot logic
		- Automate the tests
US10	**As an employee, I am able to log out from the application so that I can log in from a**	- Design Logout Screen
		- Implement logout logic
		- Automate the tests

Fig 3.3 Sprint Backlog (Part 3)

8.6 Product Backlog (Completed)

ID	Description	Sprint #	1	2	3
		Effort needed for Release 1 as in the beginning of the sprint	123	110	59
1	**Set up continuous integration system**		5	2	2
2	**Create compilable application skeleton**		5	3	3
3	**Display current UI in a simplest possible way**		5	3	3
4	**Set up Firebase as the database for the application**		7	7	2
5	**Design Log in system and Home Screen based on figma prototype**		0	0	0
Sprint 1	*Implement Firebase into HR mobile (Transfer all data into the app)*				
6	**Graphics Support on Client Side (Design XML skeleton) based on figma**		20	0	3
7	**Design drawable icons and related text on UI (Fingerprint/Login) (1st Feature)**		10	13	0
8	**Complete design of the Home screen with navigation bar**		20	8	3
9	**Complete CRUD functions with Firebase (Database) (2nd Feature)**		8	10	3
10	**Implement Camera Skeleton code for Claims**		2	10	3
11	**Implement GPS Skeleton code for Attendance**		10	20	3
Sprint 2	*Finalized minimal working version*				
12	**Complete design of Camera with full validation/exceptions testing (AR) (3rd Feature)**		10	13	13
13	**Complete design of GPS with full validation/exceptions testing (Attendance) (4th Fea**		8	8	8
14	**Implement Chatbot Skeleton**		5	5	5
15	**Complete design of Chatbot with full validation/exceptions testing (Social) (5th Featu**		8	8	8
16	**Fix bugs/errors/error handling**		20	20	20
17	**Document and Finalize report for sprint 3**		20	20	20
18	**Document and Finalize presentation for sprint 3**		40	40	40
Sprint 3	*Implement fully working version and present to the client*				

Fig 2 Product Backlog, Task in Sprint

8.7 Sprint Backlog (Completed)

UID	User Story	Task
US001	As an employee of the company, I want to log in to the company's HR application using my account credentials to access the services in it.	- Design base app with splash screen UI - Design login screen UI - Design tabs for email and password - Design submit button - Implement Validation for login - Implement Error Messages for login
US002	As an employee of the company, I want to log in to the company's HR application using my fingerprint to access the services in it.	- Design Fingerprint scanner library - Implement Fingerprint scanner logic - Implement validation for fingerprint - Implement error messages for fingerprint
US006	As an employee of the company, I want to claim my expenses made during my trip out with the client.	- Design icon for the claim expenses - Implement Camera Library - Design new claim expenses screen - Store all entered data from employee to the database.
US007	As an employee of the company, I want to update my claim expenses made during my trip out with the client.	- Design the icon for update claims - Design update claim screen - Implement update claim logic - Meet Up Client for Sprint 1
US003	As an employee of the company, I want to check in on the application's home screen so that i can mark my attendance for the day.	- Design Check-in button UI on home screen - Implement GPS library to obtain current employee location. - Implement validation for check in / out - Store date and time of check in / out into database. - Design "Check-in" success message - Design "Check-in" unsuccessful message - Automate the tests
US004	As an employee of the company, I want to view my own credentials on a page so that I can check my credentials.	- Design Profile UI on profile screen - Reuse Update Logic from CRUD - Reuse success & fail messages from User Story 3
US005	As an employee of the company, I want to view the statuses of my claims that I have made, so that I am aware if my claims are successful or not	- Design statuses screen of the application - Obtain past claims information from the database - Display past claims information
US008	As an employee of the company, I want to seek assistance and guidance for any issues that I am facing so that I can improve in my mental well-being as a worker of the company	- Design Wickie Screen UI - Implement and Integrate Chatbot into Application (Fully working version)
US009	As an employee, I am able to log out from the application so that I can log in from a different phone	- Design Logout Screen - Implement logout logic
US012 (Added for sprint 3)	As an employee of the company, I want to mark my attendance as "Checked Out" from the application's home screen so that I can check out for the day.	Design Check-Out button UI on home screen Implement Logic for setting "Check Out Status" Function Implementation of Setting Check Out Status Adding Timeout status onto Firebase RealTime Database Design "Check-Out" successful and unsuccessful message - Meet Up Client for Sprint 2
US013 (Added for Sprint 3)	As an employee of the company, I want to check in my attendance form my profile screen	Design Check-in Attendance Function in Profile Screen Retrieve List from Database Display List on UI Link UI to button on profile page
US014 (Added for Sprint 3)	As a user of the Wickie Application, I have to be connected to the Internet so that I can make full use of the Wickie.	Implement the Internet connection validation (check if connected to the Implement the Internet connection logic service (proceed if connected, Design the 'Not connected to Internet' UI popup Design the 'Connected to Internet' UI popup Set the time for popup window to appear
US10 (Added for Sprint 2)	As an employee, I want to be able to share my mood with Wickie.	- Design button to set current mood - Implement Logic to display mood on home screen - Send status to chatbot
US11 (Added for Sprint 2)	As an employee, I want the chatbot reply accurate responses so that it can provide the appropriate	- Research on Neural networks with chatbot implementation - Understand Deep Learning and Model Training for chatbot - Find a way to use database for model training - Find source codes with similar examples of chatbot

8.8 Chatbot Appendices Details

learning.py | intents.json | ChatBotAPI.py

```
{"intents": [
        {"tag": "greeting",
         "patterns": ["Hi Wickie!", "Are you there Wickie?", "Hello Wickie", "Good Morning Wickie"],
         "responses": ["Good to see you! How are you feeling? Say: I am feeling great if you feel happy, or I am feeling sad if you feel sad?", "Happy
         "context_set": ""
        },
        {"tag": "Exercise",
         "patterns": [ "Exercise Done", "I am done with the exercise"],
         "responses": ["Awesome! Are you feeling better? Say: I'm feeling better or I do not feel better"],
         "context_set": ""
        },
        {"tag": "goodbye",
         "patterns": ["Thank you", "Bye", "See you later", "Goodbye", "I am Leaving", "Have a Good day"],
         "responses": ["Happy to help! Talk to you later!", "Okaee, >_< talk to you later!", "Goodbyeee! <3"],
         "context_set": ""
        },
        {"tag": "sad",
         "patterns": ["I am feeling really sad", "I am very tired", "I am facing some issues", "I need help", "I am depressed"],
         "responses": ["Oh no! What seems to be the problem? Your mental-wellbeing is very important to me, Is it stress? Is it health-related?", "Oh
         "context_set": ""
        },
        {"tag": "name",
         "patterns": ["what is your name", "what should I call you", "whats your name?"],
         "responses": ["You can call me Wickie.", "I'm Wickie!", "I'm Wickie aka Wiki Nuts hehe."],
         "context_set": ""
        },
        {"tag": "stress",
         "patterns": ["Yeah, I think its stress", "Yep, it is stress", "Yeah, been stressful lately", "ya i feel stress sia"],
         "responses": ["Oh dear, how about we take a 5 minutes break from your work, let's do a short breathing exercise! Say Okay! I am ready to beg
```

Intents json file contains the patterns of the user's replies and responses of the user and the chatbot. These strings of text will be used for model training using tensorflow on the following code learning.py:

learning.py | intents.json | ChatBotAPI.py

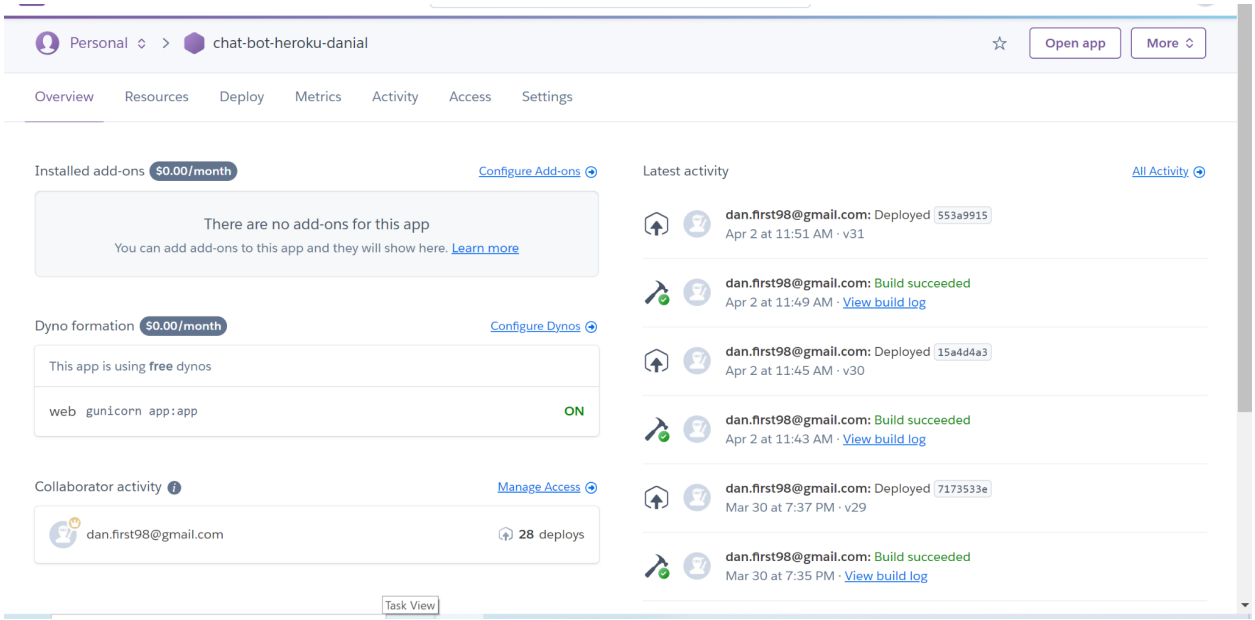
Package requirements 'click==8.0.4', 'Flask==2.0.3', 'gunicorn==20.1.0', 'itsdangerous==2.1.1', 'MarkupSafe==2.1.1', 'Werkzeug==2.0.3', 'tensorflow-cpu==2.8.0' are not satisfied Install requirements Ignore requirements

```
from flask import Flask, request, jsonify

from learning import chatWithBot
import unicodedata

app = Flask(__name__)


@app.route('/chat', methods=['GET', 'POST'])
def chatBot():
    chatInput = request.form['chatInput']
    # unicodedata.normalize('NFKD', chatInput).encode('ascii', 'ignore')
    return jsonify(chatBotReply=chatWithBot(chatInput))


if __name__ == '__main__':
    # app.run(debug=True)
    app.run(debug=True)
```

chatBot()

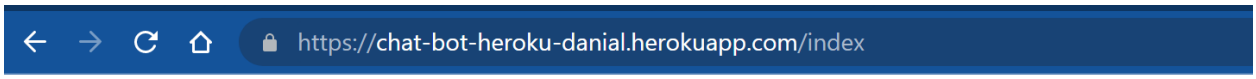
9: Git | TODO | 6: Problems | Terminal | Python Console | Event Log

In this code, the ChatbotAPI.py launches the chatbot function into python flask. This code will then be used to be deployed in the Heroku web server.




```
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dania>cd C:\Work\SIT YEAR 2\ICT2102 - Human Computer Interaction\github\mylab2

C:\Work\SIT YEAR 2\ICT2102 - Human Computer Interaction\github\mylab2>env\Scripts\activate

(env) C:\Work\SIT YEAR 2\ICT2102 - Human Computer Interaction\github\mylab2>heroku -- logs
 »   Warning: -- is not a heroku command.
Did you mean ps? [y/n]: n
 »   Error: Run heroku help for a list of available commands.

(env) C:\Work\SIT YEAR 2\ICT2102 - Human Computer Interaction\github\mylab2>heroku --logs
 »   Warning: --logs is not a heroku command.
Did you mean logs? [y/n]: y
2022-04-09T07:05:08.723021+00:00 app[web.1]:
2022-04-09T07:05:08.723987+00:00 app[web.1]: Epoch 962/1000
2022-04-09T07:05:08.726416+00:00 app[web.1]:
2022-04-09T07:05:08.726569+00:00 app[web.1]: Epoch 994/1000
2022-04-09T07:05:08.730832+00:00 app[web.1]:
2022-04-09T07:05:08.731804+00:00 app[web.1]: Epoch 963/1000
2022-04-09T07:05:08.732178+00:00 app[web.1]:
2022-04-09T07:05:08.733011+00:00 app[web.1]: Epoch 995/1000
2022-04-09T07:05:08.737992+00:00 app[web.1]:
2022-04-09T07:05:08.738916+00:00 app[web.1]: Epoch 964/1000
2022-04-09T07:05:08.739828+00:00 app[web.1]:
2022-04-09T07:05:08.740762+00:00 app[web.1]:
2022-04-09T07:05:08.744750+00:00 app[web.1]:
2022-04-09T07:05:08.745545+00:00 app[web.1]: Epoch 965/1000
2022-04-09T07:05:08.749737+00:00 app[web.1]:
2022-04-09T07:05:08.750865+00:00 app[web.1]: Epoch 997/1000
2022-04-09T07:05:08.751640+00:00 app[web.1]:
2022-04-09T07:05:08.753017+00:00 app[web.1]:
2022-04-09T07:05:08.756005+00:00 app[web.1]:
2022-04-09T07:05:08.756859+00:00 app[web.1]: Epoch 998/1000
```

```
2022-04-09T07:05:08.963883+00:00 app[web.1]:
2022-04-09T07:05:08.972392+00:00 app[web.1]: Saved model from disk
2022-04-09T07:05:09.111109+00:00 heroku[router]: at=info method=GET path="/favicon.ico" host=chat-bot-heroku-danial.herokuapp.com request_id=0c51502a-4f13-461a-ba05-b5892ff87f30 fwd="222.164.6.177" dyno=web.1 connect=0ms service=1ms status=404 bytes=393 protocol=https
2022-04-09T07:05:09.111422+00:00 app[web.1]: 10.1.81.88 - - [09/Apr/2022:07:05:09 +0000] "GET /favicon.ico HTTP/1.1" 404 232 "https://chat-bot-heroku-danial.herokuapp.com/index" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.75 Safari/537.36"
```



HEROKU

In the heroku web server, once the code is deployed, it will run the script and we can see the result by creating an environment through python flask in the command line. Once the environment is created, run heroku logs –tail, to see the details of the server, including get and post requests that can be tested using POSTMAN.

learning.py source code

```
import json
import pickle
import random

import nltk
import numpy
from nltk.stem import LancasterStemmer
from tensorflow.python.keras.layers import Dense
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.models import model_from_yaml

nltk.download('punkt')

stemmer = LancasterStemmer()

with open("intents.json") as file:
    data = json.load(file)
```

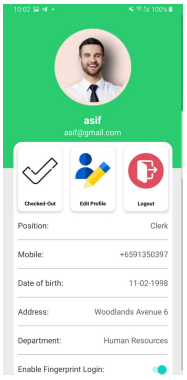
```
try:
    with open("chatbot.pickle", "rb") as file:
        words, labels, training, output = pickle.load(file)

except:
    words = []
    labels = []
    docs_x = []
    docs_y = []

    for intent in data["intents"]:
        for pattern in intent["patterns"]:
            wrds = nltk.word_tokenize(pattern)
            words.extend(wrds)
            docs_x.append(wrds)
            docs_y.append(intent["tag"])

        if intent["tag"] not in labels:
            labels.append(intent["tag"])

    words = [stemmer.stem(w.lower()) for w in words if w != "?"]
    words = sorted(list(set(words)))

    labels = sorted(labels)

    training = []
    output = []
```

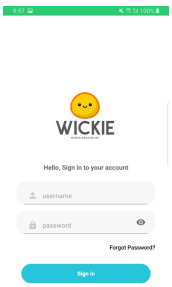
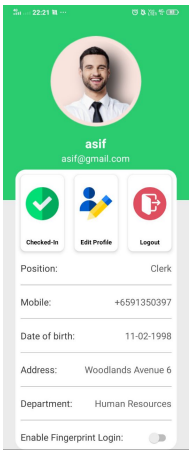
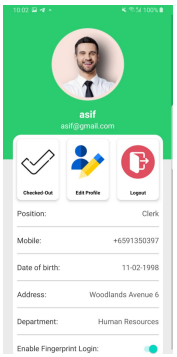
```
    output_empty = [0 for _ in range(len(labels))]

    for x, doc in enumerate(docs_x):
        bag = []

        wrds = [stemmer.stem(w.lower()) for w in doc]

        for w in words:...

        output_row = output_empty[:]
        output_row[labels.index(docs_y[x])] = 1

        training.append(bag)
        output.append(output_row)

    training = numpy.array(training)
    output = numpy.array(output)

    with open("chatbot.pickle", "wb") as file:
        pickle.dump((words, labels, training, output), file)


try:
    json_file = open('chatbotmodel.json', 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    myChatModel = model_from_yaml(loaded_model_json)
    myChatModel.load_weights("chatbotmodel.h5")
    print("Loaded model from disk")


except:
    # Make our neural network
    myChatModel = Sequential()
    myChatModel.add(Dense(8, input_shape=[len(words)], activation='relu'))
    myChatModel.add(Dense(len(labels), activation='softmax'))

    # optimize the model
    myChatModel.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

    # train the model
    myChatModel.fit(training, output, epochs=1000, batch_size=8)

    # serialize model to yaml and save it to disk
    model_json = myChatModel.to_json()
    with open("chatbotmodel.json", "w") as json_file:
        json_file.write(model_json)

    # serialize weights to HDF5
    myChatModel.save_weights("chatbotmodel.h5")
    print("Saved model from disk")


def bag_of_words(s, words):
    bag = [0 for _ in range(len(words))]

    s_words = nltk.word_tokenize(s)
    s_words = [stemmer.stem(word.lower()) for word in s_words]

    for se in s_words:
        for i, w in enumerate(words):
            if w == se:
                bag[i] = 1

    return numpy.array(bag)
```

```
def chatWithBot(inputText):
    currentText = bag_of_words(inputText, words)
    currentTextArray = [currentText]
    numpyCurrentText = numpy.array(currentTextArray)

    if numpy.all((numpyCurrentText == 0)):
        return "I didn't get that, try again"

    result = myChatModel.predict(numpyCurrentText[0:1])
    result_index = numpy.argmax(result)
    tag = labels[result_index]

    if result[0][result_index] > 0.7:
        for tg in data["intents"]:
            if tg['tag'] == tag:
                responses = tg['responses']

        return random.choice(responses)

    else:
        return "I didn't get that, try again"
```

The learning.py file is the main source code of how the model is being trained. First off, we would need a series of inputs to train the model. We load the json file into a variable called data. We will load the “intents.json” file and identify the tags and patterns accordingly. Each of the patterns and tags contains the strings and we proceed to make the neural network, linking each tag to a certain string. Once we have the neural network, then we will optimize the model, train the model, serialize the model and save it into a disk. The disk is named chatbotmodel.h5, and this is our fully trained model we are going to use. In order for the model to identify text, the bag_of_words function uses numPy to categorize the text in a series of patterns where the model will be able to understand. Hence, if the tag and patterns match accordingly, it will return the string of text randomly in the list of responses filled in the json file.

8.9 Test Cases

Test ID	US ID	Case Description	Precondition	Steps	Screenshot	Status (Pass / Fail)
9	4	View Profile Screen credentials	Logged into Asif account	1.Click on User Profile 2. User redirected to Profile page		Pass

10		Logout from Profile page	Logged in	1. User clicks on “User Profile” page 2. User clicks on Logout button 3. User redirected Login page		Pass
11		Check-in attendance from Profile Page	User checked-out Profile Page	1. User clicks on the “Check-out” button in the Home Page 2. User check-in status updated to “Checked-in”		
12	15	Check-out from Profile Page	User checked-in Profile Page	1. User clicks on the “Check-In” button in the Home Page 2. User check-in status updated to “Checked-out”		Pass

9. References

NUHS
https://www.straitstimes.com/singapore/health/more-work-from-homers-feel-stressed-than-front-line-workers-singapore-survey-on

Good HR

https://www.techtarget.com/searchhrsoftware/definition/human-resource-management-HRM#:~:text=Make%20sure%20employees%20have%20or,rules%20and%20regulations%20to%20employees.

https://developer.android.com/jetpack/guide
https://www.toptal.com/android/android-apps-mvvm-with-clean-architecture
https://www.figma.com/file/2Hz7cOgcMxFgcJYGF85D7n/ICT2105-UI-prototype?node-id=17%3A3740
https://www.youtube.com/watch?v=8Pv96bvBJL4&ab_channel=GeeksforGeeks
https://www.youtube.com/watch?v=4xlukt7GV9Y&ab_channel=Dr.ParagShukla

MVVM
https://lucid.app/lucidchart/519e5412-5373-4b0b-ac34-b2105c2ff78c/edit?invitationId=inv_b8d73ddf-0c10-4f4c-811e-3eecb31bb418

Branching Model
https://lucid.app/lucidchart/c53b18f1-c966-4fc0-8f08-73964a9ef6c0/edit?invitationId=inv_5d827857-909a-4e1a-8636-10df3e18b8dc&page=HaD9xOeFGEr9#