

**GROUP MEMBERS**

**ARMANDO BUFACCHI- ST10092068**

**CORNELIUS ACKERMAN-ST10092070**

**NOMASWAZI SIBEKO-ST10246300**

**RYAN BLIGNAUT-ST10092012**

**PATHWAY: XBCAD7319**

**LECTURER: HANDSOME MPOFU**

**WORK INTEGRATED LEARNING**

**SUBMISSION DATE: 28 NOVEMBER 2023**

**FINAL POE**

**SCRUM AGILE BOARD: [Board - Rainfall - Jira \(atlassian.net\)](#)**

**GITHUB: <https://github.com/IIEWFL/final-poe-presentation-submission-rain-tracking-plant-n-bietjie>**

**SECURITY REPORT: [Rainfall - AppSweep \(guardsquare.com\)](#)**

**BUILD REPORT: <https://github.com/IIEWFL/final-poe-presentation-submission-rain-tracking-plant-n-bietjie/actions/runs/7005748377>**

<b>TABLE OF FIGURES .....</b>	<b>4</b>
<b>1. INTRODUCTION TO THE PROJECT.....</b>	<b>5</b>
1.1. Work Agreement.....	6
1.2. Definition of Ready (DoR) .....	6
1.3. Definition of Done (DoD) .....	9
1.4. RoadMap (High-Level Plan) .....	11
<b>2. FUNCTIONAL REQUIREMENTS.....</b>	<b>15</b>
2.1. User Roles.....	15
2.2. User Experience Journey Maps .....	17
<b>3. NON-FUNCTIONAL REQUIREMENTS.....</b>	<b>18</b>
<b>4. ANALYSIS ARTIFACTS .....</b>	<b>20</b>
<b>4.1. DOMAIN MODELLING .....</b>	<b>20</b>
BOUNDED CONTEXT 1: WEATHER TRACKING .....	20
BOUNDED CONTEXT 2: RAIN LOGGING .....	20
BOUNDED CONTEXT 3: DATA MANAGEMENT AND SECURITY .....	20
BOUNDED CONTEXT 4: FIRE RISK ANTICIPATION .....	20
<b>5. SOFTWARE IMPLEMENTATION .....</b>	<b>22</b>
<b>6. DATA SCHEMAS .....</b>	<b>24</b>
Introduction .....	24
User Table.....	24
Notifications Table.....	24
Preferences Table.....	24
Location Table .....	25
Rainfall Table.....	25
Google Firebase .....	25
ERD Diagram .....	26
JSON Schema .....	27
JSON Usage Example .....	29
<b>7. ARCHITECTURE ARTIFACTS.....</b>	<b>30</b>
<b>8. SECURITY .....</b>	<b>34</b>

<b>9. DEVOPS.....</b>	<b>36</b>
<b>10. RUNNING COSTS .....</b>	<b>38</b>
<b>11. CHANGE MANAGEMENT .....</b>	<b>42</b>
<b>12. APPENDICES.....</b>	<b>45</b>
<b>REFERENCE LIST.....</b>	<b>51</b>

## TABLE OF FIGURES

Figure 1 High-level Roadmap.....	11
Figure 2 User Story Epic.....	16
Figure(s) 3 User Experience Journey Maps.....	17
Figure 4 Figure 4 UML Package Diagram.....	21
Figure 5 UML Object Diagram.....	22
Figure 6 UML Sequence Diagram.....	23
Figure 7 UML State Diagram.....	23
Figure 8 ERD Diagram.....	26
Figure 9 Architecture Artifact Diagram.....	30
Figure 10 GitHub Pipeline Diagram.....	33
Figure 11 Best Case Scenario Graph.....	35
Figure 12 Mean Case Scenario Graph.....	36
Figure 13 Worst Case Scenario Graph.....	36

# 1. INTRODUCTION TO THE PROJECT

The Client has requested us to design a Rain detecting Mobile application that will meet the expectations of his Farming business. The name of the mobile application agreed upon by the team is “RAINFALL”. Below is an introduction to the rainfall application.

Introducing RainFall, a comprehensive weather tracking application designed to meet your needs. RainFall allows you to log rainfall data efficiently, including mm measurements, dates, and customizable time frames, all while accommodating multiple logging locations. You can even enhance your experience by setting location coordinates for weather information and fire risk assessment. Our user-friendly interface comes with intuitive visuals, such as weather maps and customizable rain log graphs in both line and bar formats, with options to view data over different time periods like weekly, monthly, yearly, or user-defined rain seasons.

Our App offers robust backup options, including secure cloud backup with user accounts and convenient local data export and import features. It's designed to be functional both online and offline, ensuring you're never caught unprepared. You have the flexibility to define your unique rain season within the settings, making RainFall adaptable to various climate patterns.

Our app's aesthetics prioritize a visually pleasing and user-friendly experience, and the home page displays local weather information, either based on your preference or your device's location. Stay informed about weather conditions and potential risks through push notifications for high rain probabilities and fire risks (internet connection required). In case of fire risks, you can rely on our app's informative page for essential knowledge on handling such situations.

RainFall is a multilingual app, supporting Afrikaans, English, and another commonly used language in South Africa. Additionally, our About page within the settings provides insights into the app's purpose, development, and contact information. You can even share your rain data effortlessly by exporting visual formats, such as images. This app is your complete weather companion, offering all the tools you need to track and manage rainfall effectively while keeping you informed and safe. Download it now and take control of your local weather knowledge. Stay dry, stay safe with RainFall!

### 1.1. **Work Agreement**

- We will meet every week online, on the discord group created for this project.
- We will mostly work in teams of two per task, and communicate privately to stay updated.
- We will do small feedback sessions for each item completed and milestone reached.
- All feedback will be critical and not sugar-coated to ensure our work is of high-quality.
- We will always be willing to help each other, even if the task is not assigned to the person(s) asked.
- We will always prioritise finishing tasks earlier than the deadlines set during the weekly meetings to ensure no time issues.
- Two people will be in control of the application mainly, while the other two will focus on the documentation mainly, but we all will work interchangeably on everything to ensure a good mix of experience and knowledge.

### 1.2. **Definition of Ready (DoR)**

#### Description

We have consulted with the client and deduced the main features and functionalities of the rain tracking application, as well as requirements and non-functional requirements. The app should have four main core functionalities: Rain Logging, Data Safety and Availability, Weather tracking, and lastly the anticipation of Fire Risks. To be able to deliver this application, we would need to meet each functional and non-functional requirement deliverable, before entering the final phase where our application is complete. Mitchell (2017) suggests that the Definition of Ready would then be to meet each individual requirement first and then focus on non-functional requirements with UI development taking place thereafter. Once each requirement has been accomplished, the team will be able to move on to the next.

To understand the definition of ready for Rainfall, one must consider the four main activities of functions that contribute to the functionality of the application:

- Rain Logging
- Data Safety and Availability
- Weather
- Fire Risk

These four core functionalities are based on the stipulated user stories provided by the client as to what they desired in the applications functionality. From different user perspectives that being Regular and Farm users respectively. The Definition of Ready would then address the team's ability to take immediate action on the user stories.

### Rain Logging and Weather

- ✓ Would a user be able to add a rain log and access it? Can a user edit and remove a rain log? This would mean that the team would have to focus on making sure that a user can at the very least add a rain log and view it, before implementing location-based rain logging.
- ✓ Would a user be able to have multiple locations for rain logging? Can a user select more than one location for a rain log? So, the team would need to ensure that after a log is created that another log can be added separately to the other log. On top of this a user must then be able to switch between the logs without much difficulty. This would also include that the user be able to add rain logs for different areas. So, to meet this the minimum requirement would be to select multiple locations and add rain logs to them.
- ✓ Should a user want to view the rain log, would it be able to be presented as a line and or bar graph? And would the user be able to switch between them? The team would need to then focus on the visualization of the rain logs and then bring them to graph format. But then how would this data be analyzed? Would for instance a farm user then be able to set custom start and end dates to rain seasons to then view the data? This would mean that the functionality that must be achieved at the bare minimum is that both users can display and analyze rain log data before moving on to the next requirement.
- ✓ Rain notification would be the next step, for a user wanted to receive a notification on what the probability of rain is, and when. This means that this would require a notification-based system to be implemented and tap into local weather forecast to predict the likelihood of rain.
- ✓ A farm user may want to visualize local weather data on a map. This means the team would need to implement intents to take a user to a map and then overlay with live or offline weather data.

### Data Safety and Availability

- ✓ A Regular user would want a backup locally and on cloud that they can use to recover an account. This would mean the team would need to implement import/export functionality. So, in this case a user must be able to create an account and reload an account if the app was in this case uninstalled. Once this is complete, the team can move on to the next step of offline functionality.

1. A Farm user may want offline functionality in the case of not being able to have access to the internet or good reception. So, the team would need to make sure that the app can also download any online display data and then taking it offline to view at a later time.

### **Fire Risk Management.**

- ✓ A Regular user may want notifications on up-coming fire risks nearby to them. This would mean that like the weather the team will need to make use of external functionality to display to a user that a fire is coming through the app's notifications. At the very least notifications must be sent.
- ✓ A farm user could make use of a fire risk information page to learn more about dealing with and preventing them. The bare minimum would be to link the page to fire risk information but optimally the user would be able to view this page on the app and then read up more on it externally.



### 1.3. Definition of Done (DoD)

According to Wrike (2023), the definition of done would be based off certain criteria on how a requirement is implemented before moving on to the next requirement, and then furthermore to the next section.

The team's definition of done would be achieved through meeting the requirements stipulated by the user through the user stories. Each activity and goal would have to meet the requirements for the task to be done.

#### **Rain Logging**

A user can successfully add a rain log tracking the amount of rain within date and time frames and have access to the log while also being able to set seasons for the rain tracking. The user can also add multiple locations for rain logs with separate information while having effortless switching between the logs. A user must also be able to view the logs in the form of a graph, either line graph or bar graph thus allowing a user to analyze the data within specified time periods. Lastly a user should be able to share a graph with other users.

- ✓ Code for data collection has been implemented.
- ✓ Data storage code has also been implemented for the functionality of rain logging.
- ✓ Data validation to validate the rainfall data.
- ✓ Error handling has been implemented.

#### **Data Safety and Availability**

To achieve the definition of done for data Safety and availability a user would need to be able to perform two main functions, being User account backup and restoration as well as local as local backup and restoration. As part of these two functions a user should easily be able to create and delete an account while also being able to restore it from a local or cloud backup file. They must also be able to view account details and share the backup file.

- ✓ Data encryption code has been implemented for the functionality of the app.
- ✓ Code to backup data has been added and tested.
- ✓ Caching mechanism has been implemented in the application.

## Weather

The three functions contributing to weather are viewing weather, weather notifications and Weather map visualization. This means to be able to present this feature to the client, a user must then be able to view the weather based on a devices location as well as viewing it from any other user specified location. In addition to this a user should be able to receive local and determined location-based notifications on the weather. A user should also be able to view a map with weather overlay. With the notifications then in place a user should be able to view warnings and enable or disable the notifications.

- ✓ Code to view the weather based on a device's location as well as viewing it from any other user-specified location has been implemented.
- ✓ Code to receive local and determined location-based notifications on the weather has been added.
- ✓ Code to view a map with weather overlay has been achieved.

## Fire Risks

Fire risk's activities evolve around being able to receive Fire Risk notifications, Viewing Fire Risk Information and Map Visualization. A user must be able to set custom notifications for location-based fire risks and receive notifications of fires nearby. The application will contain a basic information page about fires and the prevention and guidelines in case thereof. A user can also visualize fires on a map similarly to how the weather is visualized. Notifications can once again be enabled or disabled, and a user can also look at international fire news feeds.

- ✓ Code to set custom notifications for location-based fire risks and receive notifications of fires nearby has been achieved.
- ✓ The team has implemented the code to visualize fires on a map similarly to how the weather is visualized.
- ✓ Code to enable and disable notifications has been added.
- ✓ Code to view international news feeds has been successfully achieved.

## 1.4. RoadMap (High-Level Plan)

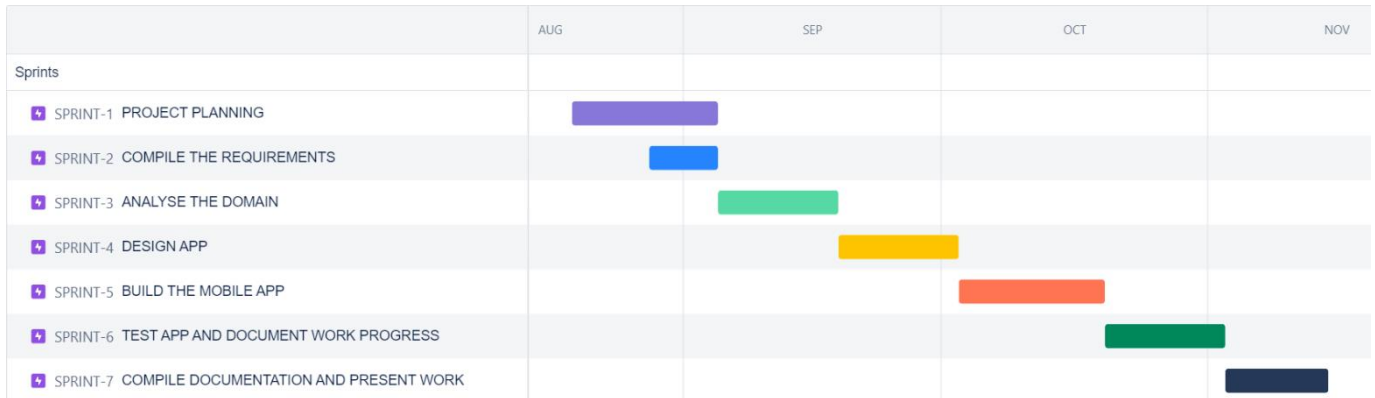


Figure 1 High-level Roadmap

## HIGH-LEVEL PLAN

### SPRINT 1: PROJECT PLANNING

#### AGILE LIFECYCLE PHASE: CONCEPT

**SPRINT GOAL:** The sprint goal would be the complete compilation of the requirements and the team understanding thereof. Our team consists of four members, namely Armando Bufacchi, Cornè Ackerman, Ryan Blignaut and Nomaswazi Sibeko. Each of us specializes in different aspects. Nomaswazi and Armando have opted to do majority of the work appertaining to Documentation and overall GUI design. Ryan is our lead programmer and Cornè has been the one who secured us the client and acts as our communication between the client and the team, he will also work alongside Ryan in making sure the Backend of our App is operating at its full capacity. We all had a say in the work agreement and chose our roles respectively. Armando and Nomaswazi delegated and worked together on the Definition of Ready (DoR) and the Definition of Done (DoD) furthermore the mentioned members also created the roadmap (High-level plan). Cornè designed the User Experience Journey map, as well as gathering all our requirements both functional and non-functional from the client and detailing them in the form of a user story epic. Ryan is working on our backend.

#### TEAM AVAILABILITY:

ARMANDO BUFACCHI- 19/08/2023

CORNELIUS ACKERMAN-19/08/2023

NOMASWAZI SIBEKO-19/08/2023

RYAN BLIGNAUT-19/08/2023

## **SPRINT 2: Compile the Requirements**

### **AGILE LIFECYCLE PHASE: Inception**

**SPRINT GOAL:** The sprint goal would be to have a meeting with the client to further discuss his requirements for the application, in order to meet his business expectations. And for the team to compile a list of those requirements to then proceed with the beginning phases of coding and implementation. Cornè took the liberty to meet with the client and discuss the basic, functional, and non-functional requirements for the application, thereafter, detailing it in the user stories epic, as well as creating a user experience journey map based on the aforementioned requirements. We may have to consider the technological implications in terms of different architecture patterns and styles, further research will be conducted.

### **TEAM AVAILABILITY:**

ARMANDO BUFACCHI- 28/08/2023

CORNELIUS ACKERMAN-28/08/2023

NOMASWAZI SIBEKO-28/08/2023

RYAN BLIGNAUT-28/08/2023

## **SPRINT 3: Analyse the Domain**

### **AGILE LIFECYCLE PHASE: Iteration**

**SPRINT GOAL:** The purpose of this sprint is further discuss the functional and non-functional requirements of the mobile application so that it meets the client's expectations and ensuring that the application is functioning efficiently. The team still needs to discuss the architecture of the application and the technology choices so they can be documented furthermore, the team will need to also design a mock-up of the application and present it to the client.

### **TEAM AVAILABILITY:**

ARMANDO BUFACCHI- 14/09/2023

CORNELIUS ACKERMAN-14/09/2023

NOMASWAZI SIBEKO-14/09/2023

RYAN BLIGNAUT-14/09/2023

#### **SPRINT 4: App design and Project Structure**

##### **AGILE LIFECYCLE PHASE: Iteration**

**SPRINT GOAL:** Once the team has presented the mock-up to the client and have received feedback from him, they can then begin with using the given feedback to design the application and give structure to the project.

##### **TEAM AVAILABILITY:**

ARMANDO BUFACCHI- 18/09/2023

CORNELIUS ACKERMAN-18/09/2023

NOMASWAZI SIBEKO-18/09/2023

RYAN BLIGNAUT-18/09/2023

#### **SPRINT 5: Build the Mobile App**

##### **AGILE LIFECYCLE PHASE: Iteration**

**SPRINT GOAL:** The team should now begin building the prototype of the mobile application as per the requirements using the agreed software (Android Studio) to build the app this should be able to begin after the User stories have been gathered as well. An implementation of the DevSecOps CI/ID Pipeline would be crucial for the testing and deployment of the application.

##### **TEAM AVAILABILITY:**

ARMANDO BUFACCHI- 2/10/2023

CORNELIUS ACKERMAN-2/10/2023

NOMASWAZI SIBEKO-2/10/2023

RYAN BLIGNAUT-2/10/2023

## **SPRINT 6: Test App and Document Work Progress**

### **AGILE LIFECYCLE PHASE: Release**

**SPRINT GOAL:** Continuation with the Build of the Application until it reaches its state of completion. Thereafter the team members will have to test whether the application is working to its full capacity. After the app has been tested, it can further be deployed for release.

### **TEAM AVAILABILITY:**

ARMANDO BUFACCHI- 2/10/2023

CORNELIUS ACKERMAN-2/10/2023

NOMASWAZI SIBEKO-2/10/2023

RYAN BLIGNAUT-2/10/2023

## **SPRINT 7: Compile Documentation and Present Work**

### **AGILE LIFECYCLE PHASE: Maintain**

**SPRINT GOAL:** At this stage of the project, the application would've been tested and deployed and then maintained to ensure that it is ready to be presented furthermore the team will need to compile the whole project document and finalize the POE document.

### **TEAM AVAILABILITY:**

ARMANDO BUFACCHI- 13/11/2023 – 28/11/2023

CORNELIUS ACKERMAN-11/10/2023 – 28/11/2023

NOMASWAZI SIBEKO-13/11/2023 – 28/11/2023

RYAN BLIGNAUT-13/11/2023 – 28/11/2023

## 2. FUNCTIONAL REQUIREMENTS

### 2.1. User Roles

#### **Regular User:**

This user role is the general user of the app, they will only want to use the basic features of the app, such as logging rainfall and sometimes viewing local weather.

#### **Farmer User:**

This user role represents users who are primarily in more rural areas, that need additional features related to the area they live in, primarily fire risks and viewing cloud coverage data.

#### **User Stories**

- As a Regular User, I want to log the amount of rain, along with the date and time frame, so that I can track rainfall in my area.
- As a Regular User, I want to have multiple rain logging locations, each containing separate information, and be able to switch between them easily, so that I can track rainfall in different areas.
- As a Farmer User, I want to visualize weather data on a map, so that I can have a graphical representation of the weather conditions.
- As a Regular User, I want to have backup functionality, including cloud backup with a user account and local export/import options, to ensure data safety and availability.
- As a Farmer user, I want the app to be functional without an internet connection, so that I can use it even in areas with no connectivity.
- As a Regular User, I want to view rain logs in both line graph and bar graph visualizations, with the ability to select different time periods, so that I can analyse rainfall data effectively.
- As a Farmer User, I want to set my own rain season start- and end-dates, and have pre-set rain seasons as options, so that I can customize the data display to my needs.
- As a Regular User, I want to receive notifications of high rainfall probability, so that I can know in advance when rain is coming.
- As a Farmer User, I want to receive notifications of fire risks, so that I can prepare before it arrives.
- As a Farmer User, I want to be able to inform and educate myself on rain risks, so I can better understand what to do and how it works.
- As a Regular User, I want the app's interface to have a modern and intuitive design, with user-friendly navigation, to enhance usability.

# User Story Epic

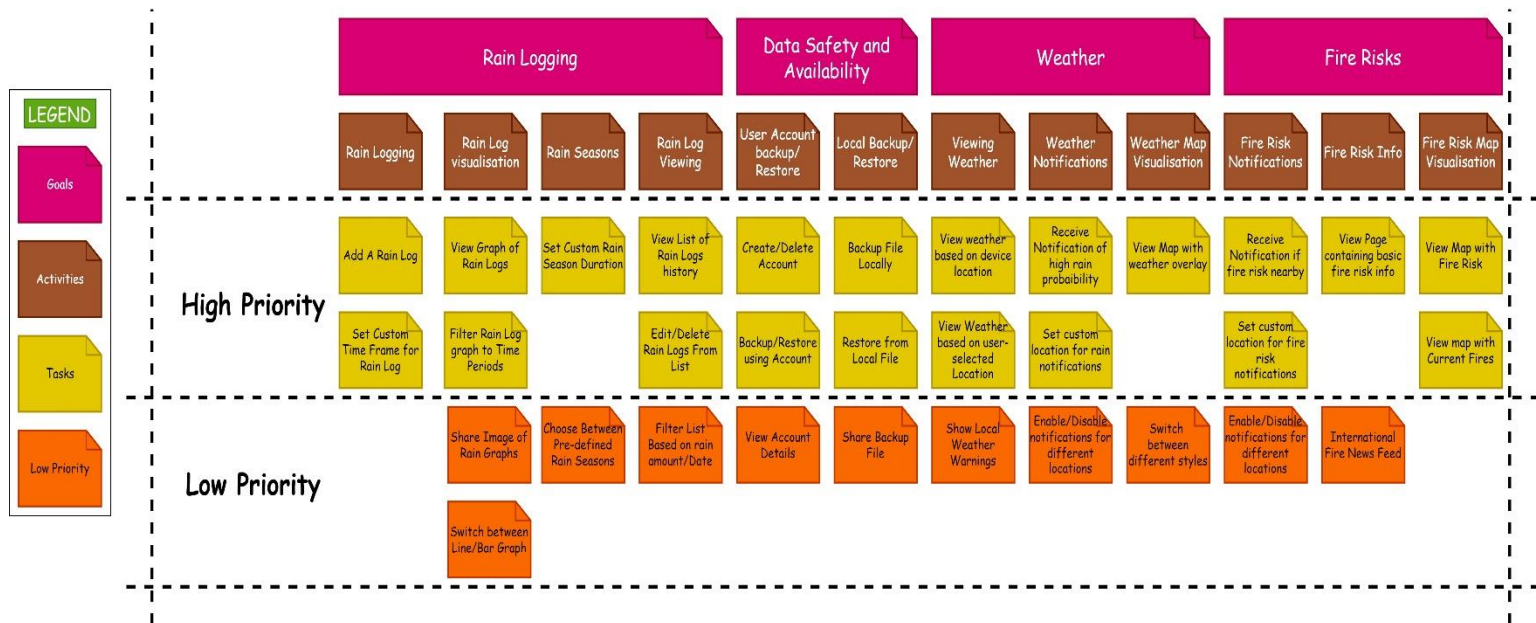
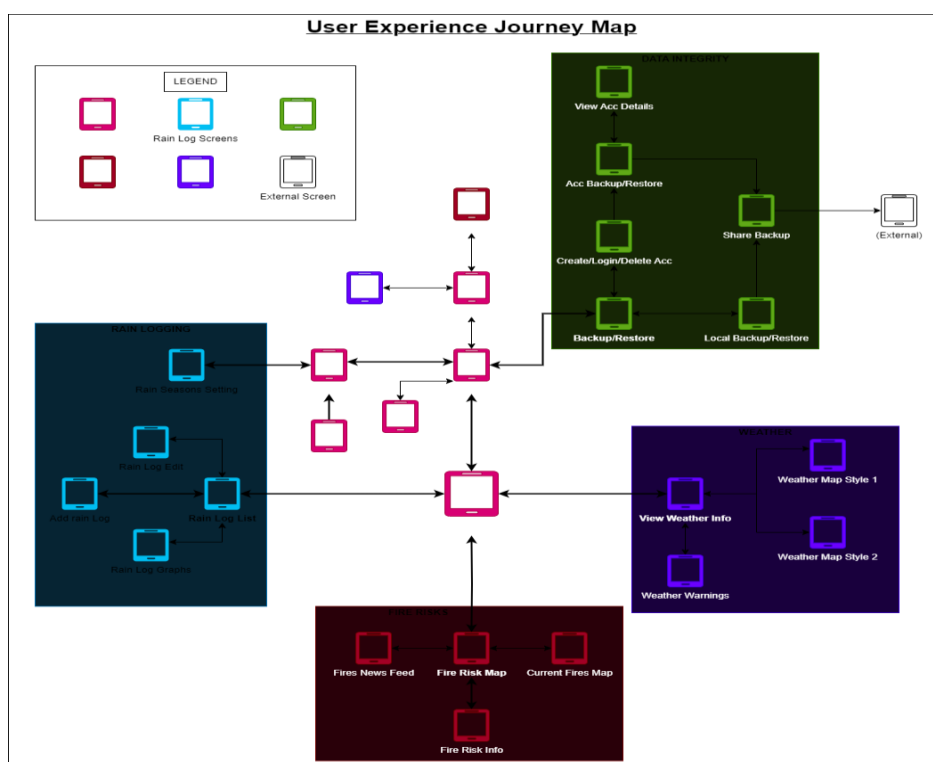
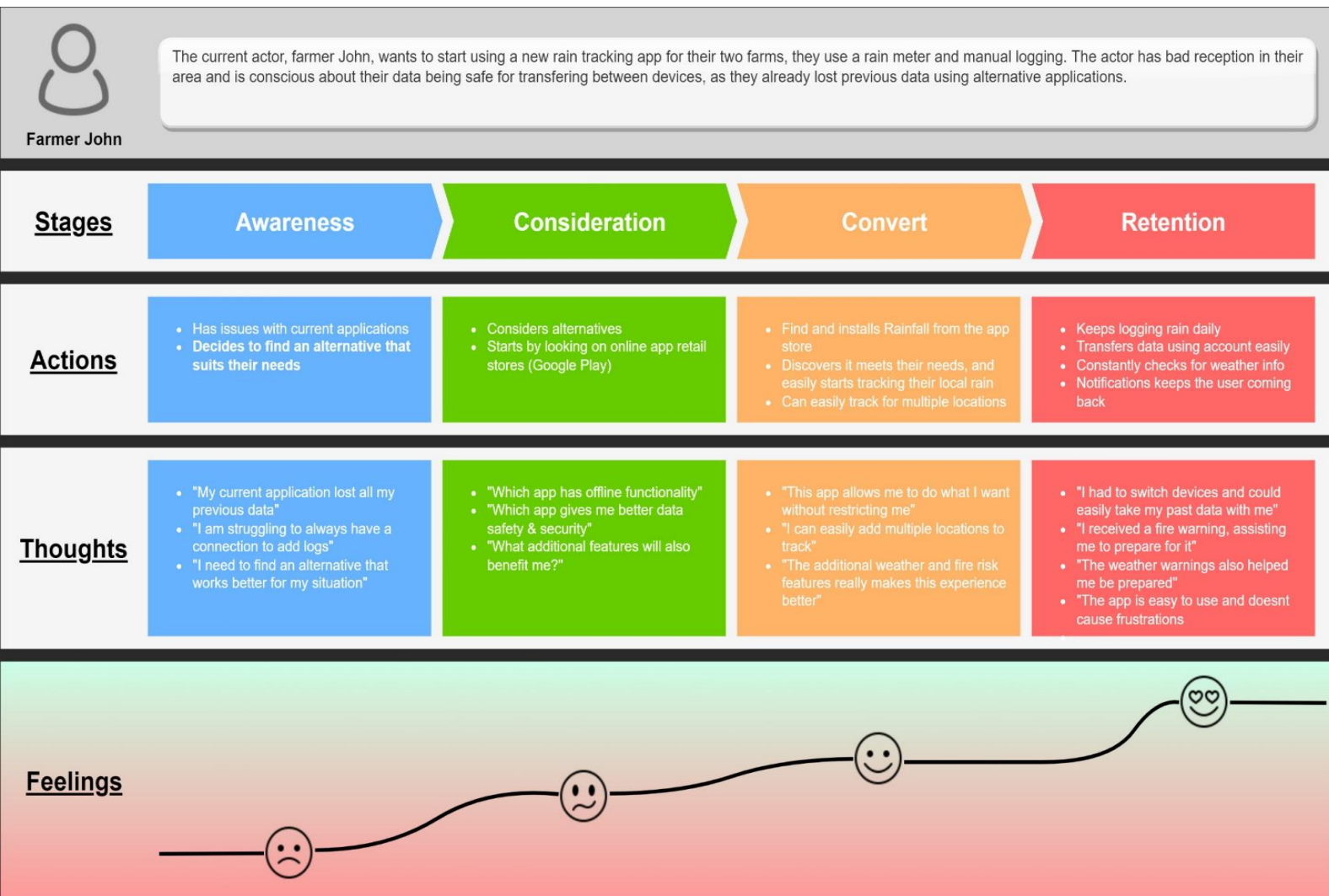


Figure 2 User Story Epic



## 2.2. User Experience Journey Maps



Figure(s) 3 User Experience Journey Maps

### 3. NON-FUNCTIONAL REQUIREMENTS

#### **Performance Requirements**

- The app should have full offline functionality for rain logging and different locations.
- All features requiring a network connection should load within 3 seconds on a 4G network, and within 5 seconds on a 3G network.

#### **Scalability Requirements**

- The app's cloud storage should be able to handle a userbase of up to 1,000 users and be easily scalable for more when the need arises.
- The app's backend infrastructure should be designed to handle a potential increase in user base by 50% over the next year.

#### **Reliability Requirements**

- The online cloud storage should have an uptime of at least 99% of the time.
- The app should have complete offline functionality without losing the rain logging features.

#### **Maintainability Requirements**

- The app's codebase should be well-documented and follow best practices to facilitate easy maintenance by future developers.
- Code reviews should be conducted for all major features or changes, involving at least one other team member for quality assurance.

#### **Security Requirements**

- User account passwords should be encrypted both in transit and in rest to ensure privacy and data safety.
- The app should store minimal personal data of the user and only ask for what is required.

#### **Usability Requirements**

- The app's user interface should be easy to understand and intuitive to ensure good accessibility for a wide range of users having different levels of experience with technology.

#### **Interoperability Requirements**

- The app should be compatible with most devices released after 2020 to ensure a wider range of availability.
- The app should target android API level 30 and above to include new security patches and to qualify for Google Play Publishing.

- The app should support integration with popular third-party weather APIs to enhance its weather data accuracy and coverage.

**Localisation Requirements**

- The app should support multiple languages, primarily Afrikaans and English, as well as a third language commonly used in South Africa.
- The app should have an "About" section in the settings, providing information on its origin, purpose, and contact details.
- The user should have the ability to share rain data in a visual format, allowing for image export.

## 4. ANALYSIS ARTIFACTS

### 4.1. DOMAIN MODELLING

#### **BOUNDED CONTEXT 1: WEATHER TRACKING**

**Description:** This bounded Context is responsible for all aspects related to weather tracking. It Includes modules for collecting weather data, providing weather forecasts, and generating weather-related notifications. This context focuses on delivering accurate and timely weather information to users based on their location preferences.

#### **BOUNDED CONTEXT 2: RAIN LOGGING**

**Description:** The Rain Logging bounded context manages the functionality related to logging and tracking rainfall data. It includes modules for recording rain data, displaying historical rain logs through graphs, and allowing users to set custom timeframes for data logging. Users can also create custom rain season durations and perform basic CRUD operations on their rain logs.

#### **BOUNDED CONTEXT 3: DATA MANAGEMENT AND SECURITY**

**Description:** This current bounded context is responsible for user account management and data security. It includes modules for user account creation and deletion, as well as backup and restoration of user data. Data safety and availability are the primary concerns within this context.

#### **BOUNDED CONTEXT 4: FIRE RISK ANTICIPATION**

**Description:** The Fire Risk Anticipation context focuses on managing and notifying users about fire risks. It includes modules for monitoring fire risk data, sending notifications based on nearby fire risks, and providing information about fire risk factors. Users can customize location parameters for fire risk notifications and access detailed information about fire incidents on a map.

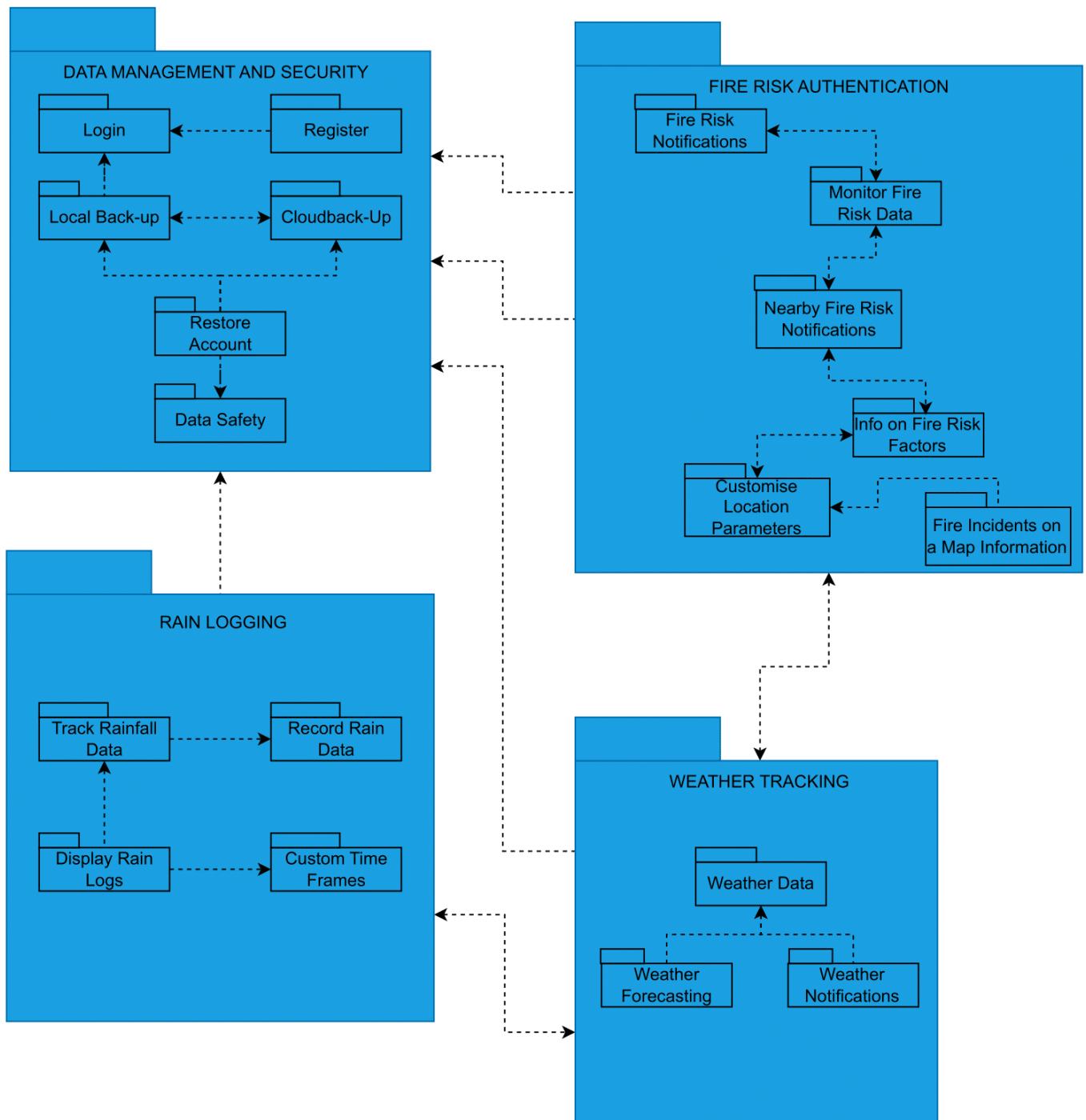


Figure 4 UML Package Diagram

## 5. SOFTWARE IMPLEMENTATION

5.1. **UML Object Diagram:** This Object diagram shows the class relationships for entities that have been identified from analysis.

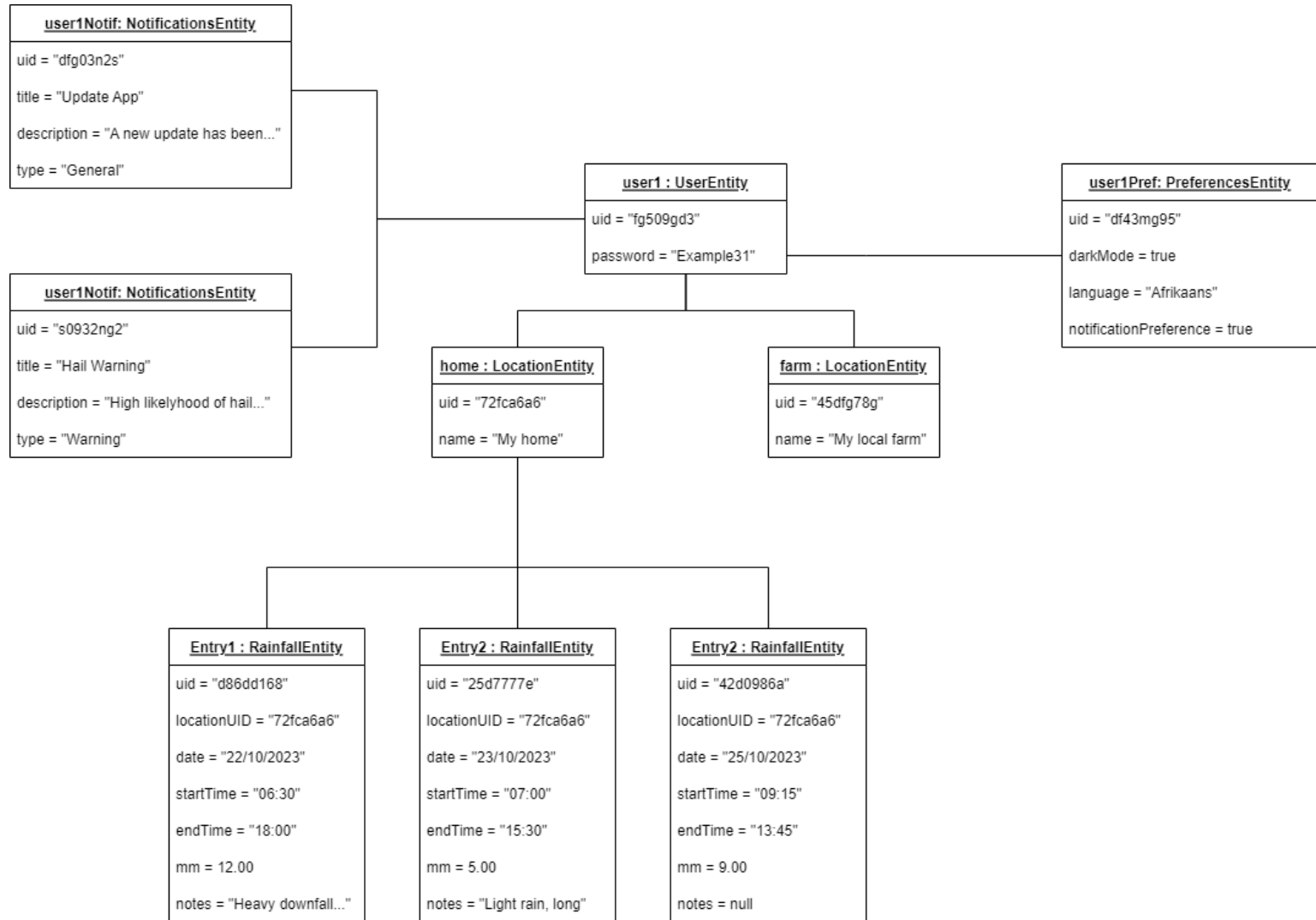


Figure 5 UML Object Diagram

5.2. **UML Sequence Diagram:** The sequence diagrams show interactions between components for important complex flows.

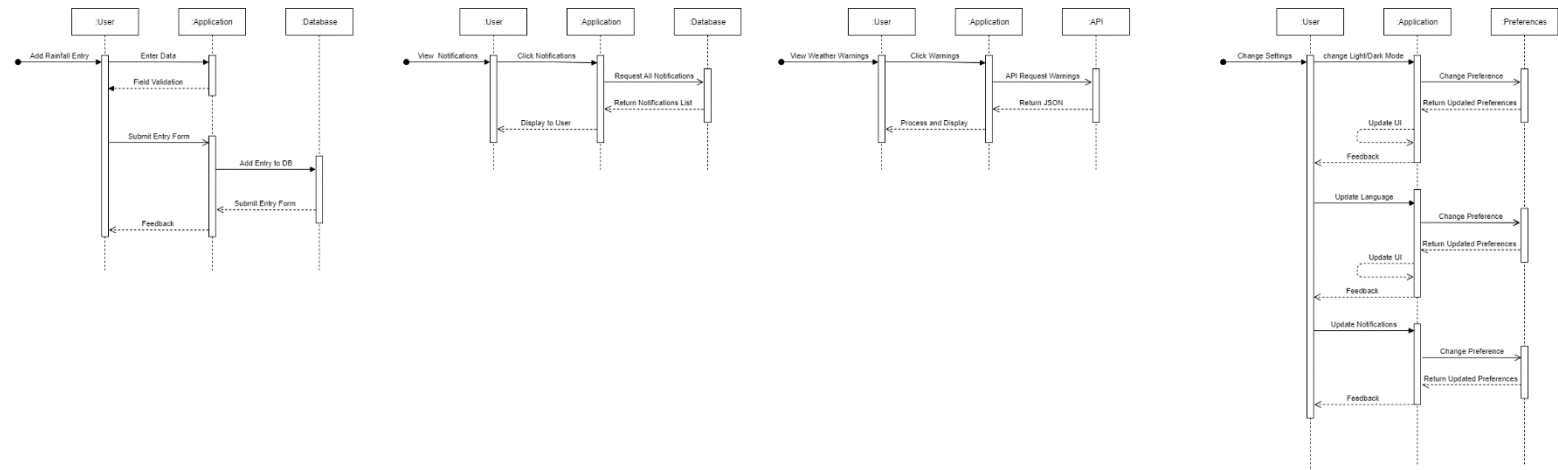


Figure 6 UML Sequence Diagram

5.3. **UML State Diagram:** This diagram will show important state transitions of the Rainfall Mobile Application project.

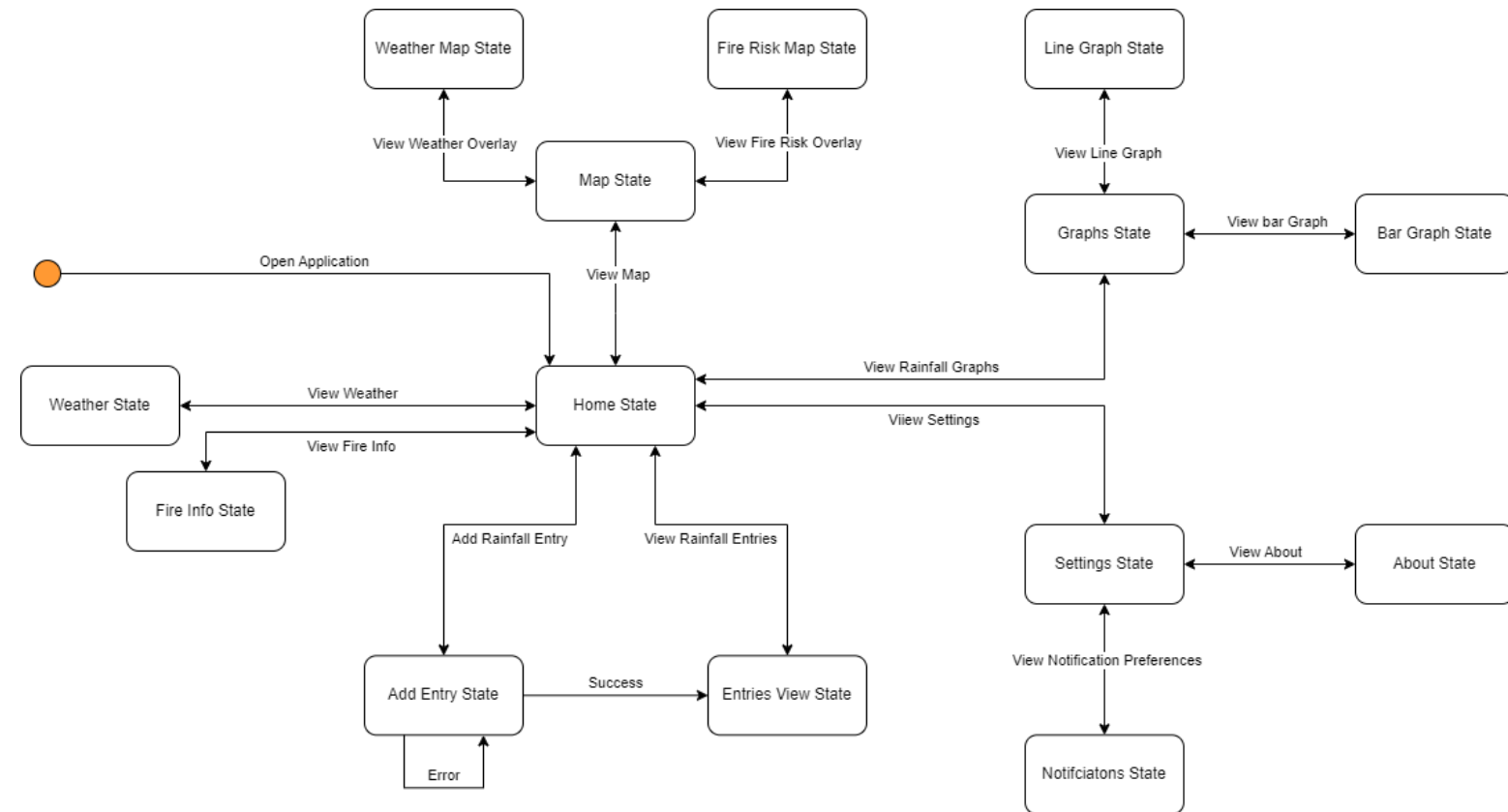


Figure 7 UML State Diagram

## 6. DATA SCHEMAS

### Introduction

Because our application must operate seamlessly without consistent internet access, we have implemented a robust offline-first application following the best practices as described in the Android Developer (2023) guide. This is achieved through a local storage system utilizing SQLite. Additionally, we offer an optional online cloud backup feature using Google Firebase services. The data storage schema primarily revolves around the local SQLite database, which is then adapted for cloud storage in JSON format.

### User Table

This table will store information about users.

#### Columns in the User table:

- user\_id (Primary Key)
- email
- password\_hash (for securely storing passwords)

### Notifications Table

This table will store information about notifications associated with users.

It will also have a foreign key to establish a relationship with the User table. This foreign key will link a notification to a specific user.

#### Columns in the Notifications Table:

- Unique Id (Primary Key)
- User Id (Foreign Key)
- Title
- Description
- Type

### Preferences Table

This table will store user-specific settings. A user will only be added to this table if they change their preferences from the default.

It will have a foreign key to establish a relationship with the User table. This foreign key will link a setting record to a specific user.

#### Columns in the Preferences Table:

- Unique Id (Primary Key)
- User Id (Foreign Key)
- DarkModeEnabled
- Language
- AllowNotifications



## Location Table

This table will store the location that a user has created. The user will be able to create multiple location to store rain fall and thus we will need multiple rows in the table.

It will have a foreign key to establish a relationship with the User table. This foreign key will link a setting record to a specific user.

### Columns in the Locations Table

- Uid (Primary Key)
- UserId (Foreign Key)
- Name

## Rainfall Table

This table will store the rainfall entries that the user will create. Multiple rainfall entries can exist for each individual location, creating a one-to-many relationship.

It will have a foreign key to establish a relationship with the Locations table, using the Uid from the location.

### Columns in the Rainfall Table

- Uid (Primary Key)
- LocationId (Foreign Key)
- DateCaptured
- StartTime
- EndTime
- Mm
- Notes

## Google Firebase

Google firebase uses a JSON document storage type (Gathoni, 2022). The JSON document structure stored on Firebase will comprise a primary user object. Within this user object, we will store the email and password. Moving forward, we will include a 'Preferences' object, encapsulating all user-defined settings for restoration purposes. Additionally, both location data and rainfall entries will be stored as arrays under the user object. Rainfall entries will be nested within the Location objects. The data structures and types within the JSON will closely mirror those of the local SQL database schema.

## ERD Diagram

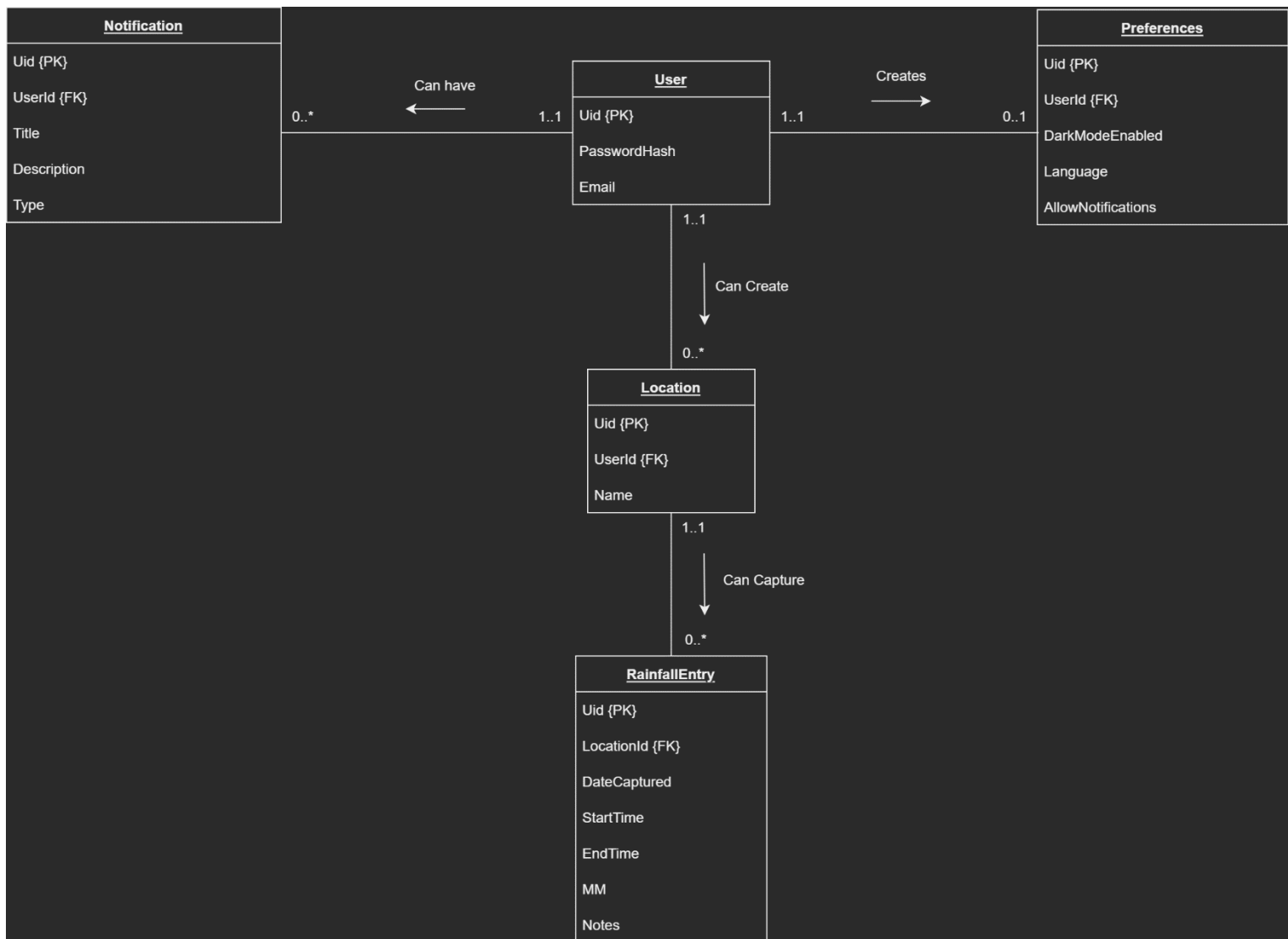


Figure 8 ERD Diagram

## JSON Schema

(Find inside .zip the .JSON document for better viewing)

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "User": {
      "type": "object",
      "properties": {
        "Uid": {
          "type": "string"
        },
        "Email": {
          "type": "string"
        },
        "PasswordHash": {
          "type": "string"
        },
        "Preferences": {
          "type": "object",
          "properties": {
            "Uid": {
              "type": "string"
            },
            "Theme": {
              "type": "string"
            },
            "Language": {
              "type": "string"
            },
            "AllowNotifications": {
              "type": "boolean"
            }
          }
        },
        "required": [
          "Uid",
          "Theme",
          "Language",
          "AllowNotifications"
        ]
      },
    },
    "Location": {
      "type": "array",
      "items": [
        {
          "type": "object",
          "properties": {
            "Uid": {
              "type": "string"
            },
            "Name": {
              "type": "string"
            },
            "Rainfall": {
              "type": "array",
              "items": [
                {
                  "type": "object",
                  "properties": {
                    "DateCaptured": {
                      "type": "string"
                    },
                    "StartTime": {
                      "type": "string"
                    },
                    "EndTime": {
                      "type": "string"
                    },
                    "MM": {
                      "type": "number"
                    },
                    "Notes": {
```

```

        "type": "null"
      }
    },
    "required": [
      "DateCaptured",
      "StartTime",
      "EndTime",
      "MM",
      "Notes"
    ]
  }
]
},
"required": [
  "Uid",
  "Name",
  "Rainfall"
]
},
{
  "type": "object",
  "properties": {
    "Uid": {
      "type": "string"
    },
    "Name": {
      "type": "string"
    },
    "Rainfall": {
      "type": "array",
      "items": {}
    }
  },
  "required": [
    "Uid",
    "Name",
    "Rainfall"
  ]
}
},
"required": [
  "Uid",
  "Email",
  "PasswordHash",
  "Preferences",
  "Location"
]
},
"required": [
  "User"
]
}

```

## JSON Usage Example

```
{
  "User": {
    "Uid": "03892174",
    "Email": "John@1.com",
    "PasswordHash": "211addddsa09321hkldsbnm",
    "Preferences": {
      "Uid": "3214321545314",
      "Theme": "Dark",
      "Language": "English",
      "AllowNotifications": true
    },
    "Location": [
      {
        "Uid": "3214321545314",
        "Name": "Farm",
        "Rainfall": [
          {
            "DateCaptured": "2019-01-01",
            "StartTime": "12:00",
            "EndTime": "13:00",
            "MM": 12.5,
            "Notes": null
          }
        ]
      }
    ],
    {
      "Uid": "432178956",
      "Name": "Home",
      "Rainfall": []
    }
  ]
}
```

## 7. ARCHITECTURE ARTIFACTS

### **Design Patterns: MVVM Design Pattern:**

MVVM is a design pattern that facilitates a clear separation of concerns among different parts of the application. It consists of three main components:

- **Model:** Represents the data and business logic of the application.
- **View:** Represents the UI elements and their layout.
- **ViewModel:** Acts as an intermediary between the Model and the View, exposing the data and business logic in a way that the View can consume without directly interacting with the Model.

### **Reasons for Choosing MVVM:**

- **Strong Concern Separation:** MVVM enables the separation of UI logic (View) from business logic (ViewModel) and data (Model). This separation results in cleaner, more maintainable, and testable code.
- **Android Lifecycle Awareness:** Android's lifecycle-aware components such as LiveData and ViewModel complement MVVM. They help manage UI updates efficiently, especially in scenarios like handling orientation changes or device configurations.
- **Testability and Maintainability:** MVVM allows for easier unit testing as the business logic resides in the ViewModel, which can be tested independently of the UI components. This enhances maintainability and code reusability.
- **Improved UI Responsiveness:** By binding the View to the ViewModel using LiveData or other observable data structures, updates to the underlying data automatically reflect in the UI, improving responsiveness and user experience.

### **Architecture Pattern: Clean Architecture**

Clean Architecture is an architectural pattern that promotes the separation of concerns by dividing the application into layers with distinct responsibilities:

- **Presentation Layer:** Contains the UI components responsible for displaying data and handling user interactions. In Android, this often includes activities, fragments, views, and view models.
- **Domain Layer:** Holds the business logic and rules of the application. It is independent of the UI and external frameworks/libraries, allowing for reusability across different platforms.
- **Data Layer:** Manages the data access and storage mechanisms, including databases, network calls, and repositories. It abstracts data sources from the rest of the application.

## **Reasons for Choosing Clean Architecture:**

- **Separation of Concerns:** Clean Architecture emphasizes the separation of concerns by defining clear boundaries between layers. This separation enables easier maintenance, scalability, and testability.
- **Independence of Frameworks/Libraries:** Business logic within the domain layer remains decoupled from UI and external dependencies. This independence makes the codebase more modular, adaptable, and less susceptible to changes in external technologies.
- **Testability and Flexibility:** Each layer's independence facilitates isolated testing. Business rules can be tested independently of UI or external systems, allowing for easier modifications and enhancements.
- **Scalability and Maintainability:** The modular nature of Clean Architecture enables easy addition or modification of features without affecting the entire system. It facilitates long-term maintainability and extensibility of the application.

The selection of MVVM and Clean Architecture in the rain tracking application was driven by the need for robust separation of concerns, improved testability, maintainability, scalability, and adaptability. These patterns create a structured and organized codebase, ensuring a reliable and robust application architecture.

## **Technology and Cloud Architecture:**

### **Cloud-Based Services:**

- **Google Cloud Firestore:** Chosen due to its seamless integration with Android, scalability, and real-time data synchronization capabilities. Firestore offers a NoSQL document database that allows efficient storage and retrieval of data, providing real-time updates across devices and ensuring data consistency.
- **OpenWeatherMap and WeatherAPI:** Integrated to fetch meteorological data required for weather tracking within the application. These external APIs provide accurate and reliable weather-related information necessary for displaying weather forecasts and warnings.
- **NASA FIRMS API:** Utilized to access fire risk data, which is crucial for the application's functionality in assessing and displaying fire risks in specific geographical areas. Integration with this API allows users to access critical information related to fire hazards.
- **Google Maps API:** Employed to display maps within the application, enabling users to visualize weather patterns, fire risk areas, and other location-based information. The API offers robust mapping functionalities that enhance user experience and interaction with the app.

## **Reasoning for Cloud Architecture Technologies:**

**Firestore Integration:** Chosen for its compatibility with Android development and its ability to handle real-time data synchronization. This ensures that weather-related data, user preferences, and other app data are efficiently stored and updated in real-time across devices.

**Weather and Fire Risk APIs:** Integrated to leverage external sources for weather and fire risk data. These APIs provide reliable and updated information necessary for accurate weather forecasts, warnings, and fire risk assessments within the application.

**Google Maps API:** Employed to enhance the user experience by visualizing weather and fire-related data on interactive maps. This feature allows users to view weather patterns, fire risk zones, and other location-based information, enhancing the app's usability.

**Scalability and Reliability:** The chosen cloud-based services are known for their scalability and reliability, ensuring that the rain tracking application can handle varying loads of data and users while maintaining consistent performance.

**Compatibility with Application Requirements:** The cloud services were selected based on their compatibility with the rain tracking app's specific needs, such as real-time data updates, accurate weather information, and reliable fire risk assessments.

The cloud architecture decisions were made strategically to incorporate robust cloud-based services that offer scalability, real-time data synchronization, accurate weather information, fire risk data, and seamless integration with the rain tracking application. These choices were aimed at enhancing the app's functionality, user experience, and overall performance while ensuring data accuracy and reliability.



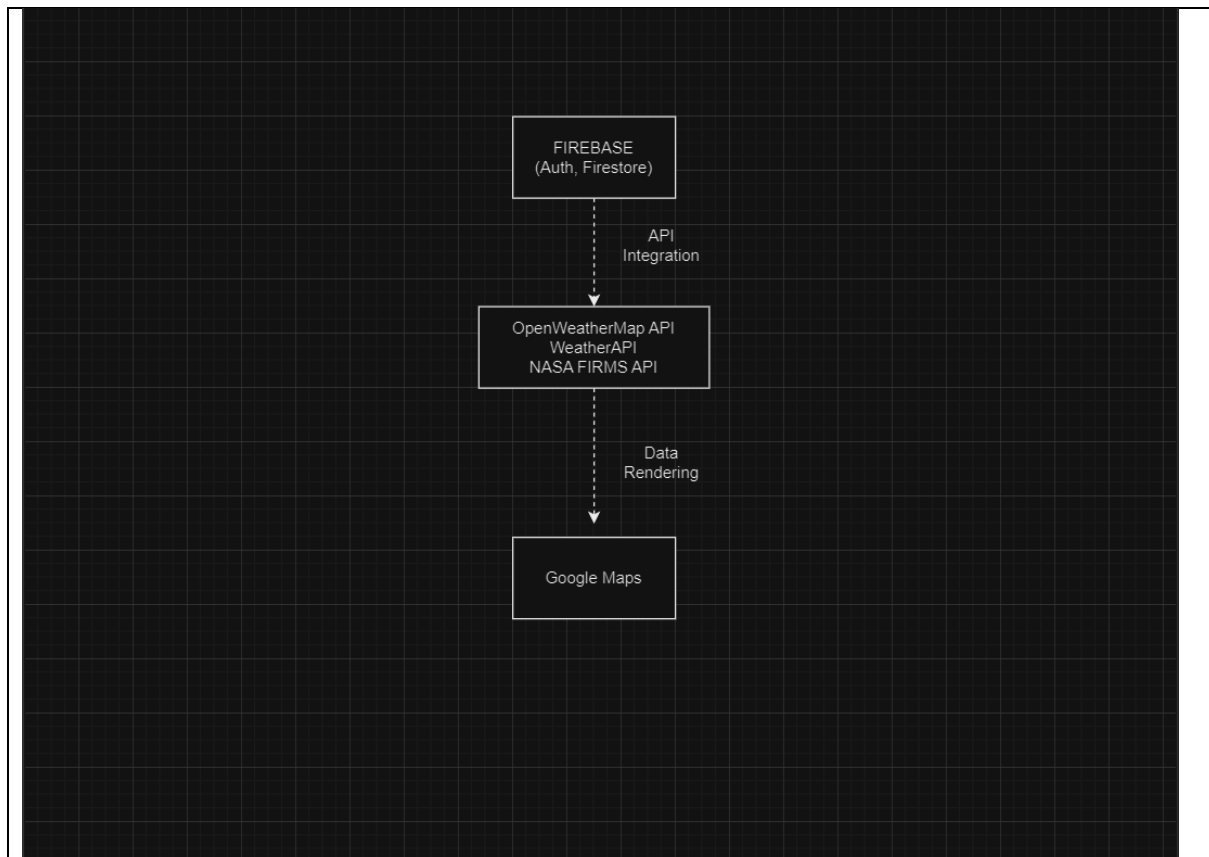


Figure 9 Architecture Artifact Diagram

## 8. SECURITY

### Security Considerations:

#### Networking Security:

- **HTTPS Encryption:** All communication between the rain tracking application and external services (APIs, cloud services) is performed using HTTPS. This encryption protocol ensures secure and encrypted data transmission, preventing data interception, tampering, and eavesdropping.
- **Firebase Authentication:** Utilization of Firebase Authentication ensures secure user authentication and management. It offers various authentication methods, including email/password, OAuth, and phone number authentication, enhancing the application's security by verifying user identities.
- **Data Encryption:** Data encryption mechanisms, especially within Firestore and other storage options, are employed to encrypt sensitive data at rest. This encryption adds an extra layer of security, safeguarding user data stored within the application against unauthorized access.

#### Vulnerabilities and Threats Mitigation:

- **Input Validation:** Implementation of robust input validation techniques prevents common security threats such as SQL injection, cross-site scripting (XSS), and other injection attacks. Validating user inputs at various levels ensures that the application handles data securely and prevents malicious data manipulation.
- **Authorization Checks:** Strict authorization checks are enforced to ensure that only authenticated and authorized users have access to specific functionalities and data within the application. Role-based access control (RBAC) or similar mechanisms are employed to manage access permissions effectively.

## **Reasoning for Security Measures:**

**HTTPS Encryption:** Chosen to encrypt data transmission between the application and external services, preventing unauthorized access to sensitive information transmitted over the network. This is crucial for securing user data during data exchanges.

**Firebase Authentication:** Utilized for its robust user authentication capabilities, Firebase Authentication ensures that only legitimate users with verified credentials can access the rain tracking application's features, thereby preventing unauthorized access.

**Data Encryption:** Implementing encryption mechanisms within storage options like Firestore ensures that even if unauthorized access occurs, sensitive data stored within the app is encrypted, thereby protecting user privacy and data confidentiality.

**Input Validation:** Integral to safeguarding against common security threats, robust input validation mitigates risks associated with malicious data inputs, preventing potential vulnerabilities like SQL injection and XSS attacks, and maintaining data integrity.

**Authorization Checks:** Enforcing strict authorization protocols ensures that access to various parts of the application is restricted to authorized users only, reducing the risk of data breaches and maintaining the confidentiality and integrity of sensitive information.

The security measures implemented in the rain tracking application were chosen to establish a robust security framework, protecting user data, preventing common vulnerabilities, and ensuring secure communication and access control throughout the application's lifecycle. These measures collectively fortify the application against potential threats and vulnerabilities, enhancing user trust and confidence in its security.

## 9. DEVOPS

Our DevOps pipeline is encapsulated in a GitHub Actions workflow. The purpose of this which focuses on continuous integration for Android app development. It is designed to automate the building and testing of an Android application whenever changes are pushed to the main branch of the repository. Here's a breakdown of the workflow steps:

Trigger:

- This workflow is triggered on a push event to the repository, but it only runs when the push occurs on the main branch.

Jobs:

- This workflow defines a single job called "Build and Test" that will run on an Ubuntu-based runner (GitHub's infrastructure for running workflows).

Steps:

1. Checkout Code: This step checks out the source code from the repository so that the subsequent steps can work with it.
2. Setup JDK: This step sets up the Java Development Kit (JDK) on the runner. It uses the actions/setup-java action to install the Zulu distribution of JDK 17.
3. Setup Gradle Cache: This step sets up caching for Gradle dependencies and the Gradle Wrapper. It caches the contents of the ~/.gradle/caches and ~/.gradle/wrapper directories. Caching can significantly improve build times by reusing previously downloaded dependencies.
4. Make gradlew executable: This step if needed on the Linux runner in order to be able to execute the gradlew script (Gradle Wrapper). The gradlew script is used to run Gradle tasks for building and testing the Android application.
5. Run Linters and Test: This step runs Gradle with the check task, this runs various static code analysis tools which includes code quality checks and running unit tests.
6. Upload Reports: This step uploads the test reports generated by the Gradle build process. The reports are located in the app/build/ report's directory. This step uses the actions/upload-artifact action to make the reports available for later inspection. The if: always () condition ensures that this step runs even if the previous steps fail. This is useful for capturing test reports and other artifacts for diagnostic purposes, regardless of whether the build succeeded or failed.
7. TODO: The app is uploaded to xxx to run A security testing third-party tool

The workflow is designed to automate the process of building and testing an Android app on the main branch, providing consistency and reliability in the development process. The workflow can be further customized and extended to include additional steps or actions as needed for the specific Android project.

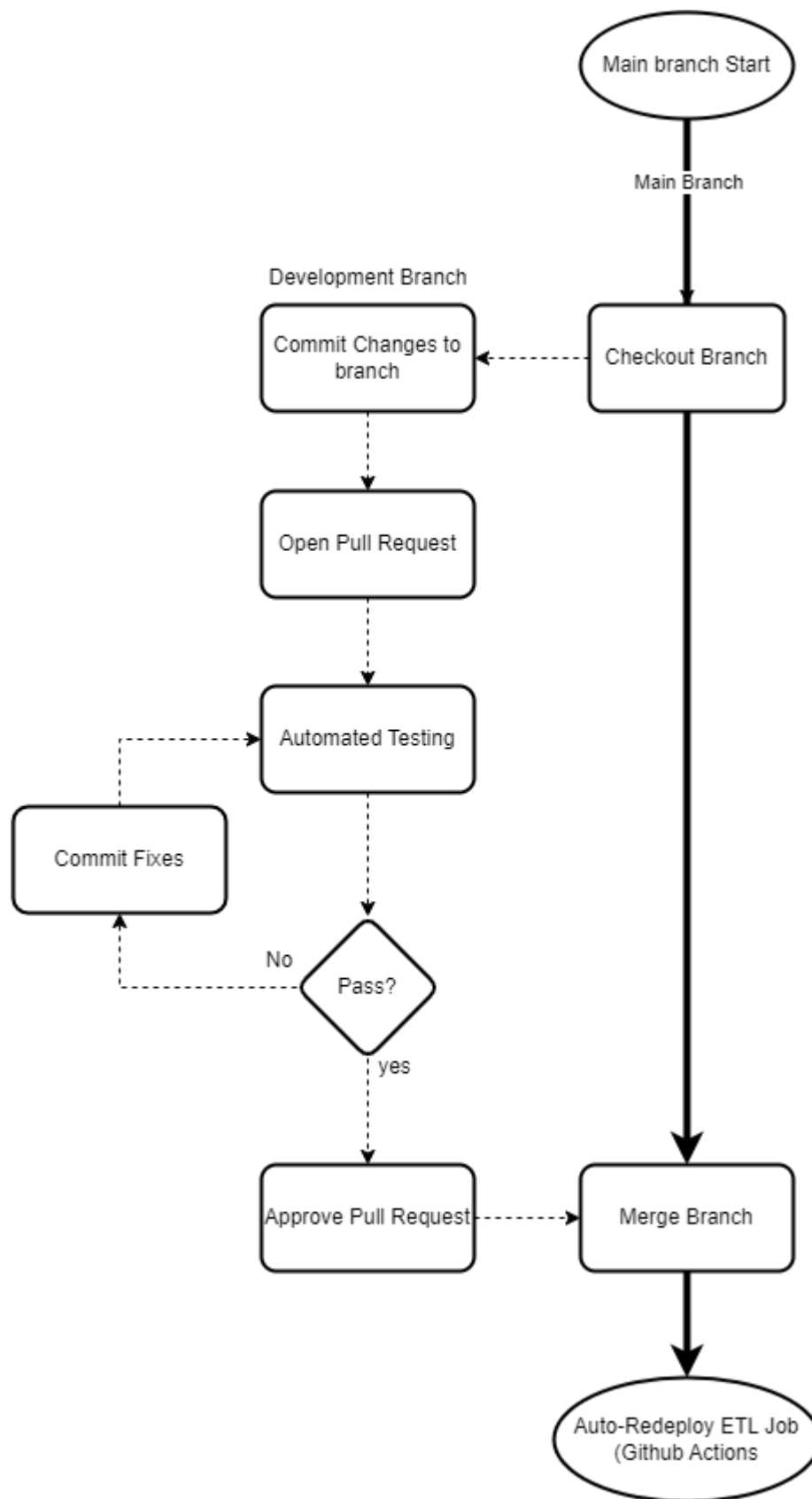


Figure 10 GitHub Pipeline Diagram

## 10. RUNNING COSTS

### 1. Infrastructure Costs:

Server Hosting (Cloud): The cost of hosting servers on a Google Cloud for backend services, storage, and databases. Estimated cost: R1900 – R9400 per month.

Database Costs: Cost associated with cloud-based database services. Estimated cost: R940 – R3760 per month.

Backup Services: Cost of backup services for data security. Estimated cost: R376 - R1880 per month.

### 2. Development and Maintenance Costs:

Development Team: Expenses for developers, designers, and maintenance staff. Estimated cost: R94 124 – R376,497 per month.

Software Licenses: Cost for any third-party software or tools. Estimated cost: R941 – R1880 per month.

Bug Fixing and Updates: Ongoing costs for maintaining and updating the application. Estimated cost: R18,820 – R94,102 per month.

### 3. Communication Costs:

Push Notifications: Cost associated with sending push notifications. Estimated cost: R188,24 – R941 per month.

### 4. Localization and Multilingual Support:

Translation Services: Cost for translating app content. Estimated cost: R1880 - R9400 per month.

Multilingual Support: Ongoing costs for maintaining language support. Estimated cost: R940 – R3770 per month.

### 5. Security and Compliance:

Security Measures: Costs for implementing and maintaining security features. Estimated cost: R1880 - R9400per month.

Compliance Costs: Costs associated with ensuring the app complies with regulations. Estimated cost: R940 - R3770 per month.

### 6. Marketing and User Acquisition:

Advertising: Cost for marketing campaigns. Estimated cost: R3770 – R37,657 per month.

App Store Fees: Fees associated with listing the app-on-app stores. Estimated cost: Varies.

### 7. Total Estimated Monthly Running Costs: R131,799 – R564,537

## 1. Scaling Points for Each Technology:

### 1.1 Google Cloud Firestore (Database):

Scaling Trigger: When the volume of rain data exceeds 1 GB.

Predicted Scaling Points:

Best Case: Month 8

Mean Case: Month 12

Worst Case: Month 15

## 2. Predictive Models for User Growth Scenarios:

### 2.1 Best Case Scenario (20% Monthly Growth):



Figure 11 Best Case Scenario Graph

## 2.2 Mean Case Scenario (10% Monthly Growth):

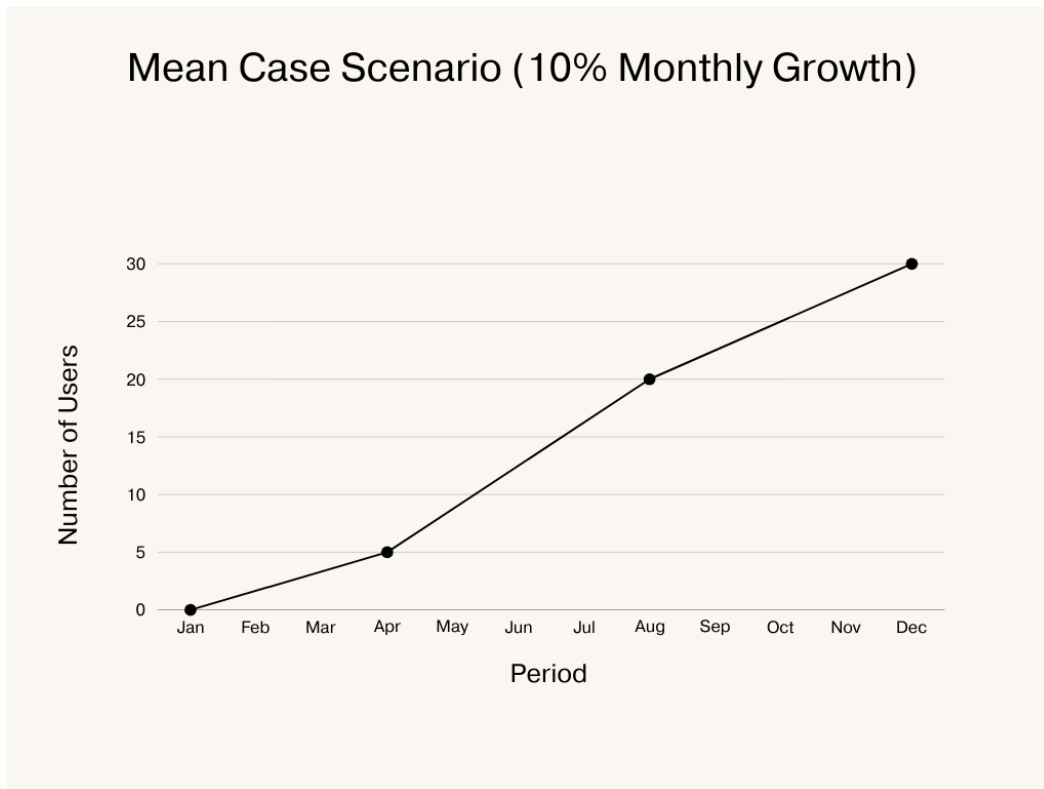


Figure 12 Mean Case Scenario Graph

## 2.3 Worst Case Scenario (5% Monthly Growth):



Figure 13 Worst Case Scenario Graph



### **3. Adopting Alternative Technologies:**

#### **3.1 Scenario:**

If Google Cloud Firestore (Database) faces limitations or increased costs at higher volumes.

#### **3.2 Alternative Technology:**

Consideration: Bigtable or BigQuery for improved scalability.

Cost Analysis: A Conduction of a detailed analysis of the costs associated with migrating to and maintaining Bigtable or BigQuery.

#### **3.3 Decision Point:**

Assessing adoption based on predicted scaling points and potential limitations of the current technology.

## **11. CHANGE MANAGEMENT**

### **How and why will the organization adopt your software?**

The software solution that has been suggested is a comprehensive tool that is intended to cater to the particular requirements of workers who are responsible for monitoring rainfall data at different sites. The application offers a number of features in addition to its primary function of tracking rainfall, such as a map of fires, weather alerts, a map of cloud cover, and support for multiple locations. This comprehensive strategy offers a one-stop solution for various related tasks in addition to streamlining the rain data tracking procedure.

The software's inherent worth to the company is derived from its capacity to fulfil the exact specifications established by the establishment. The suggested application, in contrast to current software options, provides a special feature set customized to the needs of the business. This all-inclusive suite is an affordable solution that also improves operational efficiency. The company stands to save time and resources that would otherwise be used to manage a multitude of separate software tools by combining multiple functionalities into a single application.

Moreover, the software is in complete coherence with the broad objectives specified in the project scope. The application functions as a physical representation of the company's strategic vision, combining a variety of features into a user-friendly interface that is simple to use. This alignment guarantees that the software is scalable and flexible enough to accommodate changing needs in addition to satisfying current ones. In addition to streamlining present procedures, the platform should be adaptable enough to change with the needs of the business and welcome new users with ease thanks to its user-friendly interface.

### **How and why will the users adopt your software?**

The software was designed with the users in mind, as it was set out as an important requirement for the application. The company wants the app to be easily accessible by a multitude of different users in different locations, and not to be only used internally. The application will start off with a user-testing period where the company will have a select group of participants use the app and provide feedback, from this stage the app will get constantly improved until it is at a point the company feels it can be fully adopted by different users. The goal of the app's design is to be easily used by users and easy to understand and use.

The initial onboarding for the application will consist of a one-on-one meeting with the business owner, explaining what the app's functionality is and how to use it. From there, the owner can distribute the app. The goal is for the app to require no onboarding for new users, as they should ideally be able to pick up the app and immediately understand what to do. For the smaller complicated features, or any questions, the app will have a FAQ section that the user can read through to better understand how certain features function or why something could potentially work different as expected.

The benefits of adopting this application is mainly to centralise a wide variety of features to fit a wide scope of user needs. The app will be intuitive and easy to adopt.

## **What is your strategy to gain adoption from both the organization and the users?**

We plan to make sure that our rain data tracking app is widely used by aligning the organization and users with the application's distinct value proposition. Our software provides the organization with a single, all-inclusive solution by combining crucial features like weather warnings, fire maps, rain data tracking, and cloud coverage. Because of this consolidation, the organization will save a great deal of time and money by not needing as many different software tools. Furthermore, the software offers an easily adjustable solution for both present and future user needs, integrating seamlessly with the company's project scope and goals. We hope to gain organizational support for the all-in-one application's quick adoption by highlighting the cost- and efficiency-saving benefits.

Regarding users, our approach centers on designing with them in mind and implementing a gradual onboarding process. With an emphasis on usability and accessibility, the application is designed to be user-friendly. A small number of participants in the initial user testing phase will offer insightful feedback that will enable the application to be continuously improved. The business owner will conduct one-on-one onboarding sessions to guarantee that users comprehend the features and advantages of the app, paving the way for a smooth integration into their workflows. The intention is for the app to be so intuitive that it can be easily used by new users without the need for formal onboarding. We plan to incorporate a thorough FAQ section to provide additional support to users. This section will address common questions and complexities, resulting in a seamless and independent user experience. Our adoption strategy focuses on centralizing various features, making the software easy to use, and enhancing it continuously based on user feedback in order to make it an essential tool for the organization and its users.

## **Maintenance and support**

Our maintenance and support plan emphasizes proactive measures and direct user engagement, in addition to the Service Level Agreement (SLA) that details response times, issue resolution procedures, and updates. We intend to incorporate a user feedback portal into the application to augment the user experience and guarantee the continuous enhancement of our software. Users will be able to report minor bugs, offer suggestions, and share their experiences with our development team all through this one-stop shop. We will be able to quickly address any problems, put in place any necessary fixes, and make ongoing improvements based on user insights thanks to this real-time feedback loop.

Understanding how critical it is to control expectations, we will carefully customize our SLA to the size of the team to guarantee that response times and issue resolution are in line with practical capabilities. The establishment of unambiguous channels of communication will enable the company to stay updated on the status of reported issues or requested features. Our development team and the client will become more transparent and trustworthy as a result of regular updates on the state of ongoing maintenance and enhancements.

In order to provide even more support, our plan includes at least a year of full support, during which the business owner will have easy access to the development team. This open channel of communication guarantees that any questions, concerns, requests for updates, or modifications can be handled right away. Our maintenance and support strategy prioritizes user feedback and builds a

collaborative relationship to not only fix problems quickly but also to proactively improve the software's usability and performance over time.

## 12. APPENDICES

### 1.Declaration(s) of Authenticity

IIE Module Manual

XBCAD7319/XBCAD7329


Declaration of authenticity

I, Nomaswazi Sibeko ID Number, 0012130084085

hereby declare that this portfolio, and any evidence included therein, contains my own independent work and that I have not received help from other groups.

I confirm that we have not committed plagiarism in the accomplishment of this work, nor have I falsified and/or invented experimental data.

I accept the academic penalties that may be imposed for violations of the above.



STUDENT SIGNATURE

DATE 26/11/2023

IIE Module Manual

XBCAD7319/XBCAD7329

Declaration of authenticity

I, Ryan Blignaut ID Number, 0208285099080

hereby declare that this portfolio, and any evidence included therein, contains my own independent work and that I have not received help from other groups.

I confirm that we have not committed plagiarism in the accomplishment of this work, nor have I falsified and/or invented experimental data.

I accept the academic penalties that may be imposed for violations of the above.

\_\_\_\_\_

STUDENT SIGNATURE 

DATE 2023/11/27

**Declaration of authenticity**

I, Armando Bufacchi ID Number, 0209275082086

hereby declare that this portfolio, and any evidence included therein, contains my own independent work and that I have not received help from other groups.

I confirm that we have not committed plagiarism in the accomplishment of this work, nor have I falsified and/or invented experimental data.

I accept the academic penalties that may be imposed for violations of the above.



STUDENT SIGNATURE

27 September 2023

DATE

**Declaration of authenticity**

I, Corné Ackerman ID Number, 0211045123083

hereby declare that this portfolio, and any evidence included therein, contains my own independent work and that I have not received help from other groups.

I confirm that we have not committed plagiarism in the accomplishment of this work, nor have I falsified and/or invented experimental data.

I accept the academic penalties that may be imposed for violations of the above.

Corné Ackerman  
0211045123083

STUDENT SIGNATURE



DATE 27/11/2023

## 2. Scrum Artifacts

### 2.1. SCRUM BOARD to track team performance.

TO DO16

Rain Logging:

RAIN-5

1. Implement Rain Data Collection Module

RAIN-6

2. Design Rain Logging UI

RAIN-7

3. Integrate Rain Logging Backend

+ Create

IN PROGRESS8

Rain Logging:

RAIN-9

1. Implement Rain Data Collection Module

RAIN-10

Data Safety and Availability:

RAIN-11

1. Develop Data Encryption Mechanism

RAIN-12

IN REVIEW8

Rain Logging:

RAIN-17

1. Design Rain Logging UI

RAIN-18

Data Safety and Availability:

RAIN-19

1. Establish Data Backup Procedures

RAIN-20

DONE8

Rain Logging:

RAIN-25

1. Integrate Rain Logging Backend

RAIN-26

Data Safety and Availability:

RAIN-27

1. Implement Redundancy Measures

RAIN-28

TO DO16

Data Safety and Availability:

RAIN-33

1. Develop Data Encryption Mechanism

RAIN-34

2. Establish Data Backup Procedures

RAIN-35

3. Implement Redundancy Measures

+ Create

IN PROGRESS8

Weather Tracking:

RAIN-13

Create Weather Forecast Display

RAIN-14

Fire Risks:

RAIN-15

1. Research Fire Risk Algorithms

RAIN-16

IN REVIEW8

Weather Tracking:

RAIN-21

1. Integrate Weather API

RAIN-22

Fire Risks:

RAIN-23

1. Develop Fire Risk Prediction Model

RAIN-24

DONE8

Weather Tracking:

RAIN-29

1. Implement Location-Based Weather Updates

RAIN-30

Fire Risks:

RAIN-31

1. Integrate Fire Risk Assessment

RAIN-32

TO DO16

Weather Tracking:

RAIN-37

1. Integrate Weather API

RAIN-38

2. Create Weather Forecast Display

RAIN-39

3. Implement Location-Based Weather Updates

RAIN-40

+ Create

TO DO16

Fire Risks:

RAIN-41

1. Research Fire Risk Algorithms

RAIN-42

2. Develop Fire Risk Prediction Model

RAIN-43

3. Integrate Fire Risk Assessment

RAIN-44

+ Create

47

## 2.2.SPRINT ATTENDANCE RECORD(S)

### Meeting(s)- 1 and 2:

#### 5 ANNEXURE E: Sprint Attendance Record

##### SPRINT ATTENDANCE RECORD

GROUP NAME: RainFall

SPRINT: 1 and 2: (18/08/2023 And (28/08/2023)

Attendees	Backlog Grooming	Sprint Planning	Sprint Review	Sprint Retrospective	Daily Scrums
Client	✓	n/a	✓	n/a	n/a
WIL Coordinator	n/a	n/a	n/a	n/a	n/a
Lecturer	n/a	n/a	n/a	n/a	n/a
Armando Buffacchi	✓	✓	✓	n/a	✓
Cornelius Ackerman	✓	✓	✓	n/a	✓
Nomaswazi Sibeko	✓	✓	✓	n/a	✓
Ryan Blignaut	✓	✓	✓	n/a	✓

### Meeting(s)- 3 and 4:

#### 5 ANNEXURE E: Sprint Attendance Record

##### SPRINT ATTENDANCE RECORD

GROUP NAME: RainFall

SPRINT: 3 and 4: (29/09/2023 And (5/10/2023)

Attendees	Backlog Grooming	Sprint Planning	Sprint Review	Sprint Retrospective	Daily Scrums
Client	✓	n/a	✓	n/a	n/a
WIL Coordinator	n/a	n/a	n/a	n/a	n/a
Lecturer	n/a	n/a	n/a	n/a	n/a
Armando Buffacchi	✓	✓	✓	✓	✓
Cornelius Ackerman	✓	✓	✓	✓	✓
Nomaswazi Sibeko	✓	✓	✓	✓	✓
Ryan Blignaut	✓	✓	✓	✓	✓

### Meeting(s)- 5 and 6:

#### 5 ANNEXURE E: Sprint Attendance Record

##### SPRINT ATTENDANCE RECORD

GROUP NAME: RainFall

SPRINT: 5 and 6: (20/10/2023 And (16/11/2023)

Attendees	Backlog Grooming	Sprint Planning	Sprint Review	Sprint Retrospective	Daily Scrums
Client	✓	n/a	✓	n/a	n/a
WIL Coordinator	n/a	n/a	n/a	n/a	n/a
Lecturer	n/a	n/a	n/a	n/a	n/a
Armando Buffacchi	✓	✓	✓	✓	✓
Cornelius Ackerman	✓	✓	✓	✓	✓
Nomaswazi Sibeko	✓	✓	✓	✓	✓
Ryan Blignaut	✓	✓	✓	✓	✓



## Meeting(s)- 7 and 8:

### 5 ANNEXURE E: Sprint Attendance Record

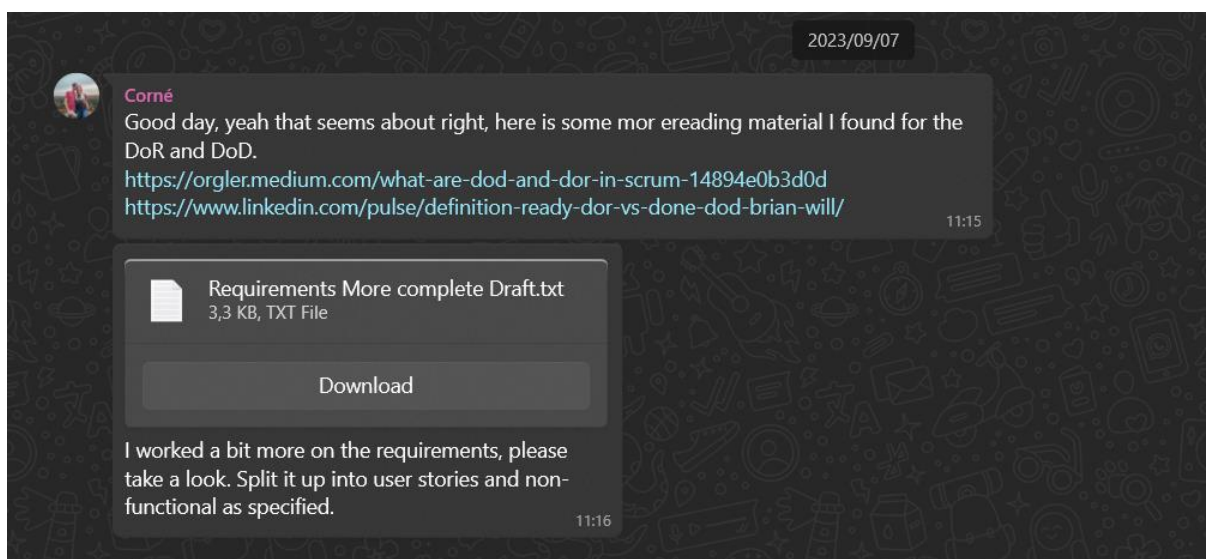
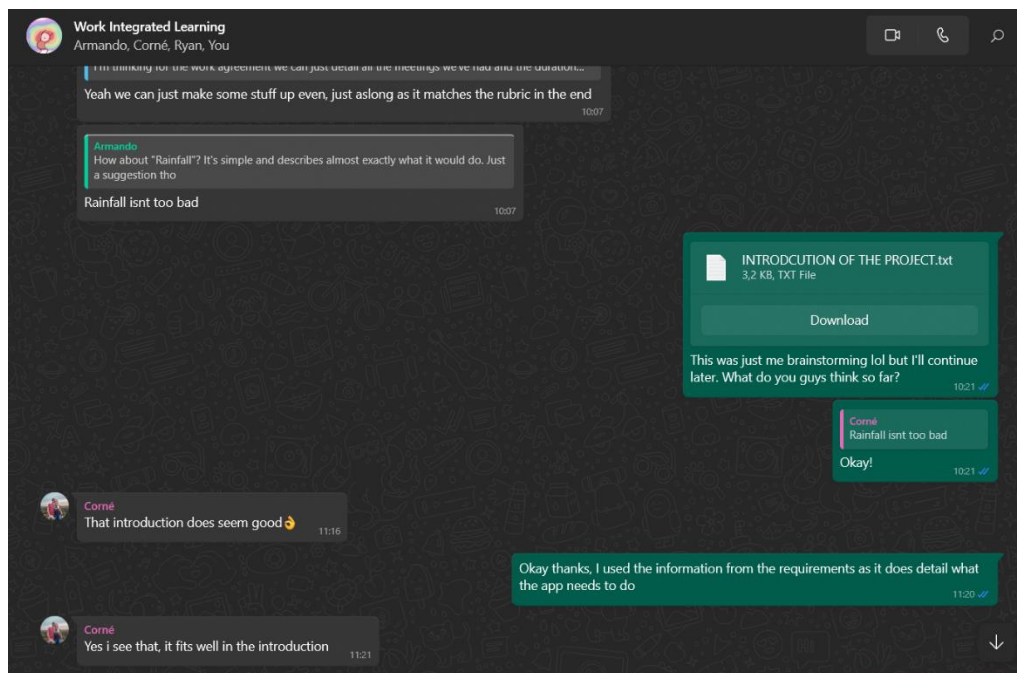
#### SPRINT ATTENDANCE RECORD

GROUP NAME: RainFall

SPRINT: 7 and 8: (23/11/2023 And (27/11/2023)

Attendees	Backlog Grooming	Sprint Planning	Sprint Review	Sprint Retrospective	Daily Scrums
Client	✓	n/a	✓	n/a	n/a
WIL Coordinator	n/a	n/a	n/a	n/a	n/a
Lecturer	n/a	n/a	n/a	n/a	n/a
Armando Buffacchi	✓	✓	✓	✓	✓
Cornelius Ackerman	✓	✓	✓	✓	✓
Nomaswazi Sibeko	✓	✓	✓	✓	✓
Ryan Blignaut	✓	✓	✓	✓	✓

## 2.3.Supporting Image Evidence:



DevOps Start.docx

16 KB, Microsoft Word Document

Open

Save as...

Hey hows it guys.  
Can we maybe have a meeting later this afternoon.

Noma maybe Corne can help with the deliverable 3, I think that he had deigned a graphic for this process but this is the doc that I have.

09:12

Yeah he sent me the github pipeline

09:13

Ryan

Hey hows it guys.

Can we maybe have a meeting later this afternoon...

Thank you!

09:13

Armando

Yes I can meet later today... What time you thinking?

09:14

Armando

🔗

Jira

rainfallpplication.atlassian.net

lemme see what i can manage, try this link in the mean time:

<https://rainfallpplication.atlassian.net/jira/core/projects/RAIN/board?atOrigin=eyJpIjozMzZmE1YjExOTk0NGM0ZmI4M2NhZmYyZGY0OTFiYTU1Cjwloiaj9>

08:46

Forwarded

AppSweep

BUILD REPORT SUMMARY

# 1 Rainfall

com.corne.rainfall

Analyzed Sep 30 2023, 08:37

Version	App size	Comments Health
1.0	25.47 MB	No comment health found.

AppSweep- --11\_27\_2023.pdf

358 KB, Adobe Acrobat Document

Open

Save as...

08:50

50

## REFERENCE LIST

Android Developer, 2023. Build an offline-first app, 12 July 2023. [Online]. Available at: <https://developer.android.com/topic/architecture/data-layer/offline-first> [Accessed 6 October 2023].

Android Developer. 2023. Save simple data with SharedPreferences. *n.d.* [Online]. Available at: [Save simple data with SharedPreferences | Android Developers](#) [Accessed 6 October 2023].

AnyChart. 2023. Any Chart – JS Chart. *n.d.* [Online]. Available at: [Gallery | JavaScript Charting Library | AnyChart JS Charts](#) [Accessed 1 September 2023].

Boldt, J. 2022. Domain Driven Design — The Bounded Context. 15 April 2022. [Online]. Available at: [https://medium.com/@johnboldt\\_53034/domain-driven-design-the-bounded-context-1a5ea7bcb4a4](https://medium.com/@johnboldt_53034/domain-driven-design-the-bounded-context-1a5ea7bcb4a4). [Accessed 30 September 2023].

Developer.android.com 2023. Save data in a local database using Room, *n.d.* [Online] Available at: <https://developer.android.com/training/data-storage/room>. [Accessed 25 September 2023].

Dimitrilc. 2022. Android Native - How to inject dependencies into a Worker with Hilt. 11 April 2022. [Online]. Available at: [kotlin - Android Native - How to inject dependencies ... | DaniWeb](#) [Accessed 30 September].

Firebase. 2023. FireBase Authentication. *n.d.* [Online]. Available at: [Firebase Authentication \(google.com\)](#) [Accessed 30 September 2023].

Gathoni, M. 2022. Introduction to firebase storage, 30 April 2022. [Online]. Available at: <https://www.makeuseof.com/introduction-to-firebase-storage/> [Accessed 6 October 2023].

GoogleMaps Platform. 2023. Maps SDK For Android. *n.d.* [Online]. Available at: [Markers | Maps SDK for Android | Google for Developers](#) [Accessed 6 October 2023].

GuardSqaure. 2023. AppSweep. *n.d.* [Online]. Available at: [Mobile Application Security Testing | AppSweep \(guardsquare.com\)](#) [Accessed 30 September 2023].

JavaDoc. 2023. Annotation Type SerializedName. *n.d.* [Online]. Available at: [SerializedName \(Gson 2.6.2 API\) \(javadoc.io\)](#) [Accessed 10 October 2023].

JiraSoftware. 2023. Jira Board. *n.d.* [Online]. Available at: [Jira Scrum Boards | Atlassian](#) [Accessed 1 November 2023].

Kedia, P.2021. Creating the Network Module with Hilt. 9 July 2021. [Online]. Available at: [Creating the Network Module with Hilt | by Priyansh Kedia | CodeX | Medium](#) [Accessed 1 September 2023].

Mitchell, I. 2017. Walking Through a Definition of Ready. 24 August 2017. Online. Available at: [Walking Through a Definition of Ready | Scrum.org](#) [Accessed 7 September 2023].

Microsoft.com 2023. Model-View-ViewModel (MVVM). 11 April 2022. [Online] Available at: <https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm>. [Accessed 5 October 2023].

Qlik Q. 2023. Predictive Modelling. *n.d.* [Online]. Available at: [What is Predictive Modeling? Types & Techniques. \(qlik.com\)](#) [Accessed 24 November 2023].

Rai, K, S. 2023. API Calls with Retrofit in Android Kotlin: A Comprehensive Guide. 8 August 2023. [Online]. Available at: [API Calls with Retrofit in Android Kotlin: A Comprehensive Guide | by Kuldeep Singh Rai | Medium](#) [Accessed 6 October 2023].

Square.github.io/retrofit/ 2023.Retrofit, *n.d.* [Online] Available at: <https://square.github.io/retrofit/>. [Accessed 5 October 2023].

Sri. 2023. Best Practice for Handling API Calls Using Retrofit in Android Studio. 21 July 2023. [Online]. Available at: [Best Practice for Handling API Calls Using Retrofit in Android Studio | by Sri | Medium](#) [Accessed 6 October 2023].

Stackoverflow.com 2023. Gson Library in Android studio. 1 June 2016. [Online] Available at: <https://stackoverflow.com/questions/37048731/gson-library-in-android-studio> [Accessed 19 October 2023].

Taylor, M. 2023. An Android Studio Code Example for SQLite Database Import. 14 April 2023. [Online]. Available at: [Android: An Android Studio Code Example for SQLite Database Import \(copyprogramming.com\)](https://copyprogramming.com/android-studio-code-example-for-sqlite-database-import) [Accessed 30 September 2023].

Techradar.com 2023. Best cloud computing service of 2023. 24 November 2023. [Online] Available at: <https://techradar.com/best/best-cloud-computing-services> [Accessed 25 November 2023].

Towardsdatascience.com 2023. 10 Common Software Architectural Patterns in a nutshell. 4 September 2017. [Online] Available at: <https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013>. [Accessed 25 September 2023].

Wrike. 2023. What is Definition of Done. *n.d.* Online. Available at: <https://www.wrike.com/project-management-guide/faq/what-is-definition-of-done-agile/> [Accessed 7 September 2023].