##### CRISISAID – DISASTER MANAGEMENT APP

##### A MINI PROJECT REPORT

***Submitted by***

##### PRATEEKA K K [211422104346]

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***in***

COMPUTER SCIENCE AND ENGINEERING



**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**OCTOBER 2024**

##### PANIMALAR ENGINEERING COLLEGE

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

##### 

##### BONAFIDE CERTIFICATE

Certified that this project report **“CRISISAID – DISASTER MANAGEMENT APP”** is the bonafide work of  **PRATEEKA K K (211422104346)** who carried out the project work under my supervision.

SIGNATURE SIGNATURE

**Dr.L.JABASHEELA ,M.E.,Ph.D ., DR.M.S.VINMATHI,M.E.,Ph.D.,**

**HEAD OF THE DEPARTMENT SUPERVISOR**

ASSOCIATE PROFEESOR

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE , PANIMALAR ENGINEERING COLLEGE, NASARATHPETTAI, NASARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidates were examined in the End Semester Project Viva- Voce Examination held on...........................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kindwords and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors

**Tmt. C. VIJAYARAJESWARI , Dr. C . SAKTHIKUMAR , M.E. , Ph.D.,** and

**Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department , **Dr.L.JABASHEELA ,M.E.,Ph.D.,**

for the support extended throughout the project.

We would like to thank my Project Guide **DR.M.S.VINMATHI,M.E.,Ph.D** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

**NAME OF THE STUDENT**

**PRATEEKA K K(211422104346)**

ABSTRACT

Disaster preparedness and response have become crucial in mitigating the effects of natural calamities on vulnerable populations. CrisisAid is a comprehensive mobile application designed to enhance disaster management efforts by providing real-time weather predictions, early warning alerts, and crucial resource access. By leveraging cutting-edge geolocation and data integration technologies, the app offers timely notifications for disasters like storms, floods, and earthquakes, enabling communities to take preventive actions. Additionally, CrisisAid facilitates access to nearby shelters, medical services, and support networks through its resource mapping and crowdsourcing features. Users can contribute real-time reports on disaster impacts, helping authorities and NGOs allocate resources more efficiently. This report outlines the design, features, and implementation of CrisisAid, emphasizing its potential to improve disaster preparedness, response, and recovery by fostering community resilience and enhancing coordination efforts.

**TABLE OF CONTENTS**

ii

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | i |
|  | **LIST OF TABLES** | iv |
|  | **LIST OF FIGURES** | v |
| **1.** | **INTRODUCTION** | 01 |
|  | 1.1 Overview | 01 |
|  | 1.2 Problem Definition | 02 |
| **2.** | **SYSTEM ANALYSIS** | 03 |
|  | 2.1 Existing System | 03 |
|  | 2.2 Proposed System | 03 |
|  | 2.3 Development Environment | 04 |
| **3.** | **SYSTEM DESIGN** | 05 |
|  | 3.1 UML Diagrams | 05 |
|  | 3.2 Database Design | 10 |
|  | 3.3 ER Diagram | 12 |
|  | 3.4 Data Flow Diagram | 13 |
| **4.** | **SYSTEM ARCHITECTURE** | 16 |
|  | 4.1 Architecture Overview | 16 |
|  | 4.2 Module Description | 17 |
| **5.** | **SYSTEM IMPLEMENTATION** | 19 |
|  | 5.1 Coding for Main File | 19 |
|  | 5.2 Coding for Modules | 21 |
|  |  |  |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  |  |  |
| **6.** | **SYSTEM TESTING** | 32 |
|  | 6.1 Testcases and Reports | 32 |
| **7.** | **CONCLUSION** | 34 |
|  | 7.1 Conclusion | 34 |
|  | 7.2 Future enhancement | 34 |
| **8.** | **APPENDICES** | 35 |
|  | Sample Screenshots | 35 |
| **9.** | **REFERENCES** | 38 |

iii

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| TABLE NO | **TABLE**  **DESCRIPTION** | PAGE NO |
|  |  |  |
| 3.2.1 | Firebase Database Design | 11 |
| 6.1 | Testcases and Report Table for CrisisAid app | 32 |

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIG NO** | **FIGURE DESCRIPTION** | **PAGE NO** |
| 3.1.1 | Use Case Diagram for CrisisAid app | 5 |
| 3.1.2 | Class Diagram for CrisisAid app | 6 |
| 3.1.3 | Sequence Diagram for CrisisAid app | 7 |
| 3.1.4 | State chart Diagram CrisisAid app | 8 |
| 3.1.5 | Activity Diagram for CrisisAid app | 9 |
| 3.3 | ER Diagram for CrisisAid app | 12 |
| 3.4.1 | Dataflow Diagram level-0 | 13 |
| 3.4.2 | Dataflow Diagram level-1 | 14 |
| 3.4.3 | Dataflow Diagram level-2 | 15 |
| 4.1 | Architecture Diagram for CrisisAid app | 16 |
| 8.1 | Main Page | 35 |
| 8.2 | Home Page | 36 |
| 8.3 | Sample Outlook Of Project | 37 |

## CHAPTER 1

#### INTRODUCTION

* 1. **OVERVIEW**

Natural disasters, including earthquakes, floods, hurricanes, and wildfires, as well as man-made crises, such as industrial accidents or civil unrest, cause widespread disruption to communities and economies. The increasing frequency and severity of these events due to factors such as climate change, population growth, and urbanization have made disaster management a critical global challenge. Effective disaster management requires a holistic approach that encompasses early warning systems, timely information dissemination, resource allocation, and community involvement in preparedness, response, and recovery.

CrisisAid is an innovative disaster management mobile application designed to address these challenges by leveraging the power of mobile technology, data analytics, and community-driven insights. The app provides real-time weather predictions and early warning alerts for natural hazards like storms, floods, and earthquakes, enabling users to take necessary precautions before disasters strike. In addition to offering weather updates, CrisisAid serves as a comprehensive platform that connects users with essential resources such as nearby shelters, medical facilities, and food distribution centers.

A unique aspect of CrisisAid is its emphasis on community engagement through crowdsourced data. Users can report real-time disaster impacts, needs, and hazards in their local areas, contributing to a collective pool of information that helps authorities and NGOs make informed decisions. This collaborative feature enhances the app’s effectiveness in disaster response and recovery efforts, improving resource allocation and support services where they are needed most. By integrating predictive analytics and geolocation features, CrisisAid ensures that users receive timely and relevant information based on their location and potential risks.

Overall, CrisisAid aims to empower individuals and communities to respond proactively to disasters, reduce risks, and accelerate recovery processes. The app addresses the full spectrum of disaster management, from preparedness and early warnings to post-disaster support, making it an essential tool for enhancing resilience in the face of crises.

* 1. **Problem Statement**

In disaster situations, the availability of accurate, real-time information is often limited, which exacerbates the severity of the situation. Traditional systems for disaster management are fragmented, with delayed communication, insufficient integration of services, and limited access to critical resources. This lack of coordination results in ineffective preparedness, delayed response times, and misallocation of resources, ultimately increasing the impact of the disaster on affected populations.

While several mobile applications provide weather alerts or disaster-related information, they often lack a holistic approach that covers the entire disaster management lifecycle. There is a need for a platform that offers not only early warnings but also facilitates access to shelters, medical facilities, and food distribution points, while empowering communities to contribute real-time data to aid organizations. This is the gap that CrisisAid aims to fill.

## CHAPTER 2

#### SYSTEM ANALYSIS

* 1. **EXISTING SYSTEM**

  Several existing disaster management applications aim to enhance public safety during emergencies. The FEMA Disaster App provides alerts and safety tips but often suffers from delayed notifications. The Red Cross Emergency App offers local shelter information but lacks adequate localization for immediate threats. Similarly, the Disaster Alert app aggregates global disaster data but struggles with resource mapping, complicating access to nearby services. While apps like MyShake provide earthquake alerts, they often miss community reporting features that enhance situational awareness. Despite their functionalities, these systems typically present complex interfaces that hinder user experience and rely on static content, failing to deliver real-time updates. These limitations underscore the need for a more integrated and user-friendly solution like CrisisAid.

###### DISADVANTAGES

* Delayed Notifications
* Limited Localized Information
* Inadequate Resource Mapping
* Static Content and Limited Updates

#### PROPOSED SYSTEM

#### CrisisAid is specifically designed to overcome the limitations of existing disaster management systems by offering a comprehensive, user-centered solution that prioritizes the needs of individuals and communities in times of crisis. Key features of the proposed system include:

**Real-time Weather Predictions:** Utilizing advanced meteorological data, CrisisAid delivers accurate forecasts and timely alerts for impending disasters, ensuring users are informed and can take proactive measures.

**Localized Early Warning Alerts**: The application sends customized notifications tailored to users' geographic locations, empowering them to respond effectively to specific threats in their area.

**Resource Access and Support Services:** Integrating an interactive mapping feature, CrisisAid connects users to nearby shelters, food distribution centers, and medical facilities, facilitating quick access to essential services during emergencies.

#### DEVELOPMENT ENVIROMENT SOFTWARE REQUIREMENT

* + - Programming Language : Java
    - Development Environment: Android Studio
    - Operating System
    - Database : Firebase
    - Accessibility Features

#### HARDWARE REQUIREMENT

* + - Mobile Device
    - Internet Connectivity
    - Audio Output
    - GPS Module.

## CHAPTER 3

### SYSTEM DESIGN

#### UML DIAGRAMS

#### 3.1.1 Use case diagram:

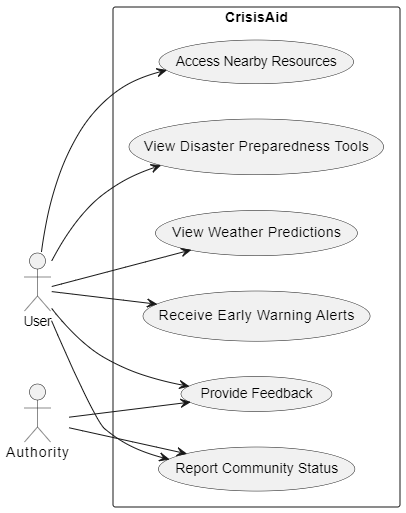
****

Fig 3.1.1 Use case diagram for CrisisAid.

This use case diagram refers to activities done by System and users and their corresponding use cases.

#### 3.1.2 Class diagram:

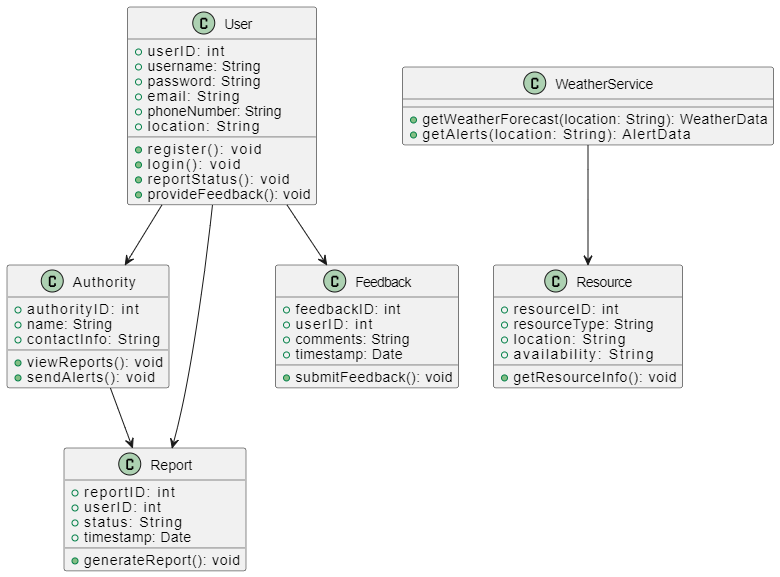


Fig 3.1.2 Class diagram for CrisisAid app

The class diagram for the CrisisAid application outlines key components, including User, Authority, WeatherService, Resource, Report, and Feedback. Each class encapsulates relevant attributes and methods, defining relationships that facilitate user interactions and resource management. This diagram serves as a foundational blueprint for developing the application's critical disaster management features.

#### 3.1.3 Sequence diagram:

#### 

Fig 3.1.3 Sequence diagram for CrisisAid app

The sequence diagram for the CrisisAid application illustrates the interactions between the user, the application, the weather service, and the authority during key processes. It demonstrates how the user logs in, requests weather forecasts, and reports community status, highlighting the flow of information and acknowledgments among the components. This diagram effectively captures the dynamic behavior of the system, providing a clear representation of user interactions and the sequence of events that facilitate effective disaster management.

#### 3.1.4 State Chart diagram:

#### 

#### The state diagram shows the app's flow, starting from idle, moving through monitoring, alert, response, and recovery phases, before returning to idle. Each state transition is clearly marked in a vertical layout.

#### 3.1.5 Activity diagram:

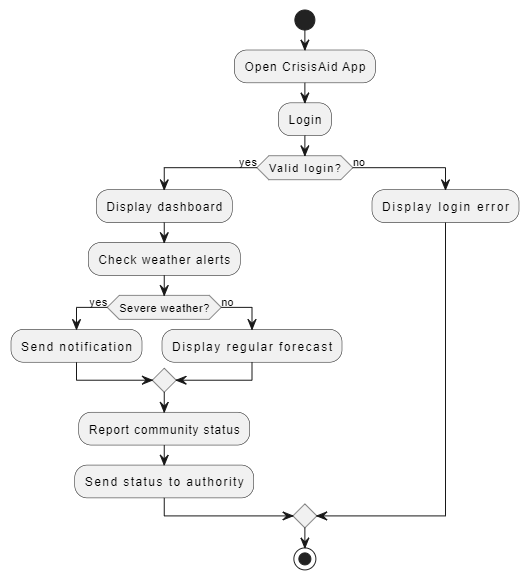


Fig 3.1.5 Activity diagram for CrisisAid app

The activity diagram illustrates the dynamic flow of control in the CrisisAid system, detailing the sequence of actions from user input to system response. It shows how users interact with key features such as weather alerts, resource access, and community reporting, emphasizing the app's efficiency in delivering timely and relevant information. The diagram provides a clear overview of the app’s workflow, ensuring that all processes are streamlined and responsive, enhancing the overall user experience during emergencies.

#### DATABASE DESIGN

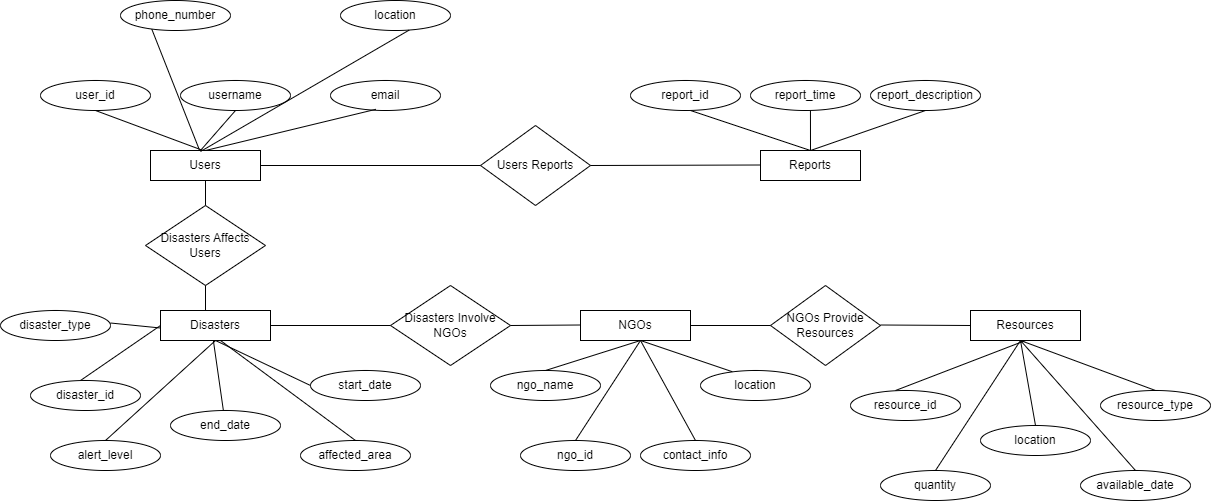
The CrisisAid app database is designed to manage disaster-related information effectively while maintaining scalability and performance. It stores vital data such as user information, real-time weather updates, resource availability (e.g., shelters and medical facilities), and community reports. Each entity is linked through relationships to ensure quick retrieval of location-specific data and efficient communication between users and authorities. The structure ensures data integrity, with necessary foreign keys and constraints. This system also facilitates user feedback, enabling continuous improvements to the app.

#### 

**Table 3.2.1.Firebase Database Design**

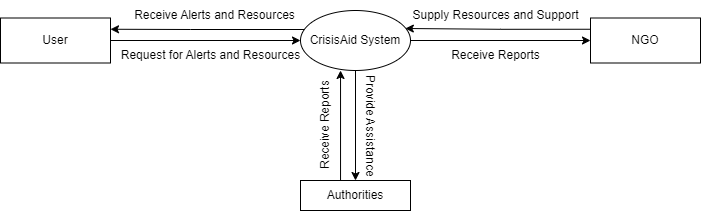
|  |  |  |
| --- | --- | --- |
| **Collection Name** | **Attributes** | **DESCRIPTION** |
| Users | -UserID   |  | | --- | |  |  |  | | --- | | -Username | | -Email  -Phone  -Location | | Stores unique identifier for each user, email, phone, and location details. |
| Weather | -LocationID  -Forecast   |  | | --- | | -Alert | | Weather forecast and disaster alert details. |
| Locations | -LocationID  -City  -State  -Country | Geographic coordinates (latitude, longitude). |
| Reports | -ReportID  -UserID  -LocationID  -Status | Contains user status reports, linked to the user and location. |
| Resources | -ResourceID  -Name | Information about available resources (e.g., shelters, medical facilities). |

#### ER DIAGRAM



The ER diagram for the CrisisAid system visually represents the relationships between key entities such as users, locations, weather data, reports, and resources. It highlights how user data is connected with geographical information, real-time weather updates, and emergency resources, ensuring efficient data management and access. By mapping out these relationships, the ER diagram helps in understanding the structure of the database, supporting the system’s functionality and ensuring data integrity across the application.

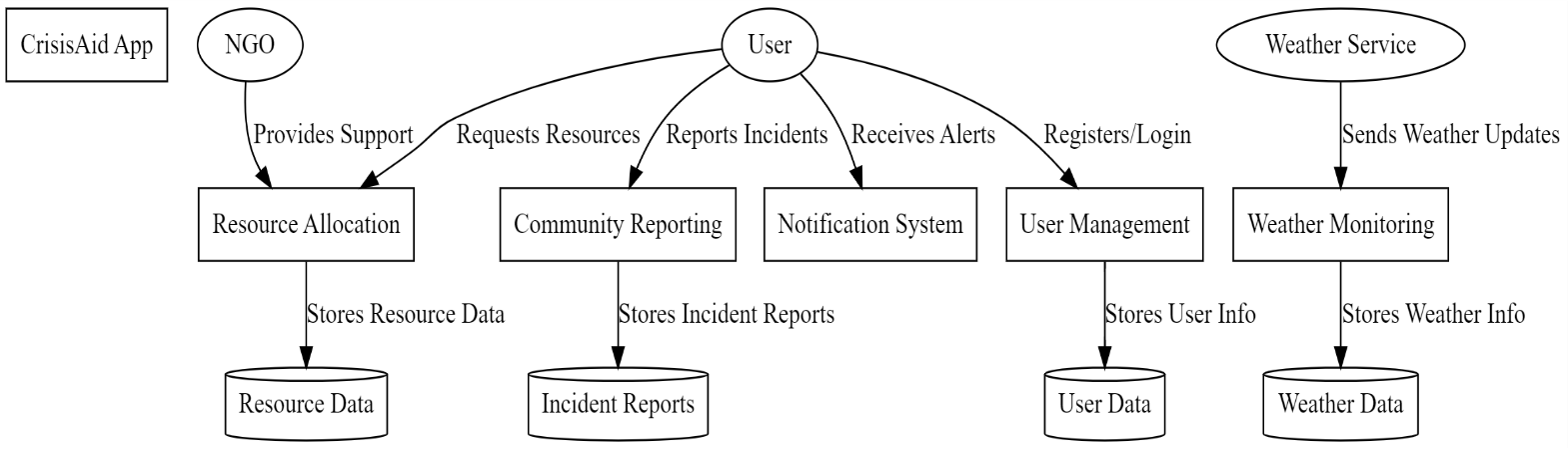
#### DATAFLOW DIAGRAM 3.4.1 0 LEVEL DFD

****

##### Fig 3.4.1 Data flow diagram level 0

The zero level of data flow diagram of CrisisAid app shows the various management levels.

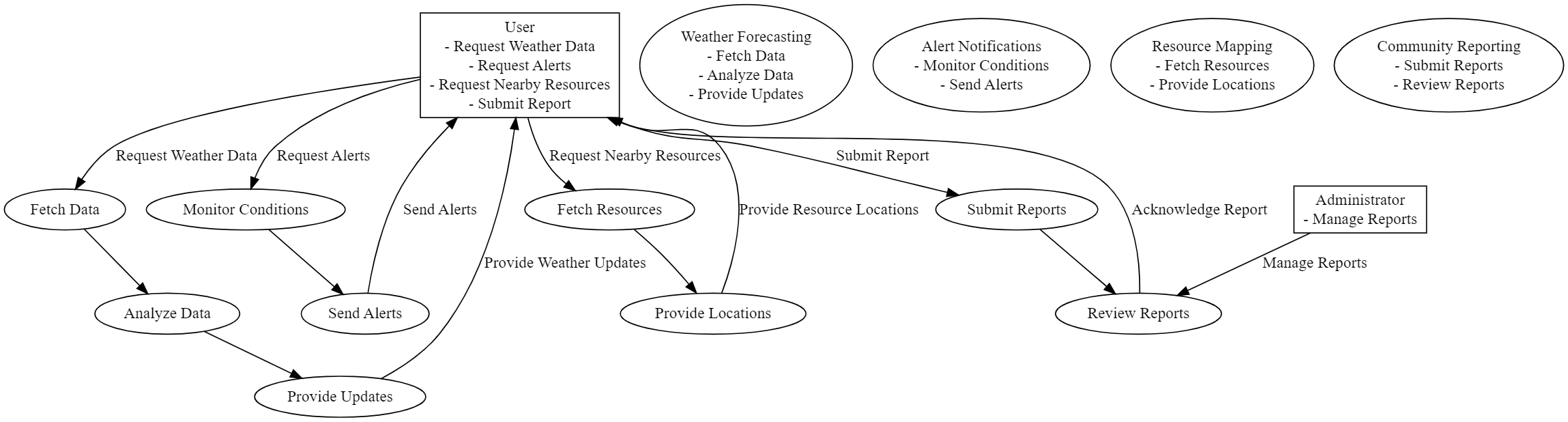
##### 3.4.2 FIRST LEVEL DFD

****

**Fig 3.4.2 Dataflow diagram level 1**

The first level of data flow diagram of CrisisAid app shows the various management levels and their corresponding report.

#### 3.4.3 SECOND LEVEL DFD

****

##### 

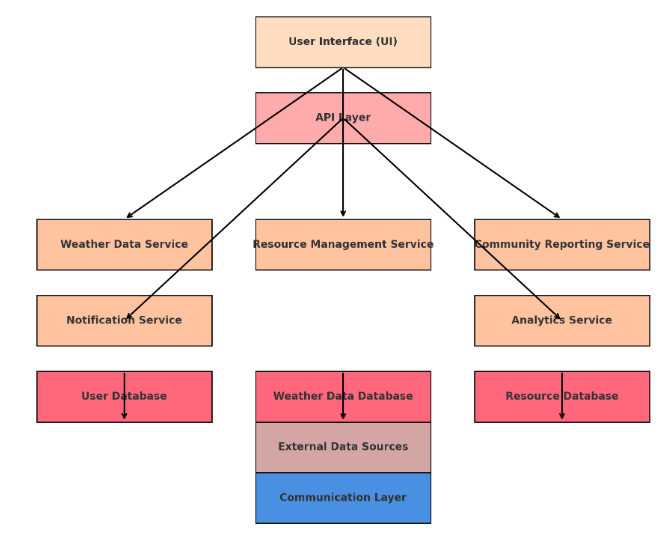
##### Fig 3.4.3 Dataflow diagram level 2

The second level of data flow diagram of CrisisAid app shows the various details of actions.

## CHAPTER 4

### SYSTEM ARCHITECTURE

#### ARCHITECTURE OVERVIEW



##### Fig 4.1 Architecture diagram for CrisisAid

The figure 4.1 consists of entire architecture diagram for the CrisisAid. The architecture diagram of the CrisisAid app outlines its key components and interactions for effective disaster management. Central to the system is the CrisisAid Application, which features a User Interface for seamless access to real-time weather updates and resources. The Weather Prediction Module delivers accurate forecasts, while the Alert System sends timely notifications based on user locations. Additionally, the Resource Mapping Module connects users to nearby shelters and medical facilities, and the Community Reporting System allows users to share real-time information, enhancing communication and resource allocation during disasters.

#### MODULE DESCRIPTION

CrisisAid consists of 7 main modules. They are

* + - User Management Module.
    - Weather Prediction Module.
    - Alert System Module.
    - Resource Mapping Module.
    - Community Reporting Module.
    - Data Integration Module.
    - User Feedback Module.

**User Management Module**:

* This module provides an intuitive and user-friendly interface that allows users to easily navigate the app. It features quick access to critical information, including weather updates, alerts, and resources. The design prioritizes usability, ensuring that users can find the information they need during emergencies with minimal effort.

**Weather Prediction Module**:

* Leveraging advanced meteorological data, this module delivers accurate real-time weather forecasts and predictions. It analyzes historical and current weather patterns to issue alerts for impending disasters, such as storms or floods, empowering users to prepare adequately and take necessary precautions.

**Alert System Module**:

* The Alert System is responsible for generating and sending localized early warning notifications to users based on their geographic locations. It ensures timely communication of critical information regarding weather changes and disaster alerts, enabling users to take prompt action to safeguard their safety.

**Resource Mapping Module**:

* This module integrates a mapping feature that connects users to nearby resources such as shelters, food distribution centers, and medical facilities. By providing location-based services, users can quickly find and access essential support during a disaster, facilitating efficient resource allocation.

**Community Reporting Module**:

* The Community Reporting Module allows users to share real-time information about their situations during emergencies. This feature encourages community engagement and crowdsourcing, enabling users to report their needs, share updates, and assist authorities in assessing the situation for better resource deployment.

**Data Integration Module**:

* This module facilitates the integration of various data sources to enhance the app's predictive capabilities. By analyzing historical data and real-time inputs, it improves the accuracy of weather predictions and resource management, supporting better decision-making during disasters.

**User Feedback Module**:

* This module collects user feedback and insights regarding the app's performance and features. By analyzing this data, developers can make continuous improvements and adapt the app to better meet user needs, ensuring it remains effective in disaster management scenarios.

## CHAPTER 5

### SYSTEM IMPLEMENTATION

#### CODING FOR MAIN FILE:

#### activity\_main.xml:

#### <?xml version="1.0" encoding="utf-8"?> <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:background="@drawable/loginbackground"> <!-- Login Button --> <Button android:id="@+id/loginButton" android:layout\_width="200dp" android:layout\_height="60dp" android:text="Login" style="@style/CustomButtonStyle" android:gravity="center" android:padding="10dp" android:layout\_centerInParent="true" /> <!-- Signup Button --> <Button android:id="@+id/signUpButton" android:layout\_width="200dp" android:layout\_height="60dp" android:text="Sign Up" style="@style/CustomButtonStyle" android:gravity="center" android:padding="10dp" android:layout\_below="@id/loginButton" android:layout\_centerHorizontal="true" android:layout\_marginTop="16dp" /> </RelativeLayout>

#### MainActivity.java:

#### package com.example.crisisaid; import android.content.Intent; import android.os.Bundle; import android.view.View; import android.widget.Button; import androidx.appcompat.app.AppCompatActivity; public class MainActivity extends AppCompatActivity { @Override protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.*activity\_main*); // Find the Login and Signup buttons Button loginButton = findViewById(R.id.*loginButton*); Button signupButton = findViewById(R.id.*signUpButton*); // Set click listener for the Login button loginButton.setOnClickListener(new View.OnClickListener() { @Override public void onClick(View v) { // Navigate to the LoginDetailsActivity Intent intent = new Intent(MainActivity.this, LoginDetailsActivity.class); startActivity(intent); } }); // Set click listener for the Signup button signupButton.setOnClickListener(new View.OnClickListener() { @Override public void onClick(View v) { // Navigate to the SignupActivity Intent intent = new Intent(MainActivity.this, SignupActivity.class); startActivity(intent); } }); } }

#### CODING FOR MODULES

#### activity\_login.xml

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@drawable/loginbackground">  
  
 <EditText  
 android:id="@+id/username"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/username"  
 android:background="@drawable/edittext\_background"  
 android:padding="16dp"  
 android:layout\_margin="16dp" />  
  
 <EditText  
 android:id="@+id/password"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/password"  
 android:background="@drawable/edittext\_background"  
 android:padding="16dp"  
 android:layout\_marginTop="16dp"  
 android:layout\_below="@id/username" />  
  
 <Button  
 android:id="@+id/submitLogin"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/submit\_login"  
 android:background="@drawable/button\_background"  
 android:textColor="#FFFFFF"  
 android:padding="16dp"  
 android:textSize="18sp"  
 android:layout\_below="@id/password"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="24dp" />  
  
</RelativeLayout>

#### LoginActivity.java

package com.example.crisisaid; // Change this to your app's package name

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class LoginActivity extends AppCompatActivity {

private EditText usernameEditText;

private EditText passwordEditText;

private Button submitButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login); // Load the login layout

// Initialize the UI elements

usernameEditText = findViewById(R.id.username);

passwordEditText = findViewById(R.id.password);

submitButton = findViewById(R.id.submitLogin);

// Set up the submit button click listener

submitButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

handleLogin(); // Handle the login process

}

});

}

// Method to handle login process

private void handleLogin() {

// Retrieve text from EditText fields

String username = usernameEditText.getText().toString();

String password = passwordEditText.getText().toString();

// Simple validation

if (username.isEmpty() || password.isEmpty()) {

Toast.makeText(this, "Please enter both username and password", Toast.LENGTH\_SHORT).show();

} else {

// For demonstration purposes, show a Toast message

Toast.makeText(this, "Login Successful", Toast.LENGTH\_SHORT).show();

// You can add logic to validate the credentials or proceed to another activity

}

}

}

#### activity\_signup.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp">

<!-- Username Input Field -->

<EditText

android:id="@+id/signupUsername"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Username"

android:inputType="text"

android:padding="10dp"

android:background="@drawable/edittext\_background" />

<!-- Email Input Field -->

<EditText

android:id="@+id/signupEmail"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Email"

android:inputType="textEmailAddress"

android:padding="10dp"

android:layout\_marginTop="10dp"

android:background="@drawable/edittext\_background" />

<!-- Phone Number Input Field -->

<EditText

android:id="@+id/signupPhoneNumber"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Phone Number"

android:inputType="phone"

android:padding="10dp"

android:layout\_marginTop="10dp"

android:background="@drawable/edittext\_background" />

<!-- Location Input Field -->

<EditText

android:id="@+id/signupLocation"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Location (City/Region)"

android:inputType="text"

android:padding="10dp"

android:layout\_marginTop="10dp"

android:background="@drawable/edittext\_background" />

<!-- Preferred Language Input Field -->

<EditText

android:id="@+id/signupLanguage"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Preferred Language"

android:inputType="text"

android:padding="10dp"

android:layout\_marginTop="10dp"

android:background="@drawable/edittext\_background" />

<!-- Emergency Contact Input Field -->

<EditText

android:id="@+id/signupEmergencyContact"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Emergency Contact"

android:inputType="phone"

android:padding="10dp"

android:layout\_marginTop="10dp"

android:background="@drawable/edittext\_background" />

<!-- Password Input Field -->

<EditText

android:id="@+id/signupPassword"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Password"

android:inputType="textPassword"

android:padding="10dp"

android:layout\_marginTop="10dp"

android:background="@drawable/edittext\_background" />

<!-- Confirm Password Input Field -->

<EditText

android:id="@+id/signupConfirmPassword"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Confirm Password"

android:inputType="textPassword"

android:padding="10dp"

android:layout\_marginTop="10dp"

android:background="@drawable/edittext\_background" />

<!-- Signup Button -->

<Button

android:id="@+id/signupButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Sign Up"

android:layout\_marginTop="20dp"

android:textSize="18sp"

android:background="@drawable/button\_background"

android:textColor="@android:color/white" />

</LinearLayout>

**SignupActivity.java**

package com.example.crisisaid;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class SignupActivity extends AppCompatActivity {

private EditText usernameEditText;

private EditText passwordEditText;

private EditText emailEditText;

private Button signupButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_signup); // Load the signup layout

// Initialize the UI elements

usernameEditText = findViewById(R.id.signupUsername);

passwordEditText = findViewById(R.id.signupPassword);

emailEditText = findViewById(R.id.signupEmail);

signupButton = findViewById(R.id.signupButton);

// Set up the signup button click listener

signupButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

handleSignup(); // Handle the signup process

}

});

}

// Method to handle signup process

private void handleSignup() {

// Retrieve text from EditText fields

String username = usernameEditText.getText().toString();

String password = passwordEditText.getText().toString();

String email = emailEditText.getText().toString();

// Simple validation

if (username.isEmpty() || password.isEmpty() || email.isEmpty()) {

Toast.makeText(this, "Please fill out all fields", Toast.LENGTH\_SHORT).show();

} else {

// For demonstration purposes, show a Toast message

Toast.makeText(this, "Signup Successful", Toast.LENGTH\_SHORT).show();

// You can add logic to handle the signup process or store the user data

}

}

}

#### AndroidManifest.xml

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.crisisaid">

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/Theme.CrisisAid">

<!-- Main Activity that starts when the app is launched -->

<activity

android:name=".MainActivity"

android:exported="true"> <!-- Make sure to specify android:exported -->

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity

android:name=".LoginActivity"

android:exported="false" /> <!-- Specify android:exported -->

<activity

android:name=".SignupActivity"

android:exported="false" /> <!-- Specify android:exported -->

<activity

android:name=".LoginDetailsActivity"

android:exported="false" /> <!-- Specify android:exported -->

</application>

</manifest>

## CHAPTER 6 SYSTEM TESTING

**6.1 TEST CASES & REPORTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST**  **CASE ID** | **TESTCASE/**  **ACTION TO BE PERFORMED** | **EXPECTED RESULT** | **ACTUAL RESULT** | **PASS/ FAIL** |
| 1 | Verify user registration with valid details | User should be successfully registered. | User registered successfully. | Pass |
| 2 | Verify user login with valid credentials | User should be able to log in successfully. | |  | | --- | | User logged in successfully. |  |  | | --- | |  | | Pass |
| 3 | Verify login with invalid credentials | User should receive an error message. | |  | | --- | | Error message displayed. |  |  | | --- | |  | | Pass |
| 4 | |  | | --- | | Test real-time weather updates |  |  | | --- | |  | | Weather data should be updated every few minutes. | Weather data updated every 5 minutes. | Pass |
| 5 | Check functionality of localized alerts | User should receive alerts based on their location. | Alerts received as expected. | Pass |
| 6 | Test resource mapping feature | Users should see nearby shelters and facilities. | Nearby shelters displayed correctly. | pass |
| 7 | Verify community reporting functionality | Users should be able to submit reports successfully. | Reports submitted successfully. | Pass |
| **TEST**  **CASE ID** | **TESTCASE/**  **ACTION TO BE PERFORMED** | **EXPECTED RESULT** | **ACTUAL RESULT** | **PASS/ FAIL** |
| 8 | Validate feedback submission | Feedback should be submitted and acknowledged. | |  | | --- | | Feedback acknowledged |  |  | | --- | |  | | Pass |
| 9 | Test app performance under heavy load | App should remain responsive without crashes. | App remained responsive. | Pass |
| 10 | Verify integration with external data sources | App should correctly fetch and display data. | |  | | --- | | Data fetched and displayed accurately. |  |  | | --- | |  | | Pass |

## CHAPTER 7

### CONCLUSION

CrisisAid is an innovative disaster management application that effectively addresses the limitations of existing systems by providing essential features for real-time weather predictions, localized early warning alerts, and streamlined access to resources. Its user-centered design not only enhances community preparedness and response during emergencies but also fosters active engagement and collaboration among users. By integrating advanced technology and facilitating community reporting, CrisisAid empowers individuals to make informed decisions that can save lives and reduce the impact of disasters. As the application continues to evolve through user feedback and technological advancements, it holds the potential to significantly improve disaster management practices, ultimately contributing to safer and more resilient communities.

#### FUTURE ENHANCEMENTS

CrisisAid aims to continually evolve by integrating enhanced predictive analytics through advanced machine learning algorithms for more accurate weather forecasts and disaster predictions. Future enhancements include the incorporation of Internet of Things (IoT) technology to collect real-time environmental data from sensors, expanding language support to cater to diverse user populations, and developing educational modules for disaster preparedness. Additionally, collaborating with local authorities will improve resource allocation and verification of shelter information. Implementing offline functionality will ensure access to critical features during network outages, while a robust feedback mechanism will facilitate continuous improvement based on user experiences. The introduction of community engagement features, customizable alerts, and interactive data visualization tools will further empower users, making CrisisAid a more effective and user-friendly disaster management solution.

## CHAPTER 8 APPENDICES

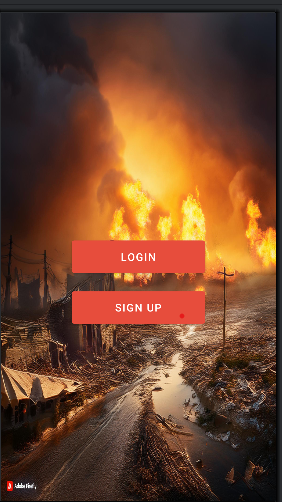
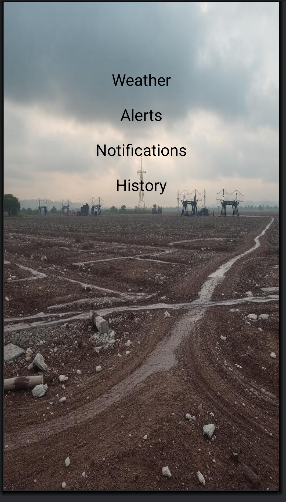


Fig 8.1 Main Page

****

**Fig 8.2 Home Page**

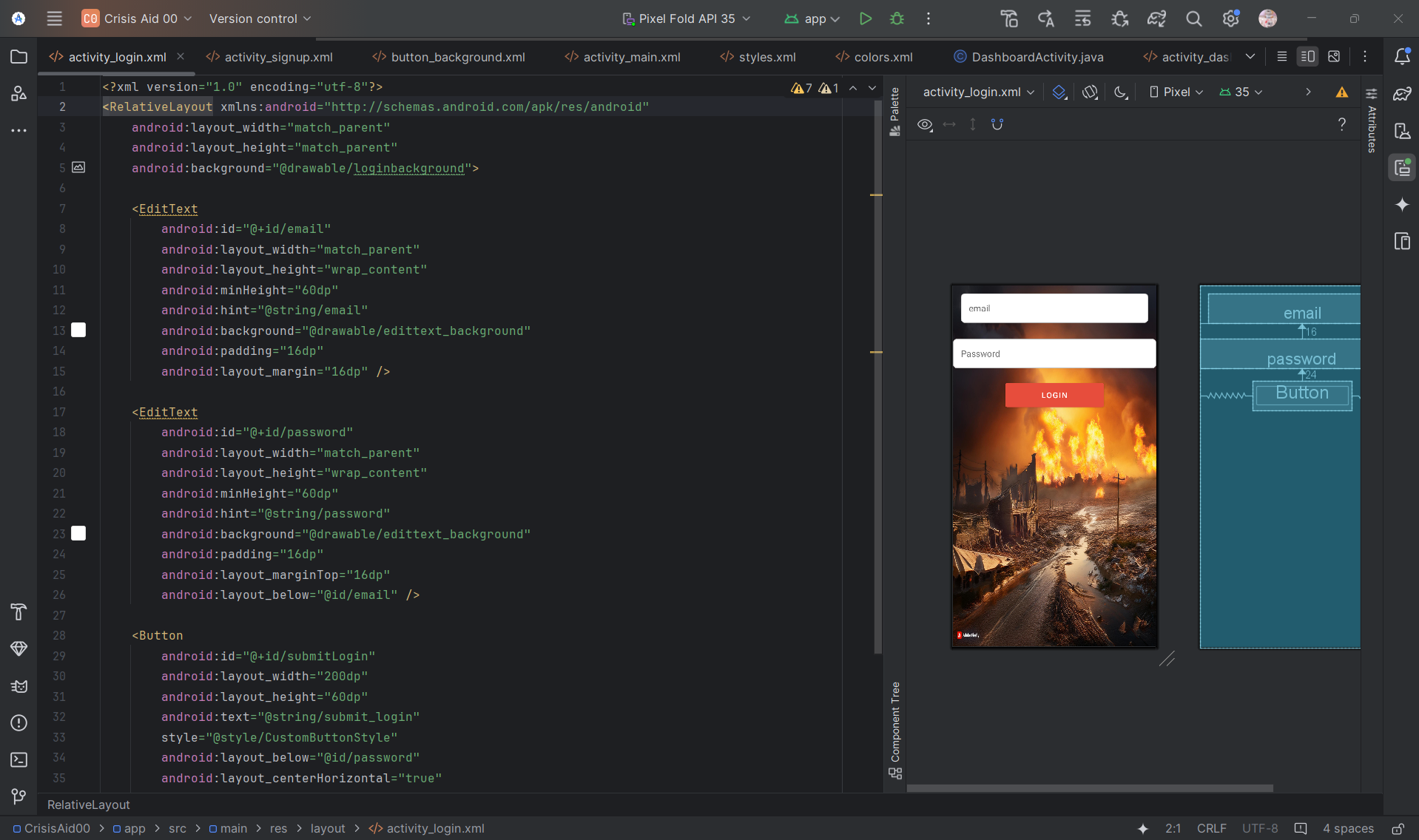
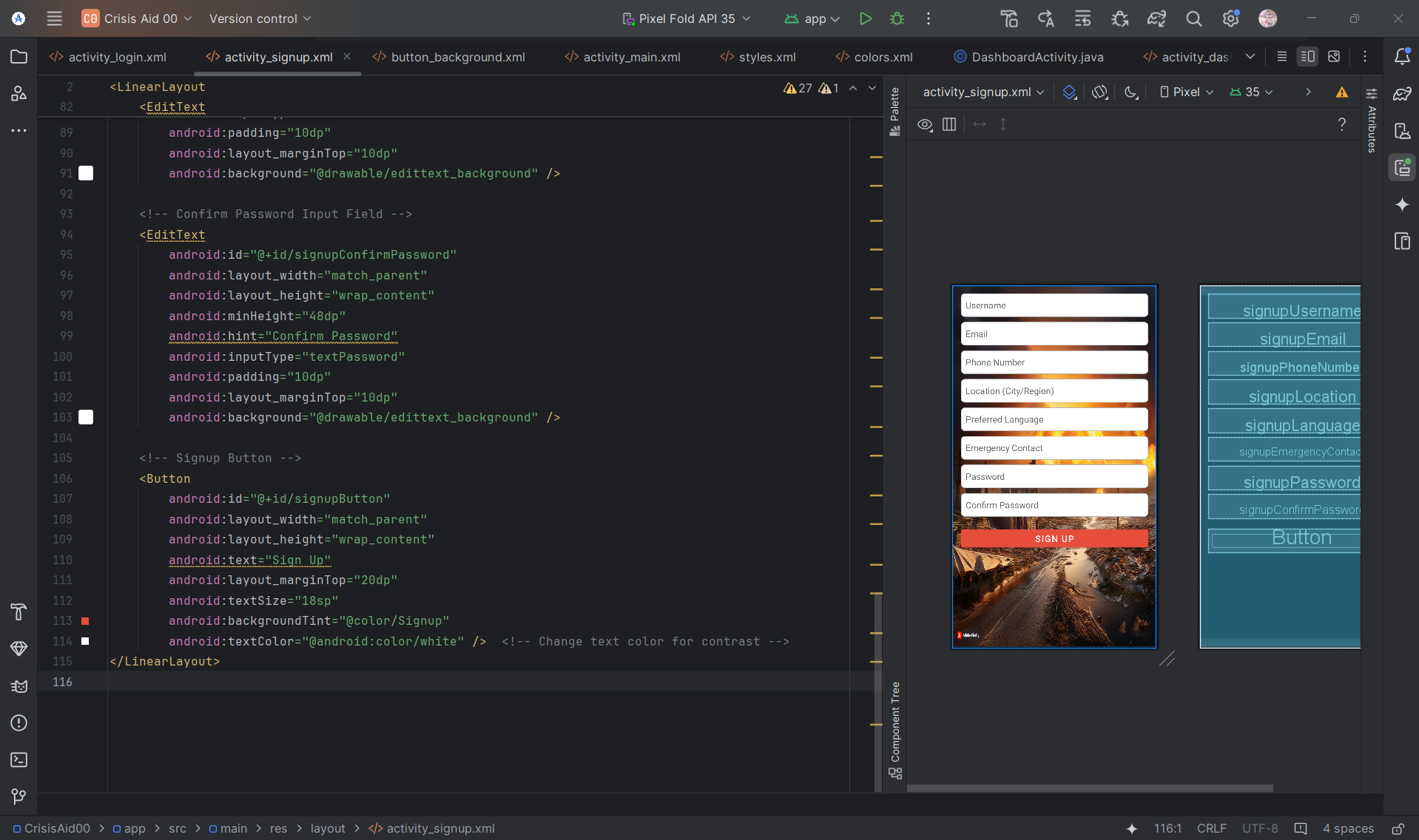


Fig 8.3 Sample Outlook of Project

****

**Fig 8.4 Sample Outlook Of Project**

## CHAPTER 9

### REFERENCES

[1] [SoYoung Park](https://ieeexplore.ieee.org/author/37086769883);[MinKyo Youm](https://ieeexplore.ieee.org/author/37087326004);[JunWoo Lee](https://ieeexplore.ieee.org/author/37087326493), [3D disaster management a study on utilization of smart CCTV-based 3D geo-spatial information](https://ieeexplore.ieee.org/document/7402015/),[2015 2nd International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)](https://ieeexplore.ieee.org/xpl/conhome/7396014/proceeding)

[2] [Lai Wei](https://ieeexplore.ieee.org/author/37591869300); [Hong Chen](https://ieeexplore.ieee.org/author/37293467300),[Emergent Management for Consumable Industry Chains after Natural Disasters](https://ieeexplore.ieee.org/document/5279869/), [2009 International Conference on Management of e-Commerce and e-Government](https://ieeexplore.ieee.org/xpl/conhome/5279723/proceeding)

[3] [Xinliang Ye](https://ieeexplore.ieee.org/author/37980915200); [Jiahong Wen](https://ieeexplore.ieee.org/author/37962736900); [Peiyi Ding](https://ieeexplore.ieee.org/author/37989928600), [Integrated natural disasters risk management in tourism destination—A case study of 5.12 WenChuan Earthquake](https://ieeexplore.ieee.org/document/5492819/), [2010 IEEE International Conference on Management of Innovation & Technology](https://ieeexplore.ieee.org/xpl/conhome/5481670/proceeding).

[4] [Yida Fan](https://ieeexplore.ieee.org/author/37065942600);[Wei Wu](https://ieeexplore.ieee.org/author/37085903243);[Ming Liu](https://ieeexplore.ieee.org/author/37085904057);[Suju Li](https://ieeexplore.ieee.org/author/37085892809);[Haixia He](https://ieeexplore.ieee.org/author/38555232800);[Yang Shu](https://ieeexplore.ieee.org/author/37085888707)o, [Capacity analysis of GF—4 on the disaster management](https://ieeexplore.ieee.org/document/7729971/), [2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)](https://ieeexplore.ieee.org/xpl/conhome/7592514/proceeding).

[5] [Ko Ko Lwin](https://ieeexplore.ieee.org/author/37085673822);[Yoshihide Sekimoto](https://ieeexplore.ieee.org/author/38028572600);[Wataru Takeuchi](https://ieeexplore.ieee.org/author/37323280700);[Koji Zettsu](https://ieeexplore.ieee.org/author/37283995500), [City Geospatial Dashboard: IoT and Big Data Analytics for Geospatial Solutions Provider in Disaster Management](https://ieeexplore.ieee.org/document/9032921/), [2019 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)](https://ieeexplore.ieee.org/xpl/conhome/9018169/proceeding)

[6] Ali Benssam; Nadia Nouali-Taboudjemat; Omar Nouali,

Requirements for an IT based platform for disaster management, 2014 1st International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)

[7] Chippi Mohanan;Vivek Menon. Disaster management in India — An analysis using COBIT 5 principles. 2016 IEEE Global Humanitarian Technology Conference (GHTC).

[8] Xinliang Ye; Jiahong Wen. Study on Disaster Risk Management Framework in Tourist Destination, 2009 International Conference on Environmental Science and Information Application Technology.

[9] Romain Pujol; Razvan Stanica; Fabrice Valois. LTE User Association for Self-Deployable Networks in Disaster Management Scenarios. 2019 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM).

[10] F. Franchi; A. Marotta; C. Rinaldi; F. Graziosi; L.D'Errico. IoT-based Disaster Management System on 5G . 2019 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM).