



Indian Institute of Engineering Science and Technology Shibpur

Department of Computer Science and Technology

Mini Project Report

On

Development of Android app for Disaster Reporting

for

4th Semester '20

Team

Sarbik Betal(ID NO:510518029)

Subham Dhar(ID NO:510518018)

Dipendu Mahata(ID NO:510518040)

Sayan Kumar Maity(ID NO:510518012)

Mentored By
Dr. S. Das Bit

CERTIFICATE

This is to certify that the progress report entitled
Development of Android app for Disaster Reporting
is an authentic record of the work carried out by SARBIK
BETAL(ID:51051829), SAYAN KUMAR MAITY (ID:510518012)
,DIPENDU MAHATA (ID: 510518040) and SUBHAM
DHAR(ID:510518018) from The Department of Computer Science
and Technology, IEST Shibpur under my guidance and. In my
opinion, this progress report satisfies the requirement for which it
is submitted. To the best of my knowledge and belief, it has not
been submitted to any other institutes or universities for the
award of degree/diploma.

Dr.SIPRA DAS BIT(Professor)
Dept. of Computer Science and Technology
Indian Institute of Engineering Science and Technology
Shibpur

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely fortunate to have got this all along the completion of my project work. Whatever I have done is only due to such guidance and assistance and I would not forget to thank them. I respect and thank Prof. Sipra Das Bit, for giving me an opportunity to do the project work and providing us all support and guidance which made me complete the project on time. I owe my profound gratitude to our project guide Nabanita Das, who took keen interest on our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system. I am thankful to the Department of Computer Science and Technology gave us a chance to build a project in the process of which we learnt many new concepts.

Index

Description	Pg No.
Abstract	4
Motivation	5
Introduction	6-11
Technology Stack	12
• Android	12
• Dart	12
• Flutter	12
• Hive DB	13-14
• NodeJS	15
• ExpressJS	16
• JWT	16
• MongoDB	17
• Mongoose ODM	17
API Server	18-20
Client Application	21-24
Deployment	25
Conclusion	26
Future Scope	27

Abstract

Our aim is to create an app which will aid the disaster relief and recovery operations by maintaining a database of the victims in various locations of the disaster affected region this database can be handled by any volunteer groups involved in the relief operations and can be utilized effectively by the government bodies to handle the situation effectively with need-specific support to the sufferers within appropriate time. This project is aimed at developing a user-friendly mobile (Android) application to aid the process of disaster management by providing an efficient data base system to deliver the necessary details to the rescue teams in the operations involved during such calamities.

Motivation

There is not a single country that can be considered free from various types of disasters. There is no doubt that the vulnerability of the people in the underdeveloped country is more than those of their developed counterparts. This can be attributed to various factors. One of them is extensive research and awareness generation activity undertaken by the governments of these countries. This could be possible only if the authorities understand the long-term effects of disaster on the overall economic development of the country. In the era of 24-hour news and social media, most of us watch powerlessly from the side-lines as disasters - from floods and earthquakes to terrorist attacks and plane crashes - unfold on our screens. But if one wants to play a role in preparing for these events and minimising the impact of such natural or man-made disasters, or get involved in the emergency response when they occur, it becomes necessary to make sure that the common public is prepared to combat such situations. One way to help resolve these problems is to give motivated conscientious people the knowledge and skills to make a difference. Hence, the project embarks upon a small part of the larger and vital journey to aid the relief and recovery operations after such calamities. It is a humble effort to bring about this change so that these calamities (which will continue to occur) do not take forms threatening the entire mankind. With this motivation in mind this app is going to act as an intermediate between the victims and the authorities allowing speedy provision of aid.

INTRODUCTION

Disaster Management

We act before, during and after disasters strike, often providing assistance in some of the world's most hostile environments.

Our disaster management activities seek to:

- Save lives and reduce human suffering
- Protect and restore livelihoods
- Reduce the risks faced by communities affected by disaster and conflict.

The Red Cross and Red Crescent Societies^T define disaster management as the organisation and management of resources and responsibilities for dealing with all humanitarian aspects of emergencies, in particular preparedness, response and recovery in order to lessen the impact of disasters.

Operations Involved In The Process Of Disaster Management

Local, regional, national and international organisations are all involved in mounting a humanitarian response to disasters. Each will have a prepared disaster management plan. These plans cover prevention, preparedness, relief and recovery.

- **Disaster prevention:** These are activities designed to provide permanent protection from disasters. Not all disasters, particularly natural disasters, can be prevented, but the risk of loss of life and injury can be mitigated with good evacuation plans, environmental planning and design standards. In January 2005, 168 Governments adopted a 10-year global plan for natural disaster risk reduction called the Hyogo

Framework. It offers guiding principles, priorities for action, and practical means for achieving disaster resilience for vulnerable communities.

- **Disaster preparedness:** These activities are designed to minimize loss of life and damage – for example by removing people and property from a threatened location and by facilitating timely and effective rescue, relief and rehabilitation. Preparedness is the main way of reducing the impact of disasters. Community-based preparedness and management should be a high priority in physical therapy practice management.

- **Disaster relief:** This is a coordinated multi-agency response to reduce the impact of a disaster and its long-term results. Relief activities include rescue, relocation, providing food and water, preventing disease and disability, repairing vital services such as telecommunications and transport, providing temporary shelter and emergency health care.

- **Disaster recovery:** Once emergency needs have been met and the initial crisis is over, the people affected and the communities that support them are still vulnerable. Recovery activities include rebuilding infrastructure, health care and rehabilitation. These should blend with development activities, such as building human resources for health and developing policies and practices to avoid similar situations in the future.

Associated Organisations And Their Functions

National Disaster Management Authority (India): National Disaster Management Authority, abbreviated as NDMA is an agency of the Ministry of Home Affairs whose primary purpose is to coordinate response to natural or man-made disasters and for capacity-building in disaster resiliency and crisis response. NDMA was established through the Disaster Management Act enacted by the Government of India on 30 May 2005. The Prime Minister is the ex-officio chairperson of the same. The agency is responsible for framing policies, laying down guidelines and best-practices and coordinating with the State Disaster Management Authorities (SDMAs) to ensure a holistic and distributed approach to disaster management. By a 9-member board chaired by the Prime Minister of India. The remainder of the board consists of members nominated based on their expertise in areas such as planning, infrastructure management, communications, meteorology and natural sciences. The day-to-day management of the agency is overseen by the office of the Vice Chair. Our app will provide information to NDMA regarding disaster which in turn will help the authorities to manage the disaster scenario in a better way.



NDMA is operationally organized into the following divisions:

- Policy & Planning
- Mitigation
- Operations & Communications
- Administration

Capacity Building:

NDMA equips and trains other Government officials, institutions and the community in mitigation and response during a crisis situation or a disaster. It operates the National Institute of Disaster Management, which develops practices, delivers hands-on training and organizes drills for disaster management. It also equips and trains disaster management cells at the state and local levels. NDMA also collaborates with the Lal Bahadur Shastri National Academy of Administration and Sardar Vallabhbhai Patel National Police Academy to impart training to administration and police officers in planning and incident response. It monitors and develops guidelines for the local Firefighting Services across the country. It collaborates with the Ministry of Health and Family Welfare in developing emergency health and ambulance services. Specifically, it focuses on capacity building in dealing with mass casualty at local hospitals.

Functions and responsibilities

NDMA, as the apex body, is mandated to lay down the policies, plans and guidelines for Disaster Management to ensure timely and effective response to disasters. Towards this, it has the following responsibilities:

- Lay down policies on disaster management;
- Approve the National Plan;
- Approve plans prepared by the Ministries or Departments of the Government of India in accordance with the National Plan;
- Lay down guidelines to be followed by the State Authorities in drawing up the State Plan;
- Lay down guidelines to be followed by the different Ministries or

Departments of the

- Government of India for the Purpose of integrating the measures for prevention of disaster or the mitigation of its effects in their development plans and projects;
- Coordinate the enforcement and implementation of the policy and plans for disaster management;
- Recommend provision of funds for the purpose of mitigation;
- Provide such support to other countries affected by major disasters as may be determined by the Central Government;
- Take such other measures for the prevention of disaster, or the mitigation, or preparedness and capacity building for dealing with threatening disaster situations.

How Technology Can Help In Disaster Management ?

The role of technology in disaster can be predictable in such a way that it minimizes the hazard and helps in reduction of vulnerability. Progress in the science and technology of natural hazards and related coping mechanisms has made it possible over the past years to introduce significant changes in the integrated approach to the problematic of natural disasters. Science and technology help us to understand the mechanism of natural hazards of atmospherically, geological, hydrological, and biological origins and to analyze the transformation of these hazards and disasters.

The wide spectrum of technologies used in all four phases of disaster management preparedness, mitigation, response and recovery are remote sensing, Geographical Information System, Global Positioning System (GPS), Satellite navigation system, Satellite communication, Amateur and community radio, television and radio broadcasting, Telephone and fax, Cellular phones, video Conferencing Networking Technologies, Internet, e-mail; On-line management databases, disaster information systems and

networks, Robotics . The application of all these technologies in disaster management are: Setting up disaster early warning systems, Quick processing and analysis of disaster systems. Applications also include Database construction.

Database Management Systems

Database is the most common way of storing and managing data. There are a number of database management applications for the purpose of building Android applications. It is essential to send out warning to all people in a vulnerable region. It can provide information on disaster professionals in a particular nation/region. To build our application we have a host of choices but we decided to shortlist a few of the more popular ones which would allow us to do our job relatively easily and with proper functionality. It can be used for setting up Disaster Information Network. To finalize our choice, we underwent a comparison among the following options tonsider necessary:

- Lay down broad policies and guidelines for the functioning of the National Institute of Disaster Management.

Technology Stack

ANDROID

Android is the most emerging mobile application development platform now-a-days. It is an open-source operating system based on Linux kernel. Android is an open-source OS, so it is free and open to use. Android covers a wide range of hardware made by different manufacturers.



DART

Dart is a client-optimized programming language for apps on multiple platforms. It is used to build mobile, desktop, backend and web applications. Dart is an object-oriented, class defined, garbage-collected language.



FLUTTER

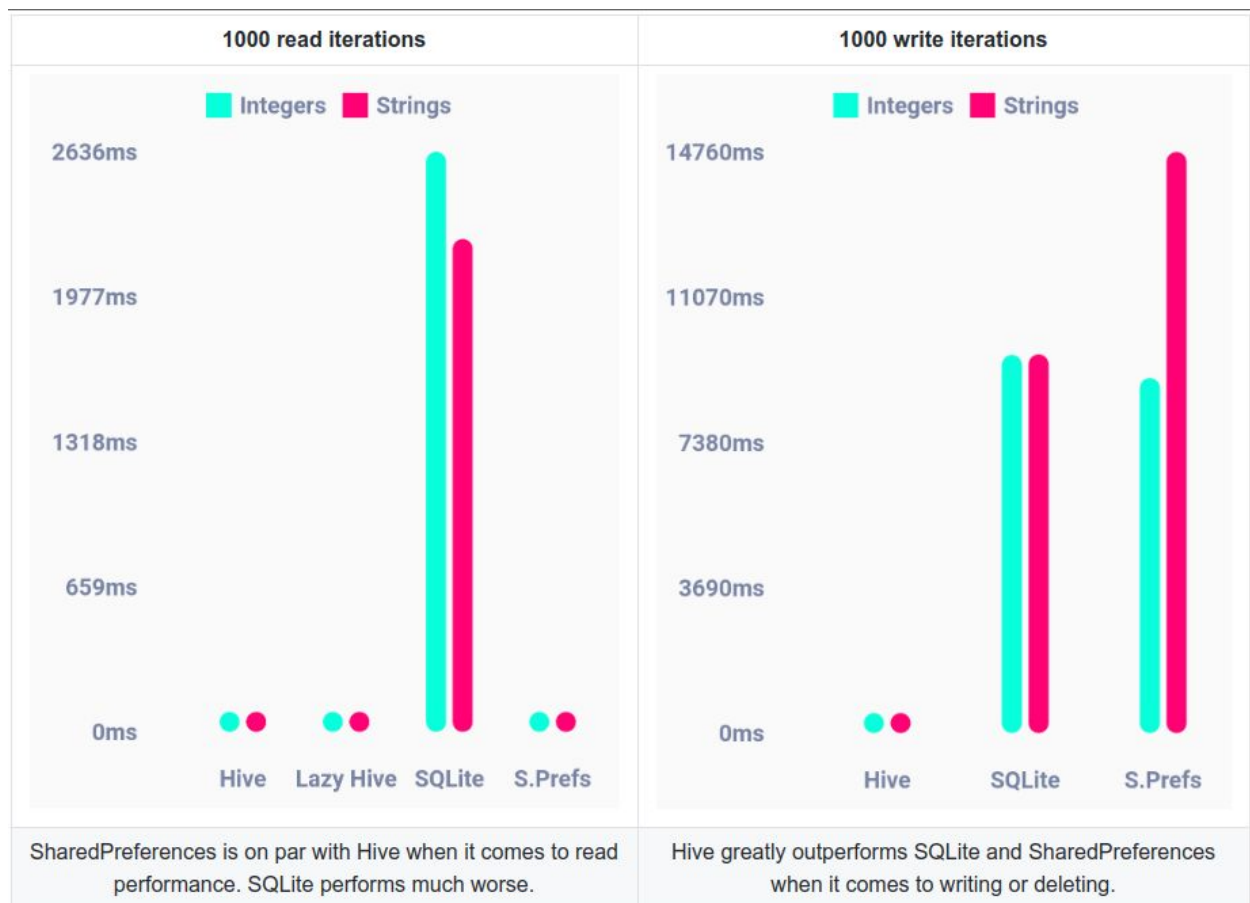
Flutter provides a great automated testing framework to test the app at unit, widget and integration level, and with CI/CD we can automate running the automated tests as part of continuous integration without any setup or configuration. It's worth considering Flutter is of great use when you need an application that works on both Android and iOS platforms.



Hive DB



Hive is a lightweight and blazing fast key-value database written in pure Dart. Hive serves our purpose of an offline database which caches the records and uploads it when the device restores internet connectivity.



Hive is designed to work better with dart than a native solution like SQLite and is more secure than shared preferences.

Why Flutter Uses Dart:

The early Flutter team evaluated more than a dozen languages, and picked Dart because it matched the way they were building user interfaces.

Dart is a big reason why developers love Flutter. Here is a quick list of the Dart features that together make it indispensable for Flutter:

- Dart is AOT (Ahead Of Time) compiled to fast, predictable, native code, which allows almost all of Flutter to be written in Dart. This not only makes Flutter fast, virtually everything (including all the widgets) can be customized.
- Dart can also be JIT (Just In Time) compiled for exceptionally fast development cycles and game-changing workflow (including Flutter's popular sub-second stateful hot reload).
- Dart makes it easier to create smooth animations and transitions that run at 60fps. Dart can do object allocation and garbage collection without locks. Dart avoids preemptive scheduling and shared memory (and thus locks). They also start up much faster.
- Dart allows Flutter to avoid the need for a separate declarative layout language like JSX or XML, or separate visual interface builders, because Dart's declarative, programmatic layout is easy to read and visualize. And with all the layout in one language and in one place, it is easy for Flutter to provide advanced tooling that makes layout a snap.
- Developers have found that Dart is particularly easy to learn because it has features that are familiar to users of both static and dynamic languages.
- Dart and Flutter both are developed by Google, which makes both of them tightly integrated with each other. Most of the API changes in Dart are done keeping flutter in mind and thus provides much better consistency between the two.

We use flutter because it gives us an easy cross platform solution for building mobile apps. Our app is targeted at the Android platform for now, but it can be migrated to iOS very easily, whenever we need.

NodeJS

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Simply said, Node.js lets you run server side javascript code. It has a large community and a robust package management service. It delivers high performance through its asynchronous non-blocking event-driven I/O model.



What is asynchronous non-blocking event-driven I/O ?

Okay, let's break it

Asynchronous - Synchronous execution usually refers to code executing in sequence. Asynchronous execution refers to execution that doesn't run in the sequence it appears in the code.

Non-blocking - Blocking refers to operations that block further execution until that operation finishes. Non-blocking refers to code that doesn't block execution. How does it help? So, when we are running a server we need to make sure that we are able to efficiently handle multiple requests at a time, and if it has not been asynchronous, then the server would pause execution until the previous request is finished.

Event-driven IO - Event-driven I/O refers to the fact that NodeJs makes use of Event Emitters and Event Listeners to finish tasks that need to be done when a particular operation happens. We can set our own custom event, have them emitted after some particular task and correspondingly implement an event listener which listens for that event and executes another piece of code for us.

ExpressJS

Express.js, described on its website as a "fast, unopinionated, minimalist server-side Web framework for Node.js". Written in JavaScript, Express acts only as a thin layer of core Web application features.



With a myriad of HTTP utility methods and middleware at our disposal, creating a robust API is quick and easy. It has **Routers** for managing routes, **Middlewares** for implementing specific functionalities to process the data, check for authentication any many other things.

We designed our API using Express. It follows the REST API principles and makes it easier to scale.

JWT

A JSON web token(JWT) is JSON Object which is used to securely transfer information over the web(between two parties). It can be used for an authentication system and can also be used for information exchange. The token is mainly composed of header, payload, signature. These three parts are separated by dots(.). JWT defines the structure of information we are sending from one party to another, and it comes in two forms – Serialized, Deserialized. The Serialized approach is mainly used to transfer the data through the network with each request and response. While the deserialized approach is used to read and write data to the web token. JWT in the deserialized form contains only the header and the payload. Both of them are plain JSON objects.



MongoDB

MongoDB is a cross-platform document oriented database program. Classified as a NoSQL database, MongoDB is a *document-based* database management system which leverages a JSON-style storage format known as binary JSON, or BSON, to achieve high throughput. BSON makes it easy for applications to extract and manipulate data, as well as allowing properties to be efficiently indexed, mapped, and nested in support of complex query operations and expressions.



So why did we use MongoDB? So leaving any other arguments apart, MongoDB just plays well with JSON. As for the server side we are using JavaScript, so the whole Node, Express and MongoDB architecture fits-in very well.

Mongoose (ODM)

MongooseJS is an Object Document Mapper (ODM) that makes using MongoDB easier by translating documents in a MongoDB database to objects in the program. Besides MongooseJS there are several other ODM's that have been developed for MongoDB including Doctrine and MongoLink.



Mongoose being the most widely used one has been added in our server.

API Server

Let's see the directory structure for the server:

We are using MVC patterned structure for the application. A typical Node and Express server setup is used in our Server Application. This piece of code sets up our server and makes it to listen on the environment specified port

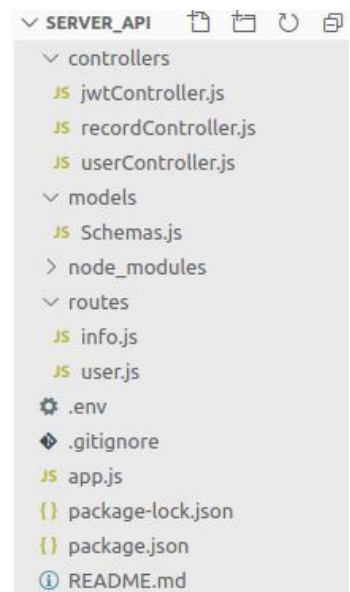
```
require('dotenv').config();
const express = require('express');
const mongoose = require('mongoose');
const app = express();

// Connect to MongoDB
mongoose.connect(process.env.MONGO_URI,
  {
    useNewUrlParser: true,
    useFindAndModify: false
  })
  .then(() => console.log('MongoDB Connected...'))
  .catch(err => console.log(err));

// Express.js Routes
const user = require("./routes/user");
const info = require("./routes/info");

app.use('/user', user);
app.use('/info', info);

//Server Init
const port = process.env.PORT || 4125;
const server = app.listen(port, () => {
  console.log(`Server started on port ${server.address().port}`);
});
```



Now we have the Express Router which manages the routes for the API

```
router.post('/signin', (req, res) => {  
  if (req.body.licence && req.body.psswd) {  
    userController.signIn(req.body.psswd, req.body.licence)  
      .then((msg) => {res.json(msg);})  
      .catch((err) => {res.status(401).json(err);})  
  } else  
    res.sendStatus(400);  
});
```

Here is our Agency Schema:

```
// Schema definition  
const AgencySchema = new Schema({  
  licence: {  
    type: String,  
    required: true,  
    unique: true  
  },  
  name: {  
    type: String,  
    required: true  
  },  
  psswd: {  
    type: String,  
    required: true  
  },  
  address: {  
    type: String,  
    required: true  
  },  
  contact: {  
    type: Number,  
    required: true  
  }  
});
```

It responds to a POST request with route `'/user/signin'` if the required credentials are present.

```
let signIn = (pass, licence) => {
  return new Promise((resolve, reject) => {
    Agency.findOne({ licence: licence })
      .then(async (result) => {
        if (result) {
          await checkPwd(pass, result.psswd);
          return result;
        }
        else
          reject({ err: "Licence number doesn't exist" })
      }).then((result) => {
        return jwtHelper.JWTgen(result);
      }).then((token) => {
        resolve({ auth_token: token });
      }).catch((msg) => {
        reject({ err: msg });
      })
  });
}
```

And then returns the user a new JSON with the JWT.

```
let JWTgen = (user) => {
  return new Promise((resolve, reject) => {
    jwt.sign({ licence: user.licence }, jwtKey, {
      algorithm: 'HS256',
      expiresIn: jwtExpiry
    }, (err, token) => {
      if (err)
        reject(err)
      else
        resolve(token);
    });
  });
}
```

Client Application

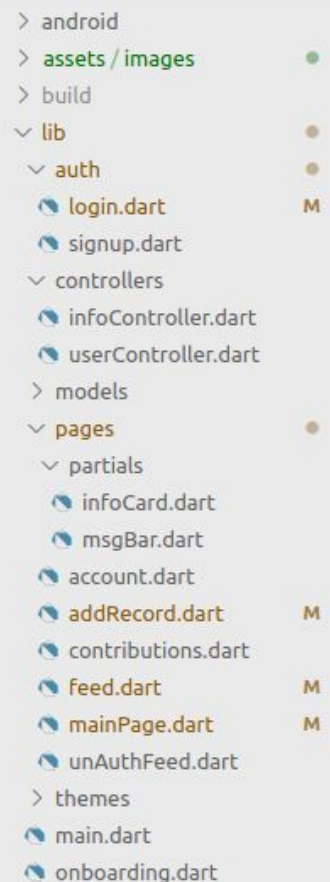
This is our directory structuring for the Android Client.

Here is the Main code that Loads up a MaterialApp Widget from flutter/material.dart . All the routes are defined here.

```
void main() => runApp(Landing());

class Landing extends StatefulWidget {
  @override
  _LandingState createState() => _LandingState();
}

class _LandingState extends State<Landing> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        primarySwatch: Colors.green,
        buttonTheme: ButtonThemeData(
          minWidth: 140.0,
          buttonColor: Colors.greenAccent,
        ),
      ),
      routes: {
        '/': (context) => OnBoardingActivity(),
        '/login': (context) => Login(),
        '/signup': (context) => SignUp(),
        '/home': (context) => PageWrapper(),
        '/noauth': (context) => UnAuthFeed(),
        '/add': (context) => AddRecord(),
      },
    );
  }
}
```



```
> android
> assets / images
> build
✓ lib
  ✓ auth
    login.dart M
    signup.dart
  ✓ controllers
    infoController.dart
    userController.dart
  > models
  ✓ pages
    ✓ partials
      infoCard.dart
      msgBar.dart
      account.dart
      addRecord.dart M
      contributions.dart
      feed.dart M
      mainPage.dart M
      unAuthFeed.dart
  > themes
    main.dart
    onboarding.dart
```

To get the login functionality from the Client side, we first take the Login form data and serialize it to JSON and send a POST request , upon successful login we are returned with a JWT which is stored in the Android KeyStore.

```
import 'package:disaster_reporting/models/user.dart';
import 'dart:convert';
import 'package:http/http.dart';

String url = 'https://srbk-mini-project.herokuapp.com/user/';

Future<Response> reqPost(route, headers, json) async {
  try {
    return await post(route, headers: headers, body: json);
  } catch (e) {
    return null;
  }
}

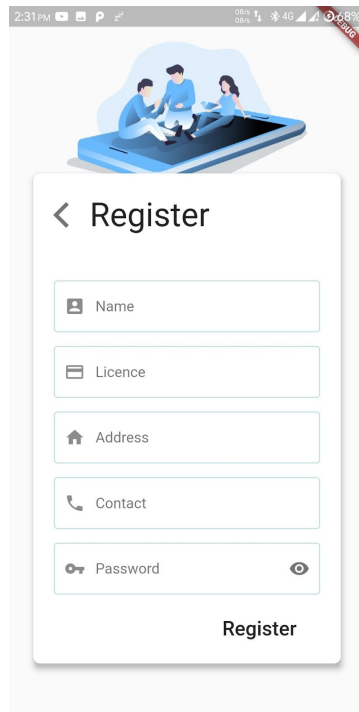
Future<Map<String, dynamic>> userSignin(Agency user) async {
  String route = url + 'signin';
  Map<String, String> headers = {"Content-type": "application/json"};
  Map<String, dynamic> userMap = {
    'licence': user.licence,
    'psswd': user.psswd,
  };
  String json = jsonEncode(userMap);
  Response response = await reqPost(route, headers, json);
  Map<String, dynamic> result;
  if (response == null) {
    result = {"err": "No internet connection"};
  } else if (response.statusCode == 200) {
    result = jsonDecode(response.body);
  } else {
    result = {"err": "Invalid username/password"};
  }
  return result;
}
```

For the Agency part we have a similar looking schema

```
class Agency {  
    String licence;  
    String name;  
    String psswd;  
    String old;  
    String address;  
    int contact;  
    String token;  
  
    Agency({  
        this.licence,  
        this.name,  
        this.psswd,  
        this.address,  
        this.contact,  
    });  
}
```


Routes and Pages

Register



2:31 PM

Register

← Register

Name

Licence

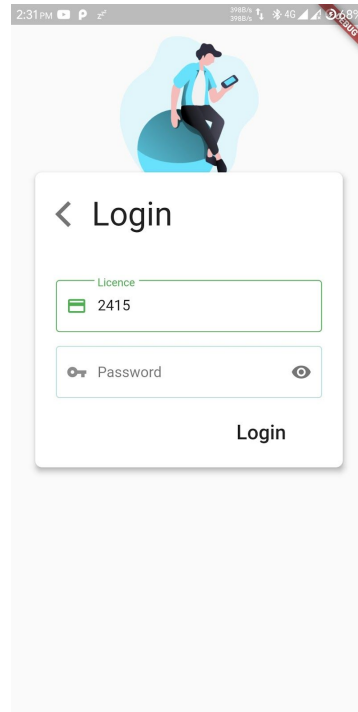
Address

Contact

Password

Register

Login



2:31 PM

Login

← Login

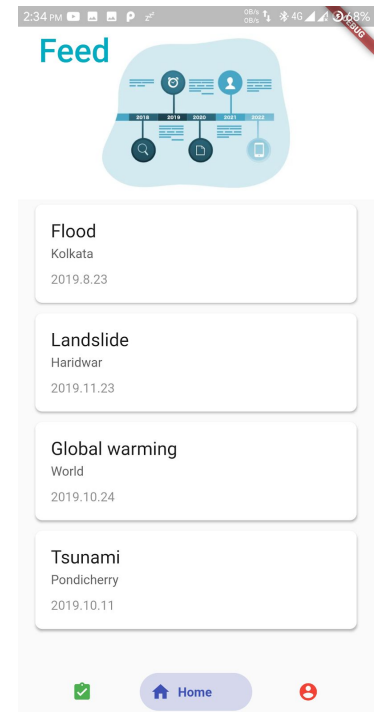
Licence

2415

Password

Login

User Feed



2:34 PM

Feed

Feed

Kolkata

2019.8.23

Landslide

Haridwar

2019.11.23

Global warming

World

2019.10.24

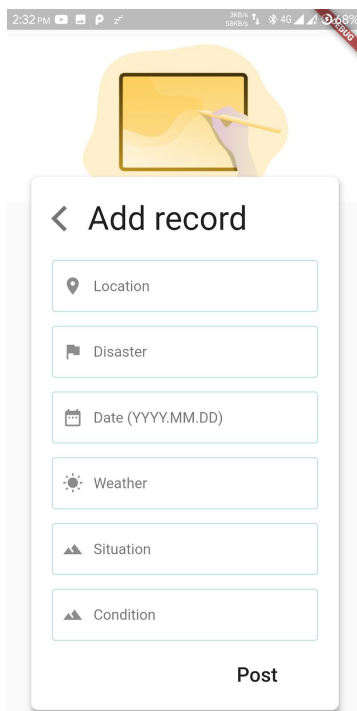
Tsunami

Pondicherry

2019.10.11

Home

Add Record



2:32 PM

Add record

← Add record

Location

Disaster

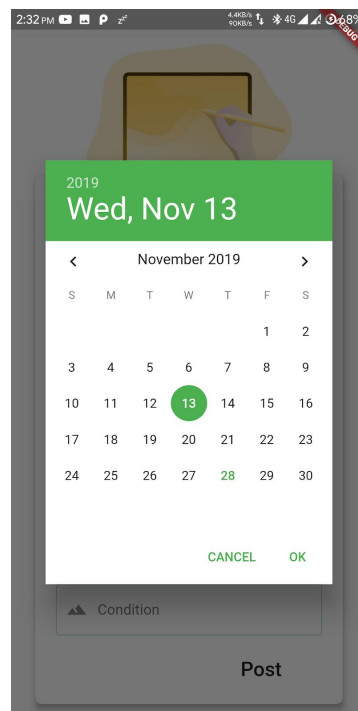
Date (YYYY.MM.DD)

Weather

Situation

Condition

Post



2:32 PM

2019

Wed, Nov 13

November 2019

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

CANCEL OK

Condition

Post

Deployment

Heroku

Heroku is a cloud application deployment platform, it makes it easier to host NodeJs applications for free with plenty of computational resources.

MLab

MLab is a fully managed MongoDB DataBase-as-a-Service platform. We have our Mongo instance hosted there.

Github Releases

All the latest builds of our android application are automatically deployed on Github Releases. We have Travis CI integrated in our project so whenever a new git tag is pushed it will automatically build and deploy the .apk files.

API Server Source: <https://github.com/sarbikbetal/mini-project-api>

Android app Source: <https://github.com/sarbikbetal/flutter-disaster-reporting>

Android app builds: <https://github.com/sarbikbetal/flutter-disaster-reporting/releases>

API Endpoint base URL: <https://srbk-mini-project.herokuapp.com/>

Conclusion

Thus, we plan to develop an application which will aid the disaster relief and recovery operations by maintaining a database of the victims in the various locations of a disaster affected region. This database can be handled by any of the associated volunteer groups involved in the relief operations and can be utilized effectively by the government bodies to handle the situation effectively with need-specific support to the sufferers within appropriate time. Our attempt at creating an app to aid the volunteers in the relief process of disaster management is one of a kind.

We have done this project for the first time so instead of going into detailed analysis, we have done it at surface level. We have tried to make it production oriented so that we can implement more and more tables.

Our app has several shortcomings:

- We have not added detailed information.
- Multiple records cannot be added by different users for the same calamities.

Future scopes

Our app has several shortcomings as we have done this project for the very first time. In future we wish to correct the aforementioned drawbacks of our app and we want to add more features to our app such as:

- We will try to increase the verbosity of the details. This is just a demo application. In our future, we will try to add more and more detailed information.
- We will try to make sure that multiple records can be added by different users for same calamities.
- We will add a feature to upload and share disaster photos to help first responders.
- Receiving alerts and notifications from the national weather service.
- Getting safety reminders (tips to survive natural disasters).
- Locate open shelters, healthcare facilities and evacuation camps in cases of severe strike of disasters
- Help about how and where to contact various volunteers of disaster management relief.
- Offline sharing of data using WIFI Direct.