



MENTAL HEALTH TRACKER AND ANALYZER

A MINI PROJECT REPORT

Submitted by in

R.KEERTHI VASAN (211420205305)

M.VIGNESH (211420205310)

V.CHANDRAN (211420205302)

partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE,

(An Autonomous Institution , Affiliated To Anna University)

POONAMALLEE

ANNA UNIVERSITY

CHENNAI 600 025

MAY 2023

BONAFIDE CERTIFICATE

Certified that this project report **“MENTAL HEALTH TRACKER AND ANALYZER “** is the bonafide work of **“ R . KEERTHI VASAN (211420205305) , M . VIGNESH (211420205310) , V . CHANDRAN (211420205302) ”** who carried out the project under my supervision.

SIGNATURE

**Dr. M. HELDA MERCY M.E., Ph.D.,
HEAD OF THE DEPARTMENT**

Department of Information Technology
Panimalar Engineering College
Poonamallee,
Chennai - 600 123

SIGNATURE

**DR.D.KARUNKUZHALI, M.Tech, Ph.D
PROFESSOR (SUPERVISOR)**

Department of Information Technology
Panimalar Engineering College
Poonamallee,
Chennai - 600 123

Submitted for the project and viva-voce examination held on _____

EXTERNAL EXAMINER

INTERNAL EXAMINER

DECLARATION

I hereby declare that the project report entitled “**MENTAL HEALTH TRACKER AND ANALYZER** ” which is being submitted in partial fulfilment of the requirement of the course leading to the award of the ‘Bachelor of Technology in Information Technology’ in **Panimalar Engineering College, An Autonomous institution Affiliated to Anna University- Chennai** is the result of the project carried out by us under the guidance and supervision of **DR.D.KARUNKUZHALI, M.Tech, Ph.D, Professor in the Department of Information Technology**. I further declared that I or any other person has not previously submitted this project report to any other institution/university for any other degree/ diploma or any other person.

(R.KEERTHI VASAN)

(M.VIGNESH)

(V . CHANDRAN)

Date:

Place: Chennai

It is certified that this project has been prepared and submitted under my guidance.

Date:

Place: Chennai

DR.D.KARUNKUZHALI, M.Tech, Ph.D

(Professor/ IT)

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co- operation and support from many, for successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to **Our Honorable Secretary and Correspondent, Dr .P. CHI NNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution.

We would like to express our deep gratitude to **Our Dynamic Directors, Mrs. C. VIJAYA RAJESHWARI and Dr .C.SAKTHI KUMAR, M.E ., Ph. D and Dr. SARANYA SREE SAKTHIKUMAR., B.E., M. B.A .,Ph .D** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, **Dr. M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our staff in charge, **DR.D.KARUNKUZHALI, M. Tech, Ph .D** Professor, Department of Information Technology for her guidance throughout the course of our project. Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	1
1	INTRODUCTION	2
	1.1. OVERVIEW OF THE PROJECT	3
	1.2 NEED FOR THE PROJECT	4
	1.3 OBJECTIVE OF THE PROJECT	6
	1.4 SCOPE OF THE PROJECT	7
2	LITERATURE SURVEY	8
3	SYSTEM DESIGN	10
	3.1 PROPOSED SYSTEM ARCHITECTURE DESIGN	10
	3.2 DATA FLOW DIAGRAM FOR PROPOSED SYSTEM	11
	3.3 MODULE DESIGN	12
4	REQUIREMENT SPECIFICATION	13
	4.1 HARDWARE REQUIREMENTS	13
	4.2 SOFTWARE REQUIREMENTS	14
5	IMPLEMENTATION	15
	5.1 SAMPLE CODE	15
	5.2 SAMPLE SCREEN SHOTS	50
6	CONCLUSION AND FUTURE ENHANCEMENTS	57
	REFERENCES	59

List of Figures

Fig .No.	Figure Name	Page No.
1.	Proposed System Architecture Design	10
2.	Data Flow Diagram For Proposed System	11
3.	Module Design	12
4.	Home page	50
5.	Questionnaire page	51
6.	Notification page	52
7.	Resources page	53
8.	Journal page	54
9.	Graph/Tracker page	55
10.	Profile page	56

ABSTRACT

Mental health is a crucial aspect of overall well-being, and the importance of maintaining good mental health cannot be overstated. Despite advancements in mental health research and treatment, many individuals with mental health disorders face significant challenges in accessing quality care, resulting in a considerable burden on healthcare systems and a loss of productivity and quality of life for those affected. The COVID-19 pandemic has exacerbated these challenges, highlighting the urgent need for accessible and effective mental health resources.

To address this need, a new platform has been developed that allows users to track their mood and symptoms, create trackers, and receive reminders. The platform also emphasizes the benefits of journaling, which can help users monitor their mental health throughout the day and provide resources for coping. By providing easy-to-use tools and resources, this platform aims to promote better mental health and support those struggling with mental health challenges.

INTRODUCTION:

Mood Tracker App is a powerful tool that can help people manage symptoms, improve mental health, and work towards recovery. The app allows users to monitor various metrics such as mood, stress levels, sleep patterns, medication adherence, and physical activity. Users can also write their thoughts and feelings in a notebook, which can help them, analyze the problem and come up with a solution.

This app uses data analytics to provide personalized insights and recommendations based on user input. For example, if an app detects that a user is in low mood during the day, it might suggest a break or vacation for that time.

One of the most important benefits of the application is that it allows users to manage their mental health. By monitoring their symptoms and understanding the consequences, users can better understand their condition and take steps to manage it. The app also helps users better communicate with their doctor as they can share detailed information about their symptoms and progress.

In a nutshell, the PTSD & Depression Mood Tracker App is a useful tool that can help people manage their mental health and work towards recovery. By monitoring symptoms and providing personalized insights, the app allows users to manage their condition and improve their overall health.

1.1. OVERVIEW OF THE PROJECT

The Mood Tracker App is a mobile application that aims to help individuals manage their mental health by monitoring their symptoms and providing personalized insights and recommendations. The app enables users to track various metrics such as mood, stress levels, sleep patterns, medication adherence, and physical activity. It also provides users with a notebook feature that allows them to write their thoughts and feelings, which can help them analyze the problem and come up with a solution.

The app uses data analytics to provide personalized recommendations based on user input. For instance, if the app detects that a user is in a low mood during the day, it might suggest taking a break or vacation during that time. The application can also help users better communicate with their doctors by providing detailed information about their symptoms and progress.

One of the most significant benefits of the app is that it allows individuals to manage their mental health by understanding their condition and taking steps to manage it. By monitoring their symptoms, users can better understand their mental health and take proactive steps towards recovery.

Overall, the App is a powerful tool that can help individuals manage their mental health effectively. The app's features, including tracking metrics, personalized insights, and notebook, make it a versatile and comprehensive mental health management tool.

1.2 NEED FOR THE PROJECT

Mental health is a crucial aspect of overall wellbeing, and mental health disorders such as PTSD and depression can significantly impact an individual's quality of life. The COVID-19 pandemic has highlighted the need for accessible and effective mental health management tools, as mental health concerns have become more prevalent. According to the World Health Organization, around one in four people globally will experience mental health disorders in their lifetime.

Unfortunately, many individuals do not receive the support and treatment they need due to barriers such as stigma, lack of access to healthcare, and inadequate resources. The Mood Tracker App is designed to address the such all edges by providing individuals with a tool to manage their mental health more effectively. The app offers a range of features that allow users to track their symptoms and progress, which can help them understand their mental health better. By identifying patterns and triggers, users can take proactive steps towards managing their mental health and seek appropriate treatment if necessary

In addition to the challenges posed by stigma and lack of access to care, many individuals with mental health disorders also struggle with self-stigma and shame. They may feel embarrassed or ashamed about their symptoms, which can prevent them from seeking help or disclosing their condition to others. The Mood Tracker App can help to address this issue by providing a safe and confidential space for individuals to track their symptoms and reflect on their mental health.

The app can also help to promote self-care and wellness by encouraging users to prioritize their mental health needs. The tracking metrics, such as sleep patterns and physical activity, can help users identify areas where they may need to make changes to improve their overall well being. The app's personalized recommendations can offer suggestions on how to address these areas and improve their mental health.

Moreover, the app can also help individuals communicate more effectively with mental health professionals. By providing detailed information about their symptoms and progress, users can share valuable insights with their care providers, allowing them to provide more effective treatment. This

can be particularly beneficial for individuals who may have difficulty communicating their symptoms or who may not have regular access to mental health care.

Importantly, the app is designed to be user-friendly and accessible to a wide range of individuals. It can be used by anyone with a smart phone or tablet, regardless of their location or access to traditional mental health services. This can be particularly beneficial for individuals who may live in remote or rural areas where mental health services are limited.

The Mood Tracker App can also help to reduce the burden on mental health professionals by providing individuals with a tool to manage their symptoms independently. While the app is not intended to replace professional mental health care, it can help to supplement it by providing users with a way to manage their symptoms and track their progress between appointments.

In conclusion, Mood Tracker App is a valuable tool for individuals with mental health disorders, particularly in the current climate where mental health concerns are more prevalent than ever. By providing a safe and confidential space for individuals to track their symptoms, personalized recommendations for managing mental health, and a means of communicating more effectively with care providers, the app can help to promote self-care, reduce stigma, and improve access to mental health care.

1.3 OBJECTIVE OF THE PROJECT

The project's goal is to promote mental health awareness and reduce the stigma attached to mental illness. The objective is to provide access to high-quality mental health care services to individuals in need, particularly those from underserved communities. The collaboration between mental health organizations, healthcare providers, government agencies, and community organizations aims to improve the delivery of mental health services.

Mental health disorders affect millions of people worldwide, and the lack of awareness and stigma attached to it only makes matters worse. People often hesitate to seek help due to the fear of being judged, ridiculed, or ostracized by society. By increasing awareness and understanding of mental health disorders and reducing the stigma, individuals will be encouraged to seek help and support.

Furthermore, improving access to quality mental health care services is essential in ensuring that individuals receive the appropriate treatment for their condition. This includes timely diagnosis, effective treatment, and ongoing support. In addition, the project aims to ensure that mental health services are available to individuals from all walks of life, regardless of their socioeconomic status.

Collaboration between mental health organizations, healthcare providers, government agencies, and community organizations is crucial in achieving these objectives. Working together will help to create a unified approach to mental health care services, ensuring that individuals receive the best possible care and support.

Overall, the project's aim is to improve the mental well-being and quality of life of individuals affected by mental health disorders. By increasing awareness, reducing stigma, improving access to care, and collaborating between organizations, we can create a healthier and more supportive environment for those in need.

1.4 SCOPE OF THE PROJECT

The scope of the Mood Tracker App project involves developing and implementing a mobile application that helps individuals manage their mental health by tracking various metrics such as mood, stress levels, sleep patterns, medication adherence, and physical activity. The app uses data analytics to provide personalized insights and recommendations based on user input. The application also allows users to write their thoughts and feelings in a notebook, enabling them to analyze their problems and come up with solutions. Additionally, the app enables users to communicate with their healthcare providers by sharing detailed information about their symptoms and progress. The project's main goal is to provide users with a powerful tool that can help manage symptoms, improve mental health, and work towards recovery.

LITERATURE SURVEY:

Mental health is an essential aspect of overall well-being. Conditions such as depression and PTSD are widespread worldwide, and their impact can be significant, affecting various aspects of an individual's life. In recent years, there has been a growing interest in the use of technology-based interventions, such as mood tracking apps, to manage mental health conditions. These interventions provide a new and innovative way to manage these conditions and improve mental health outcomes.

Numerous studies have shown that technology-based interventions, including mobile apps, can be effective in reducing symptoms of depression and anxiety (Firth et al., 2017). A systematic review and meta-analysis of 17 randomized controlled trials (RCTs) found that these interventions provide a cost-effective way to improve mental health outcomes. The review highlights the importance of technology-based interventions in the management of mental health conditions, particularly in improving access to mental health care.

One study examined the effectiveness of a mobile app-based intervention for individuals with depression and found that it was associated with significant reductions in depressive symptoms (Lyet et al., 2014). The study concluded that mobile apps could provide a useful adjunct to traditional forms of therapy and improve access to mental health care. These findings suggest that mobile apps can be a useful tool in managing mental health conditions, such as depression.

The use of mood tracking apps has also been found to improve medication adherence among individuals with mental health conditions. A study of individuals with bipolar disorder found that the use of a mood tracking app was associated with better medication adherence and fewer hospitalizations (Bopp et al., 2010). These findings suggest that the use of technology-based interventions, such as mood tracking apps, can help individuals with mental health conditions adhere to their treatment plans and manage their symptoms more effectively.

The PTSD & Depression Mood Tracker App is an innovative tool that aims to help individuals manage their mental health by monitoring their symptoms and providing personalized insights. The app allows individuals to track their mood, sleep, and activity levels, among other things, and provides personalized recommendations based on their input. The app's personalized insights and

recommendations can help individuals manage their symptoms more effectively and improve their mental health outcomes.

For instance, if an individual's mood is consistently low, the app may recommend they engage in activities that are known to improve mood, such as exercise or spending time with friends and family. These personalized recommendations can help individuals manage their symptoms more effectively and improve their overall well-being. The app also provides a platform for individuals to communicate with their healthcare providers and share their data, leading to better overall health outcomes.

The use of technology-based interventions, such as mood tracking apps, can improve mental health outcomes and provide a cost-effective way to manage mental health conditions. These interventions can help individuals adhere to their treatment plans, manage their symptoms more effectively, and improve communication with their healthcare providers. The PTSD & Depression Mood Tracker App is a promising tool that can help individuals manage their mental health by providing personalized insights and recommendations based on their input. The app's ability to improve communication between individuals and their healthcare providers can also lead to better overall health outcomes.

In conclusion, the use of technology-based interventions, such as mood tracking apps, is a promising approach in managing mental health conditions. The PTSD & Depression Mood Tracker App is an innovative tool that provides personalized insights and recommendations to help individuals manage their symptoms more effectively. The app's ability to improve communication between individuals and their healthcare providers can also lead to better overall health outcomes. With further research and development, technology-based interventions have the potential to revolutionize the way mental health conditions are managed and improve mental health outcomes worldwide.

SYSTEM DESIGN:

3.1 PROPOSED SYSTEM ARCHITECTURE DESIGN Fig

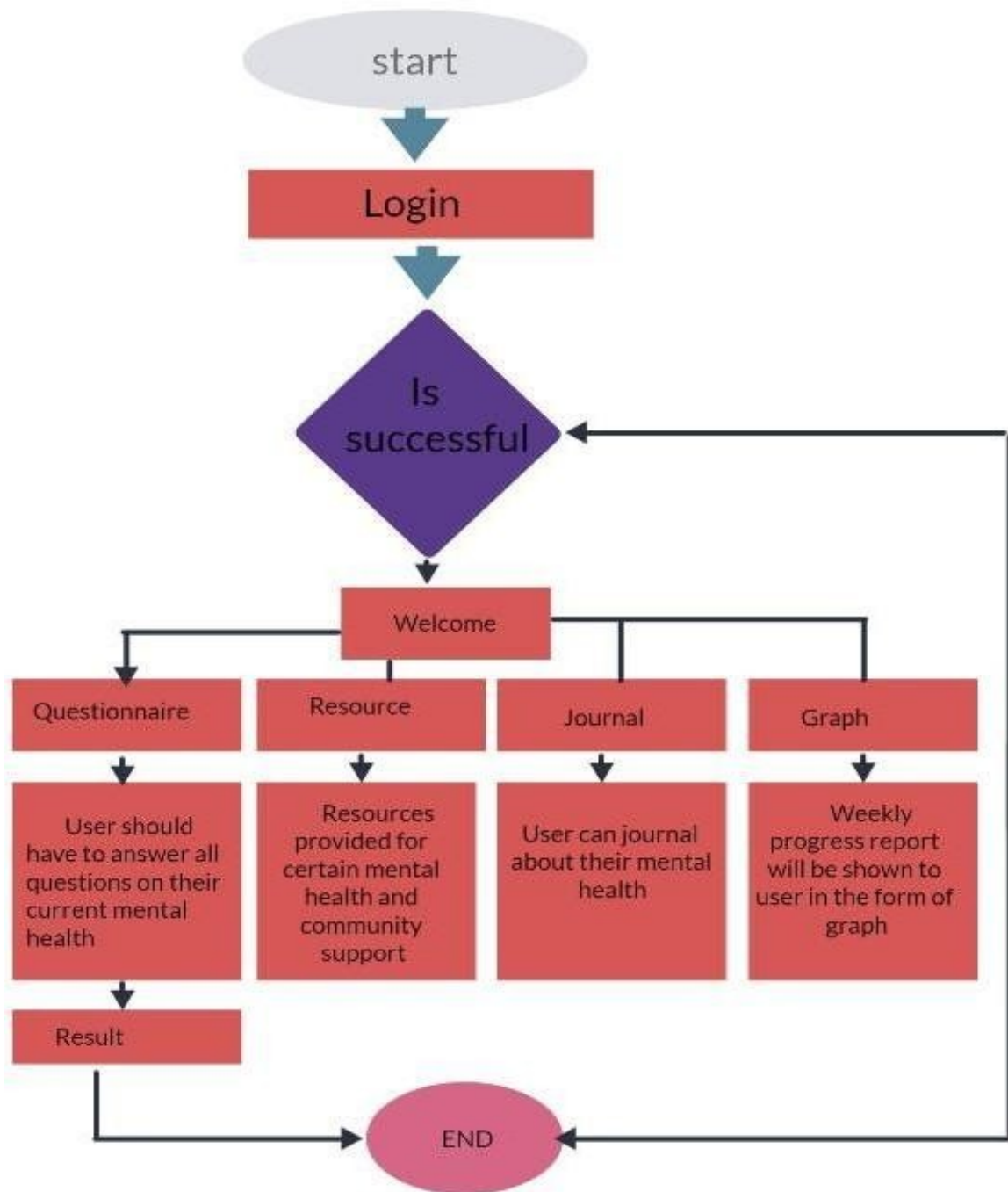


Fig 1.0

3.2 DATA FLOW DIAGRAM FOR PROPOSED SYSTEM:

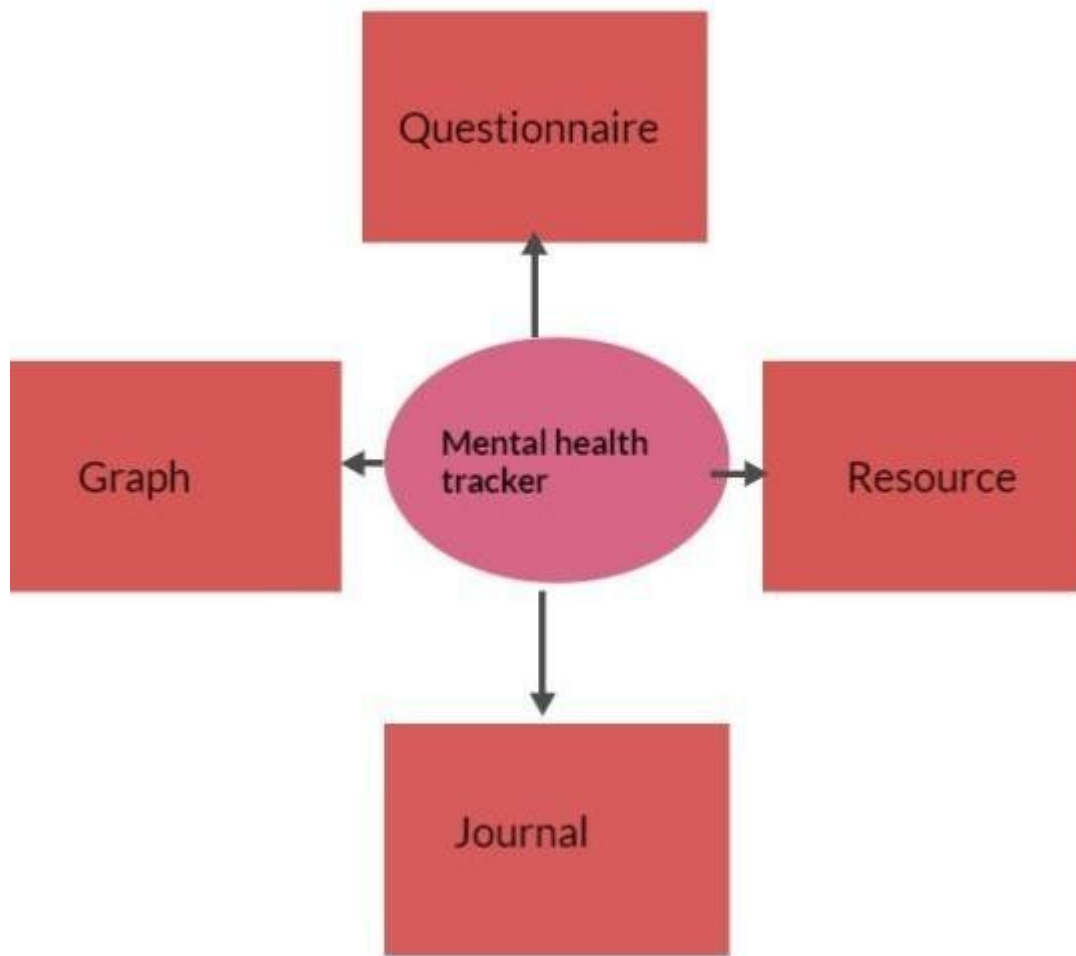


Fig 1.1

3.3 MODULE DESIGN

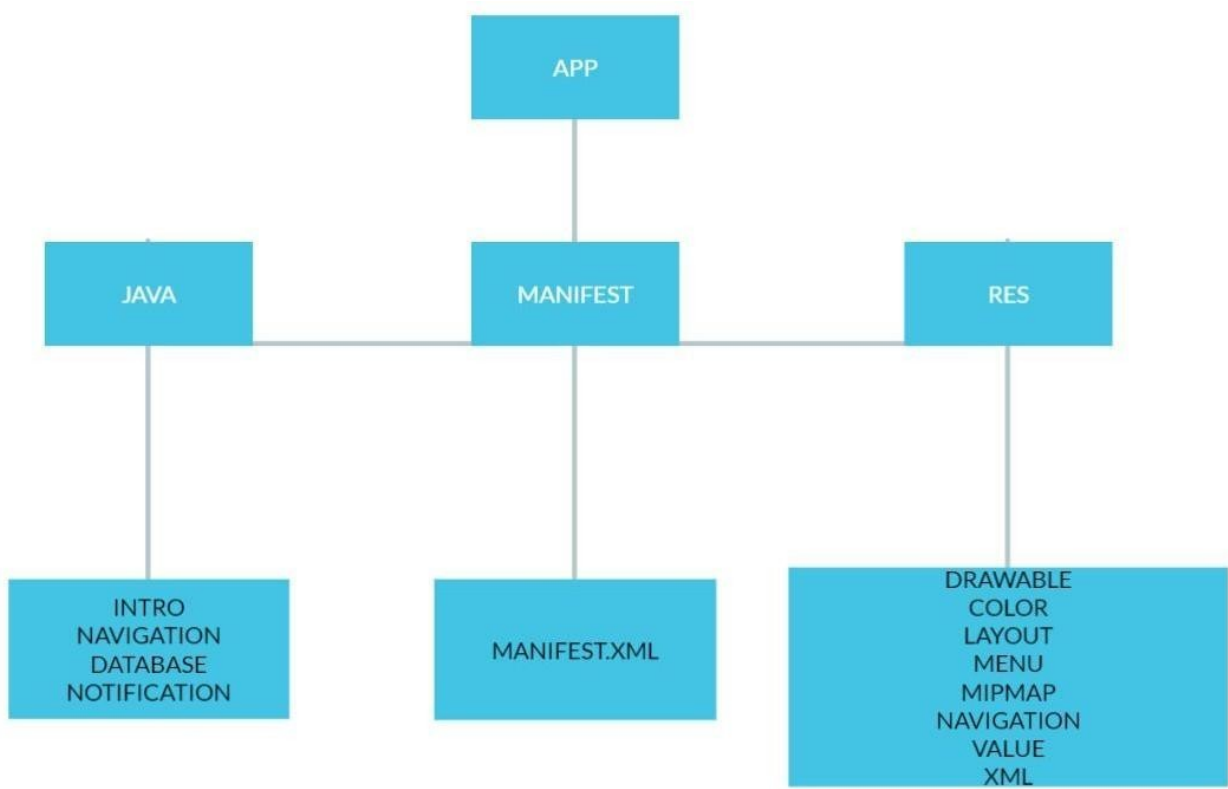


Fig 1.2

REQUIREMENT SPECIFICATION

4.1 HARDWARE REQUIREMENTS

Processor: Android applications require a minimum of a 1 GHz processor.

RAM: Android application require a minimum of 2 GB of RAM. However,

Storage: Android application require a minimum of 16 GB of storage space.

Display: Android applications require a display with a minimum resolution of 480x800 pixels.

SOFTWARE REQUIREMENTS :

Operating System: Application need the operating system android with latest version.

Integrated Development Environment (IDE): Android Studio is the used IDE for this Android application.

App Permissions: Enable the notification in the settings.

IMPLEMENTATION

5.1 SAMPLE CODE

```
package com.texasstech.talk.database;

import android.content.Context;

import androidx.room.Database;
import androidx.room.Room;
import androidx.room.RoomDatabase;

@Database(entities = {Mood.class, Resources.class, Journal.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {
    /**
     * The app's "single source of truth" is this database used
     * across all the activities and fragments. This database
     * provides Daos to all the different tables used by the
     * app with both read/write abilities.
     */
    private static AppDatabase mSingleInstance;
    public static final String DATABASE_NAME = "database";

    public abstract MoodDao moodDao();
    public abstract ResourcesDao resourcesDao();
    public abstract JournalDao journalDao();

    public static AppDatabase getDatabase(final Context context) {
        /**
         * Returns the single instance of the database that lives
         * across the lifetime of the application. AppDatabase
         * objects are expensive so only one instance should exist.
         *
         * TODO: Disable the allowing on the main thread and create
         * a separate class for asynchronous operations.
         */
        if (mSingleInstance == null)
            synchronized (AppDatabase.class)
            {
                if (mSingleInstance == null)
                {
                    mSingleInstance =
                        Room.databaseBuilder(
                            context.getApplicationContext(), AppDatabase.class, DATABASE_NAME)
                            .allowMainThreadQueries()
                            .build();
                }
            }
        return mSingleInstance;
    }
}
```

```

    }
    }
}

return mSingleInstance;
}
}

package com.texastech.talk.database;

import androidx.room.ColumnInfo;
import androidx.room.Entity;
import androidx.room.PrimaryKey;

@Entity
public class Journal
{ @PrimaryKey(autoGenerate =
true)int jid;

@ColumnInfo(name = "title")
public String title;

@ColumnInfo(name = "body")
public String body;

public Journal(String title, String body)
{this.title = title;
this.body = body;
}
}

```

```

@Entity
public class Mood {
/**
 * This entity is an abstraction of the lower-level table
 * used to store the moods that a user goes through every day.
 * Each mood is stored as a value between 1-6.
 *
 * The lower-level table looks like the following:
 *
 * -----
 * | ID | Date | Value |
 * -----
 * | 0 | 2131 | 2   |
 * | 1 | 2339 | 4   |
 * | .. | ... | ... |

```

```

* -----
*/
@PrimaryKey(autoGenerate = true)
int mid;

@ColumnInfo(name = "date")
public int date;

@ColumnInfo(name = "value")
public int value;

@ColumnInfo (name="severity_level")
public int severityLevel;

public Mood(int date, int value, int severityLevel)
{
    this.date = date;
    this.value = value;
    this.severityLevel= severityLevel;
}
}

package com.texastech.talk.database;

import androidx.room.ColumnInfo;
import androidx.room.Entity;
import androidx.room.PrimaryKey;

@Entity
public class Resources {
/**
 * Entity used to store the resources that are suggested to the user
 * based on their past moods. The suggested resources are displayed
 * as CardView objects with a "LEARN MORE" button.
 *
 * The lower-level table looks like the following:
 *
 * -----
 * | rid | title | content | hyperlink | mood |
 * |-----|
 * | 0   | depr | this is | https:// | 1   |
 * | ... | ...  | ...    | ...      | ... |
 * |-----|
 */
@PrimaryKey(autoGenerate = true)
int rid;

```

```

@ColumnInfo(name = "title")
public String title;

@ColumnInfo(name = "content")
public String content;

@ColumnInfo(name = "hyperlink")
public String hyperlink;

@ColumnInfo(name = "mood")
public int mood;

public Resources(String title, String content, String hyperlink, int mood)
{
    this.title = title;
    this.content = content;
    this.hyperlink = hyperlink;
    this.mood = mood;
}
}

package com.texastech.talk.intro;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;

import androidx.fragment.app.Fragment;
import androidx.preference.PreferenceManager;

import com.github.paolorotolo.appintro.AppIntro2;
import com.texastech.talk.MainActivity;
import com.texastech.talk.R;

public class IntroActivity extends AppIntro2 {
    /**
     * Activity launched the first time the app is installed in order to
     * introduce the user to the application purpose/features. This activity
     * contains a series of slides that are implemented in layouts and asks
     * for permissions required to use certain features of the app.
     */
    public static final String LAUNCHED_APP_BEFORE = "IntroActivity.LaunchedAppBefore";

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
    }
}

```



```

        addSlide(SlideHostFragment.newInstance(R.layout.layout_intro_slide1));
        addSlide(SlideHostFragment.newInstance(R.layout.layout_intro_slide2));
        addSlide(SlideHostFragment.newInstance(R.layout.layout_intro_slide3));
        addSlide(SlideHostFragment.newInstance(R.layout.layout_intro_slide4));
        addSlide(SlideHostFragment.newInstance(R.layout.layout_intro_slide6));

        showSkipButton(false);
        showStatusBar(false);
    }

    @Override
    public void onDonePressed(Fragment currentFragment) {
        /**
         * Executed when the user clicks the button on the last slide
         * acknowledging that they have read the terms of use and
         * would like to exit the app introduction.
         */
        super.onDonePressed(currentFragment);

        setLaunchedAppBefore();

        Intent intent = new Intent(this, MainActivity.class);
        intent.putExtra(MainActivity.QUERY_MOOD_PARAMETER, true);
        finish();
        startActivity(intent);
    }

    void setLaunchedAppBefore() {
        /**
         * Marks the user as having been introduced to the app. Therefore,
         * they will not have to go through the introduction in the future.
         */
        SharedPreferences.Editor editor = PreferenceManager.getDefaultSharedPreferences(this).edit();
        editor.putBoolean(LAUNCHED_APP_BEFORE, true);
        editor.apply();
    }
}

package com.texastech.talk.intro;

import android.os.Bundle;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;

```

```

import android.view.ViewGroup;

public class SlideHostFragment extends Fragment {

    private static final String ARG_LAYOUT_RES_ID = "layoutResId";
    private int layoutResId;

    public static SlideHostFragment newInstance(int layoutResId)
    {SlideHostFragment sampleSlide = new
    SlideHostFragment();

    Bundle args = new Bundle();
    args.putInt(ARG_LAYOUT_RES_ID, layoutResId);
    sampleSlide.setArguments(args);

    return sampleSlide;
}

@Override
public void onCreate(@Nullable Bundle savedInstanceState)
{super.onCreate(savedInstanceState);

    if (getArguments() != null && getArguments().containsKey(ARG_LAYOUT_RES_ID))
    {layoutResId = getArguments().getInt(ARG_LAYOUT_RES_ID);
    }
}

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
    @Nullable Bundle savedInstanceState) {
    return inflater.inflate(layoutResId, container, false);
}
}

package com.texastech.talk.navigation;

import android.content.Intent;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;

```

```

import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView;
import android.widget.ListView;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.texastech.talk.NotepadEntry;
import com.texastech.talk.R;
import com.texastech.talk.database.AppDatabase;
import com.texastech.talk.database.Journal;
import com.texastech.talk.database.JournalDao;

import java.util.ArrayList;
import java.util.List;

public class JournalFragment extends Fragment
{private ArrayAdapter<String> mAdapter;
private ArrayList<String> mArrayList;
private AppDatabase mDatabase;

public JournalFragment() {
    // Required.
}

public static JournalFragment newInstance()
{return new JournalFragment();
}

@Override
public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data)
{super.onActivityResult(requestCode, resultCode, data);
JournalDao journalDao = mDatabase.journalDao();
List<Journal> allJournals = journalDao.getAll();
mArrayList.clear();
for (Journal journal : allJournals)
    {mArrayList.add(journal.title);
}
mAdapter.notifyDataSetChanged();
}

@Override
public void onCreate(Bundle savedInstanceState)
{super.onCreate(savedInstanceState);
}

@Override

```

```

public void onViewCreated(@NonNull final View view, @Nullable Bundle savedInstanceState)
{super.onViewCreated(view, savedInstanceState);

/**
 * Set up listener for the journal fragment's FloatingActionButton
 */
FloatingActionButton fab = view.findViewById(R.id.floating_action_button);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO: Add code for creating a journal entry
        Intent intent = new Intent(view.getContext(), NotepadEntry.class);
        startActivityForResult(intent, 0x0);
    }
});

mDatabase = AppDatabase.getDatabase(view.getContext());
final JournalDao journalDao = mDatabase.journalDao();
List<Journal> allJournals = journalDao.getAll();
mArrayList = new ArrayList<>();
for (Journal journal : allJournals)
    {mArrayList.add(journal.title);
}

mAdapter = new ArrayAdapter<String>(view.getContext(), R.layout.list_item, mArrayList);

ListView listView = view.findViewById(R.id.journal_list_view);
listView.setAdapter(mAdapter);
listView.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)
    {
        Journal entry = journalDao.getAll().get(position);
        Intent intent = new Intent(view.getContext(), NotepadEntry.class);
        intent.putExtra("Title", entry.title);
        intent.putExtra("Body", entry.body);
        startActivityForResult(intent, 0x0);
    }
});

mAdapter.notifyDataSetChanged();
}

@Override
public void onResume()
{
    super.onResume();
    mAdapter.notifyDataSetChanged();
}

```

```

    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_journal, container, false);
    }
}

package com.texastech.talk.navigation;

import android.content.Intent;
import android.graphics.Color;
import android.net.Uri;
import android.os.Bundle;
import android.text.Layout;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.ScrollView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.cardview.widget.CardView;
import androidx.fragment.app.Fragment;

import com.texastech.talk.R;
import com.texastech.talk.database.AppDatabase;
import com.texastech.talk.database.Mood;
import com.texastech.talk.database.MoodDao;
import com.texastech.talk.database.Resources;
import com.texastech.talk.database.ResourcesDao;

import java.util.ArrayList;
import java.util.List;

public class ResourcesFragment extends Fragment {
    /**
     * Displays the resources that the user should be reading

```

```

    * based on their mood information.
    */
    public ResourcesFragment() {
        // Required.
    }

    public static ResourcesFragment newInstance()
    {return new ResourcesFragment();
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState)
    {super.onCreate(savedInstanceState);
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
        @Nullable ViewGroup container,
        @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_resources, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState)
    {super.onViewCreated(view, savedInstanceState);

        // Get the relevant articles
        List<Resources> articles = getRelevantResources(view);
        Log.d("Resources", String.format("Found %d articles for your mood", articles.size()));
        for (Resources res : articles) {
            Log.d("Resources", String.format("Found hyperlink %s", res.hyperlink));
        }

        // Depressed = 1, Sad = 2, Angry = 3, Scared = 4, Moderate = 5, Happy = 6
        final int MoodDepressed = 1;
        final int MoodSad = 2;
        final int MoodAngry = 3;
        final int MoodScared = 4;
        final int MoodModerate = 5;
        final int MoodHappy = 6;

        /**
         *
         */
        LinearLayout scrollableLinearLayout = new LinearLayout(view.getContext());

```

```

scrollableLinearLayout.setOrientation(LinearLayout.VERTICAL);
scrollableLinearLayout.setLayoutParams(new LinearLayout.LayoutParams(
    ViewGroup.LayoutParams.MATCH_PARENT,
    ViewGroup.LayoutParams.MATCH_PARENT
));

```

```

// Display the articles in a CardView

```

```

for (final Resources res : articles) {

```

```

    /**

```

```

        * <TextView

```

```

        *         android:layout_width="match_parent"
        *         android:layout_height="wrap_content"
        *         android:layout_marginBottom="8dp"
        *         android:text="Card title"
        *         android:textColor="#000"
        *         android:textSize="18sp" />

```

```

    */

```

```

    TextView textTitle = new TextView(view.getContext());
    textTitle.setLayoutParams(new LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.WRAP_CONTENT
    ));

```

```

    textTitle.setPadding(0, 0, 0, 8);
    textTitle.setText(res.title);
    textTitle.setTextColor(Color.WHITE);
    textTitle.setTextSize(20);

```

```

    /**

```

```

        * <TextView

```

```

        *         android:layout_width="match_parent"
        *         android:layout_height="wrap_content"
        *         android:text="Card content"
        *         android:textColor="#555" />

```

```

    */

```

```

    TextView contentText = new TextView(view.getContext());
    contentText.setLayoutParams(new LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.WRAP_CONTENT
    ));

```

```

    contentText.setText(res.content);
    contentText.setTextSize(14);
    contentText.setTextColor(Color.WHITE);

```

```

    /**

```

```

        * <LinearLayout

```

```

        *         android:layout_width="match_parent"

```

```

*         android:layout_height="wrap_content"
*         android:orientation="vertical"
*         android:padding="16dp">
*         <TextView
*             android:layout_width="match_parent"
*             android:layout_height="wrap_content"
*             android:layout_marginBottom="8dp"
*             android:text="Card title"
*             android:textColor="#000"
*             android:textSize="18sp" />
*         <TextView
*             android:layout_width="match_parent"
*             android:layout_height="wrap_content"
*             android:text="Card content"
*             android:textColor="#555" />
*     </LinearLayout>
*/

LinearLayout textLayout = new LinearLayout(view.getContext());
textLayout.setLayoutParams(new LinearLayout.LayoutParams(
    ViewGroup.LayoutParams.MATCH_PARENT,
    ViewGroup.LayoutParams.WRAP_CONTENT
));
textLayout.setOrientation(LinearLayout.VERTICAL);
textLayout.setPadding(16, 16, 16, 16);
textLayout.addView(textTitle);
textLayout.addView(contentText);

/**
 * <Button
 *         android:layout_width="wrap_content"
 *         android:layout_height="wrap_content"
 *         android:text="Learn more"/>
 */

Button learnMoreBtn = new Button(view.getContext());
RelativeLayout.LayoutParams buttonLayoutParams = new
    RelativeLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
    ViewGroup.LayoutParams.WRAP_CONTENT
);
buttonLayoutParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
learnMoreBtn.setLayoutParams(buttonLayoutParams);
learnMoreBtn.setText("Learn more");
learnMoreBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent hyperlinkIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(res.hyperlink));
        startActivity(hyperlinkIntent);
    }
});

```



```

    }
});

/**
 * <LinearLayout
 *     android:layout_width="match_parent"
 *     android:layout_height="wrap_content">
 *     <Button
 *         android:layout_width="wrap_content"
 *         android:layout_height="wrap_content"
 *         android:text="Learn more"/>
 *     </LinearLayout>
 */
RelativeLayout buttonLayout = new RelativeLayout(view.getContext());
buttonLayout.setLayoutParams(new LinearLayout.LayoutParams(
    ViewGroup.LayoutParams.MATCH_PARENT,
    ViewGroup.LayoutParams.WRAP_CONTENT
));
buttonLayout.addView(learnMoreBtn);

/**
 * <ImageView
 *     android:layout_width="match_parent"
 *     android:layout_height="160dp"
 *     android:scaleType="centerCrop"
 *     android:src="@drawable/abc_vector_test" />
 */
ImageView image = new ImageView(view.getContext());
image.setLayoutParams(new
    LinearLayout.LayoutParams(ViewGroup.LayoutParams.
        MATCH_PARENT,
        200
    ));
image.setScaleType(ImageView.ScaleType.CENTER_CROP);
int imageCode = 0x0;
switch(res.mood) {
    case MoodDepressed:
        imageCode = R.drawable.depressed_background;
        break;
    case MoodSad:
        imageCode = R.drawable.sad_background;
        break;
    case MoodAngry:
        imageCode = R.drawable.angry_background;
        break;
    case MoodScared:
        imageCode = R.drawable.scared_background;

```

```

        break;
    case MoodModerate:
        imageCode = R.drawable.moderate_background;
        break;
    case MoodHappy:
        imageCode = R.drawable.happy_background;
        break;
    }
    image.setImageResource(imageCode);

/**
 * <LinearLayout
 *     android:layout_width="match_parent"
 *     android:layout_height="wrap_content"
 *     android:orientation="vertical">
 */
LinearLayout cardLayout = new LinearLayout(view.getContext());
cardLayout.setLayoutParams(new LinearLayout.LayoutParams(
    ViewGroup.LayoutParams.MATCH_PARENT,
    ViewGroup.LayoutParams.WRAP_CONTENT
));
cardLayout.setOrientation(LinearLayout.VERTICAL);
cardLayout.addView(image);
cardLayout.addView(textLayout);
cardLayout.addView(buttonLayout);

/**
 * <androidx.cardview.widget.CardView
 *     android:id="@+id/card_view"
 *     android:layout_width="match_parent"
 *     android:layout_height="wrap_content"
 *     android:layout_margin="16dp"
 *     android:backgroundTint="#E6E6E6">
 */
CardView card = new CardView(view.getContext());
LinearLayout.LayoutParams cardLayoutParams = new
    LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
    ViewGroup.LayoutParams.WRAP_CONTENT
);
cardLayoutParams.setMargins(16, 16, 16, 16);
card.setLayoutParams(cardLayoutParams);
cardLayout.setOrientation(LinearLayout.VERTICAL);
card.setPadding(20, 20, 20, 20);
card.setBackgroundColor(Color.parseColor("#193C51"));
card.addView(cardLayout);

```

```

        scrollableLinearLayout.addView(card);
    }

    ScrollView scrollView = new ScrollView(view.getContext());
    scrollView.setLayoutParams(new LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT
    ));
    scrollView.addView(scrollableLinearLayout);

    /**
     * <androidx.cardview.widget.CardView
     *     android:id="@+id/card_view"
     *     android:layout_width="match_parent"
     *     android:layout_height="wrap_content"
     *     android:layout_margin="16dp"
     *     android:backgroundTint="#E6E6E6" ...
     */
    LinearLayout linearLayout = view.findViewById(R.id.resources_layout);
    linearLayout.addView(scrollView);
}

private List<Resources> getRelevantResources(View view) {
    /**
     * Returns the resources relevant to the user's mood.
     */
    AppDatabase database = AppDatabase.getDatabase(view.getContext());
    ResourcesDao resDao = database.resourcesDao();

    // Get last mood
    MoodDao moodDao = database.moodDao();
    List<Mood> allMoods = moodDao.getAll();
    Mood lastMood = new Mood(0, 5, 1);
    if (allMoods.size() > 0) {
        lastMood = allMoods.get(allMoods.size() - 1);
    }

    List<Resources> resources = new ArrayList<>();
    List<Resources> allArticles = resDao.getAll();
    for (Resources resource : allArticles) {
        if (resource.mood == lastMood.value)
            {resources.add(resource);
        }
    }
}

return resources;

```

```
}
}
```

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

```
<resources>
```

```
    <color name="colorPrimary">#008577</color>
```

```
    <color name="colorPrimaryDark">#00574B</color>
```

```
    <color name="colorAccent">#D81B60</color>
```

```
    <!-- Core colors -->
```

```
    <color name="colorAppPrimary">#16202A</color>
```

```
    <color name="colorAppSecondary">#16202A</color>
```

```
    <!-- Intro slides colors -->
```

```
    <color name="colorIntroSlideHeader">#FFFFFF</color>
```

```
    <color name="colorIntroSlideBody">#FFFFFF</color>
```

```
    <color name="colorIntroSlide1Background">#343434</color>
```

```
    <color name="colorIntroSlide2Background">#3B6DB2</color>
```

```
    <color name="colorIntroSlide3Background">#CC8A2D</color>
```

```
    <color name="colorIntroSlide4Background">#1C4E3B</color>
```

```
    <color name="colorIntroSlide5Background">#A82222</color>
```

```
    <color name="colorIntroSlide6Background">#318D9A</color>
```

```
    <!-- BottomNavigationView colors -->
```

```
    <color name="colorBottomNavActive">#20A0EE</color>
```

```
    <color name="colorBottomNavInactive">#45535D</color>
```

```
    <color name="colorBottomNavBackground">#16202A</color>
```

```
    <color name="colorBottomNavSeparator">#45535D</color>
```

```
    <!-- Alert dialog colors -->
```

```
    <color name="colorAlertDialogBackground">#16202A</color>
```

```
    <color name="colorAlertDialogButton">#20A0EE</color>
```

```
    <color name="colorAlertDialogText">#EFEFEF</color>
```

```
</resources>
```

```
<resources>
```

```
    <string name="app_name">Talk</string>
```

```
    <!-- Intro slides -->
```

```
    <string name="intro_slide1_header">Welcome to Talk</string>
```

```
    <string name="intro_slide1_body">Talk is a personal tool that helps you keep track of your  
mental health</string>
```

```
    <string name="intro_slide2_header">Journal</string>
```

```
    <string name="intro_slide2_body">Use the journal to write more detailed information about how
```

```

you're feeling</string>
    <string name="intro_slide3_header">Mood Graph</string>
    <string name="intro_slide3_body">Use the graph to view your mood history</string>
    <string name="intro_slide4_header">Resources</string>
    <string name="intro_slide4_body">Get suggested articles and videos that can help you improve
your mood</string>
    <string name="intro_slide5_header">Location and Apps</string>
    <string name="intro_slide5_body">Track your movement and app usage to better understand
your moods</string>
    <string name="intro_slide6_header">Terms and Conditions</string>
    <string name="intro_slide6_body">Please read the terms and conditions before you use the
app</string>

<!-- BottomNavigationView strings -->
    <string name="menu_journal">Journal</string>
    <string name="menu_resources">Resources</string>
    <string name="menu_statistics">Statistics</string>
    <string name="menu_settings">Settings</string>
</resources>

<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Primary and secondary are the same for now -->
        <item name="colorPrimary">@color/colorAppPrimary</item>
        <item name="colorPrimaryDark">@color/colorAppSecondary</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="alertDialogStyle">@style/Theme.MaterialComponents.Dialog.Alert</item>
        <item name="android:textColor">#FFFFFF</item>
    </style>

    <style name="FloatingActionBtnTheme">
        <item name="android:backgroundTint">@color/colorBottomNavActive</item>
        <item name="android:tint">@android:color/white</item>
    </style>

    <style name="DarkAlertDialog" parent="Theme.MaterialComponents.Dialog.Alert">
        <item name="colorAccent">@color/colorAlertDialogButton</item>
        <item name="android:textColorPrimary">@color/colorAlertDialogText</item>
        <item name="android:background">@color/colorAlertDialogBackground</item>
    </style>

    <style name="PreferenceScreen" parent="Theme.AppCompat">
        <item name="android:textColor">#ffffff</item>

```

```

</style>
</resources>

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.texastech.talk">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".NotepadEntry"></activity>
        <activity
            android:name=".MainActivity"
            android:clearTaskOnLaunch="true"
            android:finishOnTaskLaunch="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".intro.IntroActivity" />

        <receiver android:name=".notification.AlarmReceiver" />
    </application>

</manifest>

```

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/colorBottomNavBackground"
    tools:context=".MainActivity">

```

```

<fragment

```

```

        android:id="@+id/nav_host_fragment"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        app:defaultNavHost="true"
        app:navGraph="@navigation/nav_graph"/>

```

```

<View
    android:id="@+id/view"
    android:layout_width="match_parent"
    android:layout_height="0.1sp"
    android:background="@color/colorBottomNavSeparator"/>

```

```

<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottom_nav_view"
    android:layout_width="match_parent"
    android:layout_height="56dp"
    android:layout_gravity="start"
    app:menu="@menu/bottom_nav_items"
    app:labelVisibilityMode="unlabeled"
    app:itemIconTint="@color/bottom_navigation_colors"
    app:itemTextColor="@color/bottom_navigation_colors"
    app:itemBackground="@color/colorBottomNavBackground"
    tools:layout_editor_absoluteX="0dp"/>

```

```

</LinearLayout>

```

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/colorBottomNavBackground"
    tools:context=".NotepadEntry">

```

```

<EditText
    android:id="@+id/entry_title"
    android:layout_width="match_parent"
    android:layout_height="52dp"
    android:hint="Title"
    android:textColorHint="@android:color/darker_gray"
    android:textColor="@android:color/white" />

```

```

<EditText
    android:id="@+id/entry_body"
    android:layout_width="match_parent"
    android:layout_height="523dp"
    android:hint="Write about what you feel"
    android:textAlignment="center"
    android:textColor="@android:color/white"
    android:textColorHint="@android:color/darker_gray" />

<Button
    android:id="@+id/save_entry"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Save" />
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".navigation.JournalFragment">

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/floating_action_button"
        style="@style/FloatingActionBtnTheme"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        android:src="@drawable/ic_add"
        android:elevation="6dp"
        app:borderWidth="0dp"/>

    <ListView
        android:id="@+id/journal_list_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>

```



```

<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nav_graph"
    app:startDestination="@id/journalFragment">

    <fragment
        android:id="@+id/journalFragment"
        android:name="com.texastech.talk.navigation.JournalFragment"
        android:label="fragment_journal"
        tools:layout="@layout/fragment_journal" >
        <action
            android:id="@+id/action_journalFragment_to_resourcesFragment"
            app:destination="@id/resourcesFragment" />
    </fragment>
    <fragment
        android:id="@+id/resourcesFragment"
        android:name="com.texastech.talk.navigation.ResourcesFragment"
        android:label="fragment_resources"
        tools:layout="@layout/fragment_resources" >
        <action
            android:id="@+id/action_resourcesFragment_to_statisticsFragment"
            app:destination="@id/statisticsFragment" />
    </fragment>
    <fragment
        android:id="@+id/statisticsFragment"
        android:name="com.texastech.talk.navigation.StatisticsFragment"
        android:label="fragment_statistics"
        tools:layout="@layout/fragment_statistics" >
        <action
            android:id="@+id/action_statisticsFragment_to_settingsFragment"
            app:destination="@id/settingsFragment" />
    </fragment>
    <fragment
        android:id="@+id/settingsFragment"
        android:name="com.texastech.talk.navigation.SettingsFragment"
        android:label="fragment_settings"
        tools:layout="@layout/fragment_settings" />
</navigation>

```

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/outro_slide_layout"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:background="@color/colorIntroSlide6Background"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="100dp"
        android:layout_gravity="center"
        android:text="@string/intro_slide6_header"
        android:textSize="29sp"
        android:textColor="@color/colorIntroSlideHeader"
        android:fontFamily="sans-serif-thin"/>

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:layout_marginTop="80dp"
        android:layout_gravity="center"
        app:srcCompat="@drawable/ic_intro_slide6"/>

    <TextView
        android:layout_width="240dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="80dp"
        android:text="@string/intro_slide6_body"
        android:textSize="15sp"
        android:textAlignment="center"
        android:textColor="@color/colorIntroSlideBody"
        android:fontFamily="sans-serif"/>
</LinearLayout>

```

```

/**
 * Automatically generated file. DO NOT MODIFY
 */
package com.texastech.talk;

public final class BuildConfig {
    public static final boolean DEBUG = Boolean.parseBoolean("true");
    public static final String APPLICATION_ID = "com.texastech.talk";
    public static final String BUILD_TYPE = "debug";
    public static final String FLAVOR = "";
    public static final int VERSION_CODE = 1;
    public static final String VERSION_NAME = "1.0";
}

package com.texastech.talk.database;

import androidx.room.DatabaseConfiguration;
import androidx.room.InvalidTracker;
import androidx.room.RoomOpenHelper;
import androidx.room.RoomOpenHelper.Delegate;
import androidx.room.RoomOpenHelper.ValidationResult;
import androidx.room.util.DBUtil;
import androidx.room.util.TableInfo;
import androidx.room.util.TableInfo.Column;
import androidx.room.util.TableInfo.ForeignKey;
import androidx.room.util.TableInfo.Index;
import androidx.sqlite.db.SupportSQLiteDatabase;
import androidx.sqlite.db.SupportSQLiteOpenHelper;
import androidx.sqlite.db.SupportSQLiteOpenHelper.Callback;
import androidx.sqlite.db.SupportSQLiteOpenHelper.Configuration;
import java.lang.Override;
import java.lang.String;
import java.lang.SuppressWarnings;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Set;

@SuppressWarnings({"unchecked", "deprecation"})
public final class AppDatabase_Impl extends AppDatabase {
    private volatile MoodDao _moodDao;

    private volatile ResourcesDao _resourcesDao;

    private volatile JournalDao _journalDao;

    @Override
    protected SupportSQLiteOpenHelper createOpenHelper(DatabaseConfiguration configuration) {

```

```

final SupportSQLiteOpenHelper.Callback _openCallback = new RoomOpenHelper(configuration, new
RoomOpenHelper.Delegate(1) {
    @Override
    public void createAllTables(SupportSQLiteDatabase _db) {
        _db.execSQL("CREATE TABLE IF NOT EXISTS `Mood` (`mid` INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL, `date` INTEGER NOT NULL, `value` INTEGER NOT NULL,
`severity_level` INTEGER NOT NULL)");
        _db.execSQL("CREATE TABLE IF NOT EXISTS `Resources` (`rid` INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL, `title` TEXT, `content` TEXT, `hyperlink` TEXT, `mood` INTEGER NOT
NULL)");
        _db.execSQL("CREATE TABLE IF NOT EXISTS `Journal` (`jid` INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL, `title` TEXT, `body` TEXT)");
        _db.execSQL("CREATE TABLE IF NOT EXISTS room_master_table (id INTEGER PRIMARY
KEY,identity_hash TEXT)");
        _db.execSQL("INSERT OR REPLACE INTO room_master_table (id,identity_hash) VALUES(42,
'733f05d836672d525142469532f8a03b')");
    }

    @Override
    public void dropAllTables(SupportSQLiteDatabase _db) {
        _db.execSQL("DROP TABLE IF EXISTS `Mood`");
        _db.execSQL("DROP TABLE IF EXISTS `Resources`");
        _db.execSQL("DROP TABLE IF EXISTS `Journal`");
        if (mCallbacks != null) {
            for (int _i = 0, _size = mCallbacks.size(); _i < _size; _i++) {
                mCallbacks.get(_i).onDestructiveMigration(_db);
            }
        }
    }

    @Override
    protected void onCreate(SupportSQLiteDatabase _db) {
        if (mCallbacks != null) {
            for (int _i = 0, _size = mCallbacks.size(); _i < _size; _i++) {
                mCallbacks.get(_i).onCreate(_db);
            }
        }
    }

    @Override
    public void onOpen(SupportSQLiteDatabase _db) {
        mDatabase = _db;
        internalInitInvalidationTracker(_db);
        if (mCallbacks != null) {
            for (int _i = 0, _size = mCallbacks.size(); _i < _size; _i++) {
                mCallbacks.get(_i).onOpen(_db);
            }
        }
    }

    @Override
    public void onPreMigrate(SupportSQLiteDatabase _db) {
        DBUtil.dropFtsSyncTriggers(_db);
    }
}

```

```

    }

    @Override
    public void onPostMigrate(SupportSQLiteDatabase _db) {
    }

    @Override
    protected RoomOpenHelper.ValidationResult onValidateSchema(SupportSQLiteDatabase _db) {
        final HashMap<String, TableInfo.Column> _columnsMood = new HashMap<String,
TableInfo.Column>(4);
        _columnsMood.put("mid", new TableInfo.Column("mid", "INTEGER", true, 1, null,
TableInfo.CREATED_FROM_ENTITY));
        _columnsMood.put("date", new TableInfo.Column("date", "INTEGER", true, 0, null,
TableInfo.CREATED_FROM_ENTITY));
        _columnsMood.put("value", new TableInfo.Column("value", "INTEGER", true, 0, null,
TableInfo.CREATED_FROM_ENTITY));
        _columnsMood.put("severity_level", new TableInfo.Column("severity_level", "INTEGER", true, 0, null,
TableInfo.CREATED_FROM_ENTITY));
        final HashSet<TableInfo.ForeignKey> _foreignKeysMood = new HashSet<TableInfo.ForeignKey>(0);
        final HashSet<TableInfo.Index> _indicesMood = new HashSet<TableInfo.Index>(0);
        final TableInfo _infoMood = new TableInfo("Mood", _columnsMood, _foreignKeysMood, _indicesMood);
        final TableInfo _existingMood = TableInfo.read(_db, "Mood");
        if (! _infoMood.equals(_existingMood)) {
            return new RoomOpenHelper.ValidationResult(false, "Mood(com.texastech.talk.database.Mood).\n"
                + " Expected:\n" + _infoMood + "\n"
                + " Found:\n" + _existingMood);
        }
        final HashMap<String, TableInfo.Column> _columnsResources = new HashMap<String,
TableInfo.Column>(5);
        _columnsResources.put("rid", new TableInfo.Column("rid", "INTEGER", true, 1, null,
TableInfo.CREATED_FROM_ENTITY));
        _columnsResources.put("title", new TableInfo.Column("title", "TEXT", false, 0, null,
TableInfo.CREATED_FROM_ENTITY));
        _columnsResources.put("content", new TableInfo.Column("content", "TEXT", false, 0, null,
TableInfo.CREATED_FROM_ENTITY));
        _columnsResources.put("hyperlink", new TableInfo.Column("hyperlink", "TEXT", false, 0, null,
TableInfo.CREATED_FROM_ENTITY));
        _columnsResources.put("mood", new TableInfo.Column("mood", "INTEGER", true, 0, null,
TableInfo.CREATED_FROM_ENTITY));
        final HashSet<TableInfo.ForeignKey> _foreignKeysResources = new HashSet<TableInfo.ForeignKey>(0);
        final HashSet<TableInfo.Index> _indicesResources = new HashSet<TableInfo.Index>(0);
        final TableInfo _infoResources = new TableInfo("Resources", _columnsResources, _foreignKeysResources,
_indexResources);
        final TableInfo _existingResources = TableInfo.read(_db, "Resources");
        if (! _infoResources.equals(_existingResources)) {
            return new RoomOpenHelper.ValidationResult(false,
"Resources(com.texastech.talk.database.Resources).\n"
                + " Expected:\n" + _infoResources + "\n"
                + " Found:\n" + _existingResources);
        }
        final HashMap<String, TableInfo.Column> _columnsJournal = new HashMap<String,
TableInfo.Column>(3);
        _columnsJournal.put("jid", new TableInfo.Column("jid", "INTEGER", true, 1, null,

```

```

TableInfo.CREATED_FROM_ENTITY));
    _columnsJournal.put("title", new TableInfo.Column("title", "TEXT", false, 0, null,
TableInfo.CREATED_FROM_ENTITY));
    _columnsJournal.put("body", new TableInfo.Column("body", "TEXT", false, 0, null,
TableInfo.CREATED_FROM_ENTITY));
    final HashSet<TableInfo.ForeignKey> _foreignKeysJournal = new HashSet<TableInfo.ForeignKey>(0);
    final HashSet<TableInfo.Index> _indicesJournal = new HashSet<TableInfo.Index>(0);
    final TableInfo _infoJournal = new TableInfo("Journal", _columnsJournal, _foreignKeysJournal,
    _indicesJournal);
    final TableInfo _existingJournal = TableInfo.read(_db, "Journal");
    if (! _infoJournal.equals(_existingJournal)) {
        return new RoomOpenHelper.ValidationResult(false, "Journal(com.texastech.talk.database.Journal).\n"
            + " Expected:\n" + _infoJournal + "\n"
            + " Found:\n" + _existingJournal);
    }
    return new RoomOpenHelper.ValidationResult(true, null);
}
}, "733f05d836672d525142469532f8a03b", "773d83fddeab381801cc886a5b2d8247");
final SupportSQLiteOpenHelper.Configuration _sqliteConfig =
SupportSQLiteOpenHelper.Configuration.builder(configuration.context)
    .name(configuration.name)
    .callback(_openCallback)
    .build();
final SupportSQLiteOpenHelper _helper = configuration.sqliteOpenHelperFactory.create(_sqliteConfig);
return _helper;
}

@Override
protected InvalidationTracker createInvalidationTracker() {
    final HashMap<String, String> _shadowTablesMap = new HashMap<String, String>(0);
    HashMap<String, Set<String>> _viewTables = new HashMap<String, Set<String>>(0);
    return new InvalidationTracker(this, _shadowTablesMap, _viewTables, "Mood", "Resources", "Journal");
}

@Override
public void clearAllTables() {
    super.assertNotMainThread();
    final SupportSQLiteDatabase _db = super.getOpenHelper().getWritableDatabase();
    try {
        super.beginTransaction();
        _db.execSQL("DELETE FROM `Mood`");
        _db.execSQL("DELETE FROM `Resources`");
        _db.execSQL("DELETE FROM `Journal`");
        super.setTransactionSuccessful();
    } finally {
        super.endTransaction();
        _db.query("PRAGMA wal_checkpoint(FULL)").close();
        if (!_db.inTransaction()) {
            _db.execSQL("VACUUM");
        }
    }
}
}

```

```

@Override
public MoodDao moodDao() {
    if (_moodDao != null) {
        return _moodDao;
    } else {
        synchronized(this) {
            if(_moodDao == null) {
                _moodDao = new MoodDao_Impl(this);
            }
            return _moodDao;
        }
    }
}

```

```

@Override
public ResourcesDao resourcesDao() {
    if (_resourcesDao != null) {
        return _resourcesDao;
    } else {
        synchronized(this) {
            if(_resourcesDao == null) {
                _resourcesDao = new ResourcesDao_Impl(this);
            }
            return _resourcesDao;
        }
    }
}

```

```

@Override
public JournalDao journalDao() {
    if (_journalDao != null) {
        return _journalDao;
    } else {
        synchronized(this) {
            if(_journalDao == null) {
                _journalDao = new JournalDao_Impl(this);
            }
            return _journalDao;
        }
    }
}

```

```

package com.texastech.talk.database;

import android.database.Cursor;
import androidx.room.EntityDeletionOrUpdateAdapter;
import androidx.room.EntityInsertionAdapter;
import androidx.room.RoomDatabase;
import androidx.room.RoomSQLiteQuery;
import androidx.room.util.CursorUtil;
import androidx.room.util.DBUtil;
import androidx.sqlite.db.SupportSQLiteStatement;
import java.lang.Override;
import java.lang.String;
import java.lang.SuppressWarnings;
import java.util.ArrayList;
import java.util.List;

@SuppressWarnings({"unchecked", "deprecation"})
public final class JournalDao_Impl implements JournalDao {
    private final RoomDatabase __db;

    private final EntityInsertionAdapter<Journal> __insertionAdapterOfJournal;

    private final EntityDeletionOrUpdateAdapter<Journal> __deletionAdapterOfJournal;

    public JournalDao_Impl(RoomDatabase __db) {
        this.__db = __db;
        this._insertionAdapterOfJournal = new EntityInsertionAdapter<Journal>(_db) {
            @Override
            public String createQuery() {
                return "INSERT OR ABORT INTO `Journal` (`jid`,`title`,`body`) VALUES (nullif(?, 0),?,?)";
            }

            @Override
            public void bind(SupportSQLiteStatement stmt, Journal value) {
                stmt.bindLong(1, value.jid);
                if (value.title == null) {
                    stmt.bindNull(2);
                } else {
                    stmt.bindString(2, value.title);
                }
                if (value.body == null) {
                    stmt.bindNull(3);
                } else {
                    stmt.bindString(3, value.body);
                }
            }
        };
    }
};

```



```

this._deletionAdapterOfJournal = new EntityDeletionOrUpdateAdapter<Journal>(_db) {
    @Override
    public String createQuery() {
        return "DELETE FROM `Journal` WHERE `jid` = ?";
    }

    @Override
    public void bind(SupportSQLiteStatement stmt, Journal value) {
        stmt.bindLong(1, value.jid);
    }
};

@Override
public void insert(final Journal journal) {
    __db.assertNotSuspendingTransaction();
    __db.beginTransaction();
    try {
        __insertionAdapterOfJournal.insert(journal);
        __db.setTransactionSuccessful();
    } finally {
        __db.endTransaction();
    }
}

@Override
public void insertAll(final Journal... journals) {
    __db.assertNotSuspendingTransaction();
    __db.beginTransaction();
    try {
        __insertionAdapterOfJournal.insert(journals);
        __db.setTransactionSuccessful();
    } finally {
        __db.endTransaction();
    }
}

@Override
public void delete(final Journal journal) {
    __db.assertNotSuspendingTransaction();
    __db.beginTransaction();
    try {
        __deletionAdapterOfJournal.handle(journal);
        __db.setTransactionSuccessful();
    } finally {
        __db.endTransaction();
    }
}

```

```

@Override
public List<Journal> getAll() {
    final String _sql = "SELECT * FROM journal";
    final RoomSQLiteQuery _statement = RoomSQLiteQuery.acquire(_sql, 0);
    __db.assertNotSuspendingTransaction();
    final Cursor _cursor = DBUtil.query(__db, _statement, false, null);
    try {
        final int _cursorIndexOfJid = CursorUtil.getColumnIndexOrThrow(_cursor, "jid");
        final int _cursorIndexOfTitle = CursorUtil.getColumnIndexOrThrow(_cursor, "title");
        final int _cursorIndexOfBody = CursorUtil.getColumnIndexOrThrow(_cursor, "body");
        final List<Journal> _result = new ArrayList<Journal>(_cursor.getCount());
        while(_cursor.moveToNext()) {
            final Journal _item;
            final String _tmpTitle;
            _tmpTitle = _cursor.getString(_cursorIndexOfTitle);
            final String _tmpBody;
            _tmpBody = _cursor.getString(_cursorIndexOfBody);
            _item = new Journal(_tmpTitle, _tmpBody);
            _item.jid = _cursor.getInt(_cursorIndexOfJid);
            _result.add(_item);
        }
        return _result;
    } finally {
        _cursor.close();
        _statement.release();
    }
}
}
}

```

```

package com.texastech.talk.database;

```

```

import android.database.Cursor;
import androidx.room.EntityDeletionOrUpdateAdapter;
import androidx.room.EntityInsertionAdapter;
import androidx.room.RoomDatabase;
import androidx.room.RoomSQLiteQuery;
import androidx.room.util.CursorUtil;
import androidx.room.util.DBUtil;
import androidx.sqlite.db.SupportSQLiteStatement;
import java.lang.Override;
import java.lang.String;
import java.lang.SuppressWarnings;
import java.util.ArrayList;
import java.util.List;

```

```

@SuppressWarnings({"unchecked", "deprecation"})
public final class MoodDao_Impl implements MoodDao {

```

```

private final RoomDatabase __db;

private final EntityInsertionAdapter<Mood> __insertionAdapterOfMood;

private final EntityDeletionOrUpdateAdapter<Mood> __deletionAdapterOfMood;

public MoodDao_Impl(RoomDatabase __db) {
    this.__db = __db;
    this._insertionAdapterOfMood = new EntityInsertionAdapter<Mood>(__db) {
        @Override
        public String createQuery() {
            return "INSERT OR ABORT INTO `Mood` (`mid`,`date`,`value`,`severity_level`) VALUES
(nullif(?, 0),?,?,?)";
        }

        @Override
        public void bind(SupportSQLiteStatement stmt, Mood value) {
            stmt.bindLong(1, value.mid);
            stmt.bindLong(2, value.date);
            stmt.bindLong(3, value.value);
            stmt.bindLong(4, value.severityLevel);
        }
    };
    this._deletionAdapterOfMood = new EntityDeletionOrUpdateAdapter<Mood>(__db) {
        @Override
        public String createQuery() {
            return "DELETE FROM `Mood` WHERE `mid` = ?";
        }

        @Override
        public void bind(SupportSQLiteStatement stmt, Mood value) {
            stmt.bindLong(1, value.mid);
        }
    };
}

@Override
public void insert(final Mood mood) {
    __db.assertNotSuspendingTransaction();
    __db.beginTransaction();
    try {
        __insertionAdapterOfMood.insert(mood);
        __db.setTransactionSuccessful();
    } finally {
        __db.endTransaction();
    }
}

@Override

```

```

public void insertAll(final Mood... moods) {
    __db.assertNotSuspendingTransaction();
    __db.beginTransaction();
    try {
        __insertionAdapterOfMood.insert(moods);
        __db.setTransactionSuccessful();
    } finally {
        __db.endTransaction();
    }
}

```

```

@Override
public void delete(final Mood mood) {
    __db.assertNotSuspendingTransaction();
    __db.beginTransaction();
    try {
        __deletionAdapterOfMood.handle(mood);
        __db.setTransactionSuccessful();
    } finally {
        __db.endTransaction();
    }
}

```

```

@Override
public List<Mood> getAll() {
    final String _sql = "SELECT * FROM mood";
    final RoomSQLiteQuery _statement = RoomSQLiteQuery.acquire(_sql, 0);
    __db.assertNotSuspendingTransaction();
    final Cursor _cursor = DBUtil.query(__db, _statement, false, null);
    try {
        final int _cursorIndexOfMid = CursorUtil.getColumnIndexOrThrow(_cursor, "mid");
        final int _cursorIndexOfDate = CursorUtil.getColumnIndexOrThrow(_cursor, "date");
        final int _cursorIndexOfValue = CursorUtil.getColumnIndexOrThrow(_cursor, "value");
        final int _cursorIndexOfSeverityLevel = CursorUtil.getColumnIndexOrThrow(_cursor,
"severity_level");
        final List<Mood> _result = new ArrayList<Mood>(_cursor.getCount());
        while(_cursor.moveToNext()) {
            final Mood _item;
            final int _tmpDate;
            _tmpDate = _cursor.getInt(_cursorIndexOfDate);
            final int _tmpValue;
            _tmpValue = _cursor.getInt(_cursorIndexOfValue);
            final int _tmpSeverityLevel;
            _tmpSeverityLevel = _cursor.getInt(_cursorIndexOfSeverityLevel);
            _item = new Mood(_tmpDate, _tmpValue, _tmpSeverityLevel);
            _item.mid = _cursor.getInt(_cursorIndexOfMid);
            _result.add(_item);
        }
    }
    return _result;
}

```

```

    } finally {
        _cursor.close();
        _statement.release();
    }
}
}

```

```
package com.texastech.talk.database;
```

```

import android.database.Cursor;
import androidx.room.EntityDeletionOrUpdateAdapter;
import androidx.room.EntityInsertionAdapter;
import androidx.room.RoomDatabase;
import androidx.room.RoomSQLiteQuery;
import androidx.room.util.CursorUtil;
import androidx.room.util.DBUtil;
import androidx.sqlite.db.SupportSQLiteStatement;
import java.lang.Override;
import java.lang.String;
import java.lang.SuppressWarnings;
import java.util.ArrayList;
import java.util.List;

```

```

@SuppressWarnings({"unchecked", "deprecation"})
public final class ResourcesDao_Impl implements ResourcesDao {
    private final RoomDatabase __db;

    private final EntityInsertionAdapter<Resources> __insertionAdapterOfResources;

    private final EntityDeletionOrUpdateAdapter<Resources> __deletionAdapterOfResources;

    public ResourcesDao_Impl(RoomDatabase __db) {
        this.__db = __db;
        this._insertionAdapterOfResources = new EntityInsertionAdapter<Resources>(__db) {
            @Override
            public String createQuery() {
                return "INSERT OR ABORT INTO `Resources` (`rid`,`title`,`content`,`hyperlink`,`mood`)
VALUES (nullif(?, 0),?,?,?,?)";
            }

            @Override
            public void bind(SupportSQLiteStatement stmt, Resources value) {
                stmt.bindLong(1, value.rid);
                if (value.title == null) {
                    stmt.bindNull(2);
                } else {
                    stmt.bindString(2, value.title);
                }
                if (value.content == null) {

```

```

        stmt.bindNull(3);
    } else {
        stmt.bindString(3, value.content);
    }
    if (value.hyperlink == null) {
        stmt.bindNull(4);
    } else {
        stmt.bindString(4, value.hyperlink);
    }
    stmt.bindLong(5, value.mood);
}
};
this._deletionAdapterOfResources = new EntityDeletionOrUpdateAdapter<Resources>(_db) {
    @Override
    public String createQuery() {
        return "DELETE FROM `Resources` WHERE `rid` = ?";
    }

    @Override
    public void bind(SupportSQLiteStatement stmt, Resources value) {
        stmt.bindLong(1, value.rid);
    }
};
}

@Override
public void insert(final Resources resources) {
    __db.assertNotSuspendingTransaction();
    __db.beginTransaction();
    try {
        __insertionAdapterOfResources.insert(resources);
        __db.setTransactionSuccessful();
    } finally {
        __db.endTransaction();
    }
}

@Override
public void insertAll(final Resources... resources) {
    __db.assertNotSuspendingTransaction();
    __db.beginTransaction();
    try {
        __insertionAdapterOfResources.insert(resources);
        __db.setTransactionSuccessful();
    } finally {
        __db.endTransaction();
    }
}
}

```

```

@Override
public void delete(final Resources resource) {
    __db.assertNotSuspendingTransaction();
    __db.beginTransaction();
    try {
        __deletionAdapterOfResources.handle(resource);
        __db.setTransactionSuccessful();
    } finally {
        __db.endTransaction();
    }
}

```

```

@Override
public List<Resources> getAll() {
    final String _sql = "SELECT * FROM resources";
    final RoomSQLiteQuery _statement = RoomSQLiteQuery.acquire(_sql, 0);
    __db.assertNotSuspendingTransaction();
    final Cursor _cursor = DBUtil.query(__db, _statement, false, null);
    try {
        final int _cursorIndexOfRid = CursorUtil.getColumnIndexOrThrow(_cursor, "rid");
        final int _cursorIndexOfTitle = CursorUtil.getColumnIndexOrThrow(_cursor, "title");
        final int _cursorIndexOfContent = CursorUtil.getColumnIndexOrThrow(_cursor, "content");
        final int _cursorIndexOfHyperlink = CursorUtil.getColumnIndexOrThrow(_cursor, "hyperlink");
        final int _cursorIndexOfMood = CursorUtil.getColumnIndexOrThrow(_cursor, "mood");
        final List<Resources> _result = new ArrayList<Resources>(_cursor.getCount());
        while(_cursor.moveToNext()) {
            final Resources _item;
            final String _tmpTitle;
            _tmpTitle = _cursor.getString(_cursorIndexOfTitle);
            final String _tmpContent;
            _tmpContent = _cursor.getString(_cursorIndexOfContent);
            final String _tmpHyperlink;
            _tmpHyperlink = _cursor.getString(_cursorIndexOfHyperlink);
            final int _tmpMood;
            _tmpMood = _cursor.getInt(_cursorIndexOfMood);
            _item = new Resources(_tmpTitle, _tmpContent, _tmpHyperlink, _tmpMood);
            _item.rid = _cursor.getInt(_cursorIndexOfRid);
            _result.add(_item);
        }
        return _result;
    } finally {
        _cursor.close();
        _statement.release();
    }
}

```

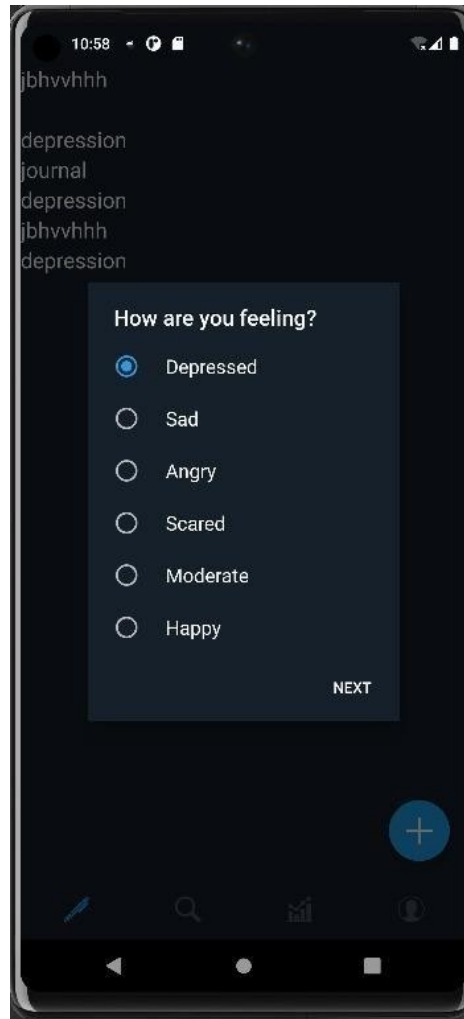
5.2 SAMPLE SCREEN SHOTS

HOME :



Fig 2.0

QUESTIONNAIRE:



10:58

jbhvvhhh

depression

journal

depression

jbhvvhhh

depression

How are you feeling?

☒ Depressed

☐ Sad

☐ Angry

☐ Scared

☐ Moderate

☐ Happy

NEXT

+

✍️ 🔍 📊 👤

Fig 2.1

NOTIFICATION:

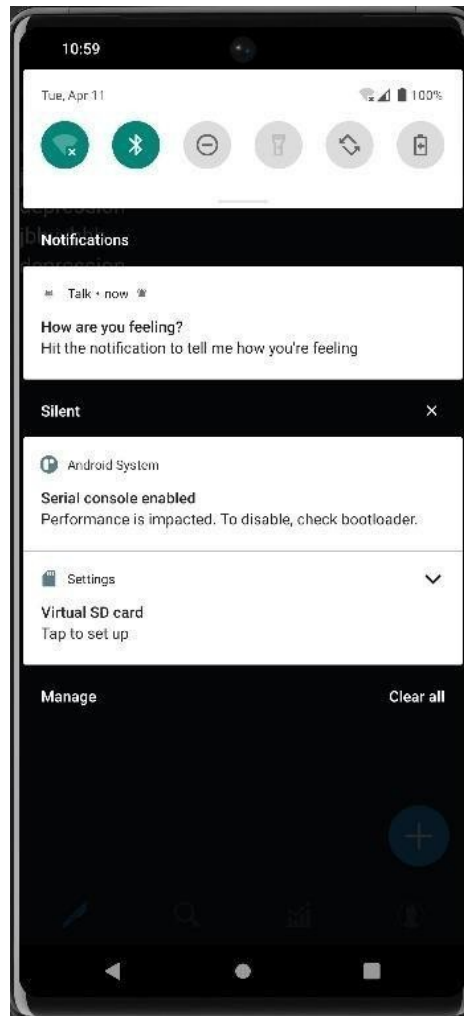


Fig 2.2

RESOURCES:

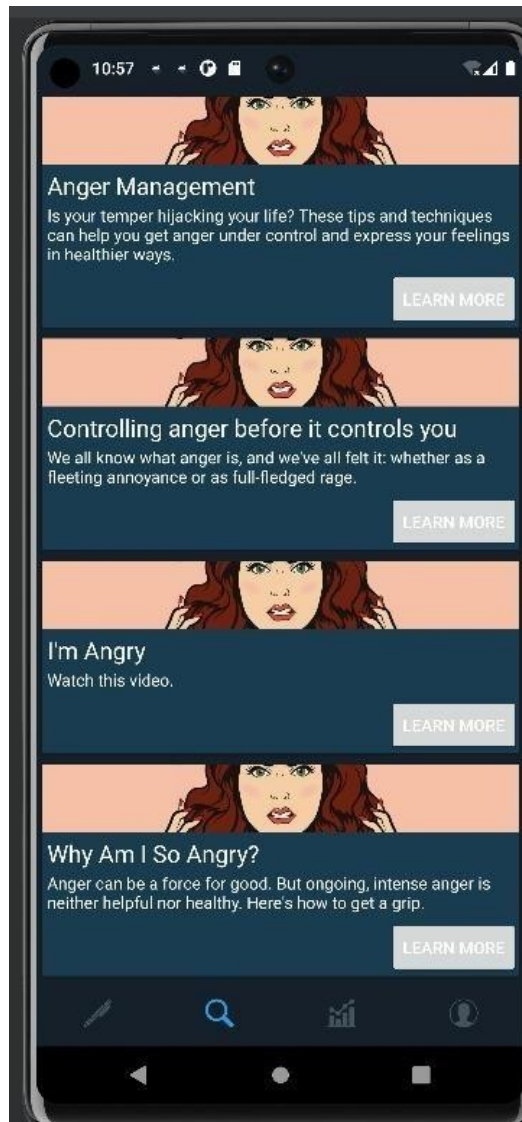


Fig 2.3

JOURNAL:

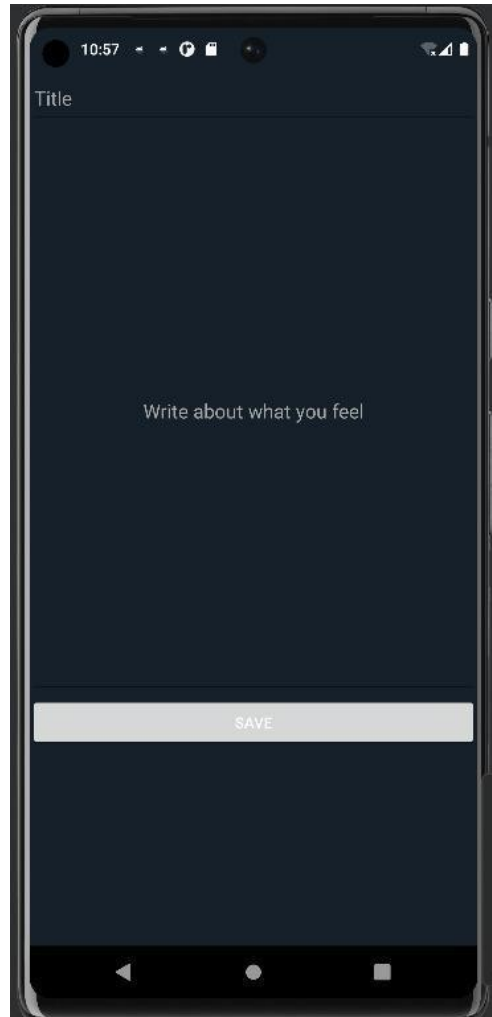


Fig 2.4

GRAPH/TRACKER:

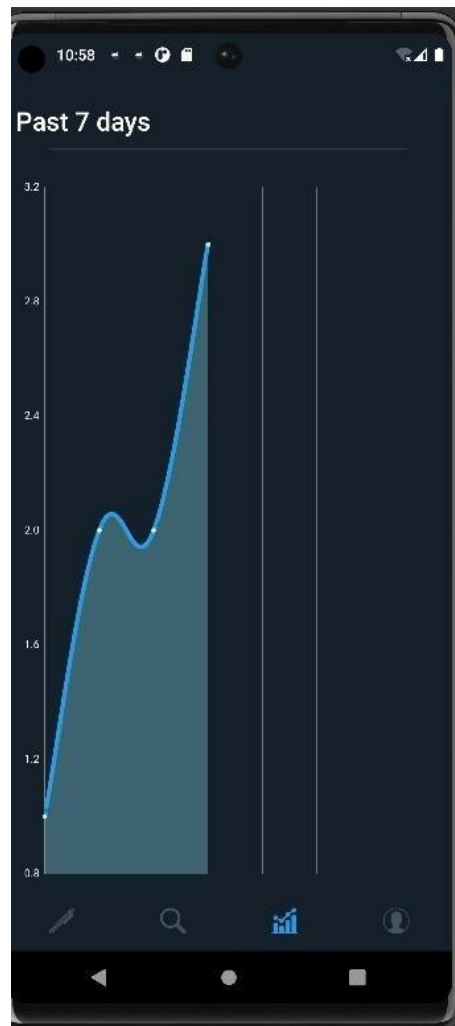


Fig 2.5

PROFILE:



Fig 2.6

FUTURE ENHANCEMENTS:

AI Chat bot Conversation

- The chat bot converses with the user and engage in positive talks hence not letting him/her alone. It also suggests fun activities and articles for keeping the user motivated.
- It reads the messages the user sends and responds accordingly.

Sentiment Analysis

- This is the main module of the application. It records both audios through in person and in call meetings as well as the text messages. It then separates the messages according to the speaker and analysis them. It then identifies the messages as positive, negative or neutral.

Feedback

- The application provides feedback based on sentiment analysis done by the application. It shows whether the conversations done were positive, negative or neutral. It then suggests what should have been spoken in place of negative words to the user (or relative).
- It takes the conversations (both audio and text messages) and analyses them for further processing.

CONCLUSION

The PTSD and Depression Mood Tracker app is an innovative solution that has the potential to revolutionize mental health treatment. The app provides a comprehensive toolkit for people struggling with PTSD and depression, allowing them to track their symptoms, identify triggers, and manage their condition more effectively.

One of the key features of the app is the daily check-in feature, which allows users to monitor their mood and mental state on a regular basis. This helps them to stay on top of their symptoms and take action when they notice any changes or patterns.

The app also includes a range of treatable symptoms, which provides users with a wealth of information on the various symptoms associated with PTSD and depression. This information is invaluable for those who are just starting to explore their condition or who are looking to deepen their understanding of their symptoms.

In addition to these features, the app also provides mood charts, alerts, and troubleshooting tips, which can help users to better manage their condition and work towards recovery. The privacy and security measures built into the app are also important, as they ensure that users' personal information is kept safe and secure at all times.

Overall, the development of apps like the PTSD and Depression Mood Tracker app demonstrates the potential of technology to help people with their mental health and improve their overall well-being. With continued innovation and development in this area, we can expect to see even more powerful tools and resources emerge to support mental health in the future.

Reference:

1. **AI Can Now Detect Depression From Your Voice, And It's Twice As Accurate As Human Practitioners** :<https://www.forbes.com/sites/ganeskesari/2021/05/24/ai-can-now-detectdepression-from-just-your-voice/?sh=2eecab914c8d>
2. **Your Next Therapist Could Be a Chatbot App** :
<https://www.discovermagazine.com/mind/your-next-therapist-could-be-a-chat-bot-app>
3. **Mental Health and COVID-19:** <https://www.camh.ca/en/health-info/guides-and-publications>
4. **Closing treatment gap of mental disorders in India:**
<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6278227/>
5. **Addiction : An information guide** <https://www.camh.ca/-/media/files/guides-and-publications/addiction-guide-en.pdf>
6. **Finding Mental Health Services** <https://www.camh.ca/-/media/files/guides-and-publications/challenges-choices-2003.pdf>
7. **Talking About Mental Illness - Community Guide** <https://www.camh.ca/-/media/files/guides-and-publications/tami-community-guide.pdf>.

