INTRODUCTION

1.1 OVERVIEW

The app is a sample project that demonstrates how to use the Android Compose UI toolkit to build a survey app. The app allows the user to answer a series of questions. It showcases some of the key features of the Compose UI toolkit, data management, and user interactions. With Compose, we were able to create a visually stunning interface that's intuitive and easy to use. By leveraging composer's declarative programming model, we were able to simplify the app's logic and make it more maintainable. One of the key benefits of using Compose is its ability to handle state and data management seamlessly. We utilized composer's state management features to store user responses and to update the UI accordingly. This made it easy to manage the app's flow and provide a smooth user experience.

Another advantage of using Compose is the ability to customize user interactions. We used composes gesture detection features to enable users to swipe between questions, making the app feel more interactive and engaging. Overall, this sample project serves as a great example of how to use the Android Compose UI toolkit to create a functional and visually appealing

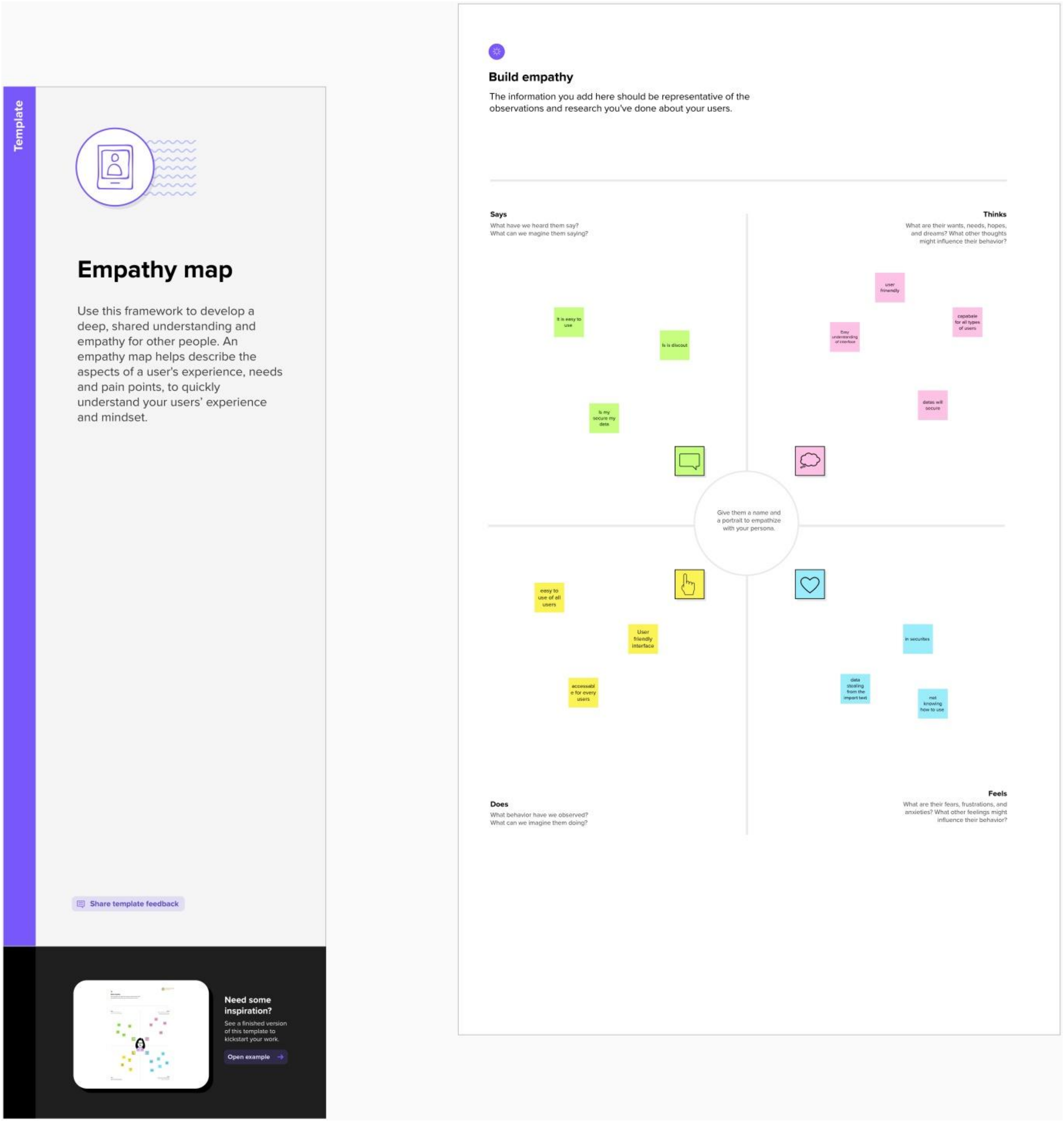
1.2 PURPOSE

The purpose of the survey app built using the Android Compose UI toolkit is to showcase the capabilities of the toolkit and to provide a sample project for developers to learn from. By building a functional app that allows users to answer a series of questions, the app demonstrates the key features of Compose, including data management, user interaction, theming, and compos ability. Through this app, developers can learn how to leverage Compose to create modern, visually appealing apps that provide a great user experience. The app's source code can serve as a valuable reference for developers who want to build similar apps or incorporate Compose into their existing projects.

In addition, the survey app can also be used as a starting point for developers who want to build more complex apps that require user input and data management. By understanding how Compose handles state and data management, developers can create more sophisticated apps that meet users' expectations. Overall, the purpose of the survey app is to demonstrate the power and flexibility of the Android Compose UI toolkit and to help developers learn how to use it effectively.

2. PROBLEM DEFINITION & DESIGN THINKING

2.1 EMPATHY MAP



Template

Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

Share template feedback

Need some inspiration?

See a finished version of this template to kickstart your work.

Open example →

Build empathy

The information you add here should be representative of the observations and research you've done about your users.

Says

What have we heard them say?
What can we magine them saying?

- It is easy to use
- Is is discout
- Is my secure my data

Thinks

What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

- user frinendly
- capabale for all types of users
- Easy understanding of interface
- datas will secure

Give them a name and a portrait to empathize with your persona.

Does

What behavior have we observed?
What can we imagine them doing?

- easy to use of all users
- User friendly interface
- accessabl e for every users

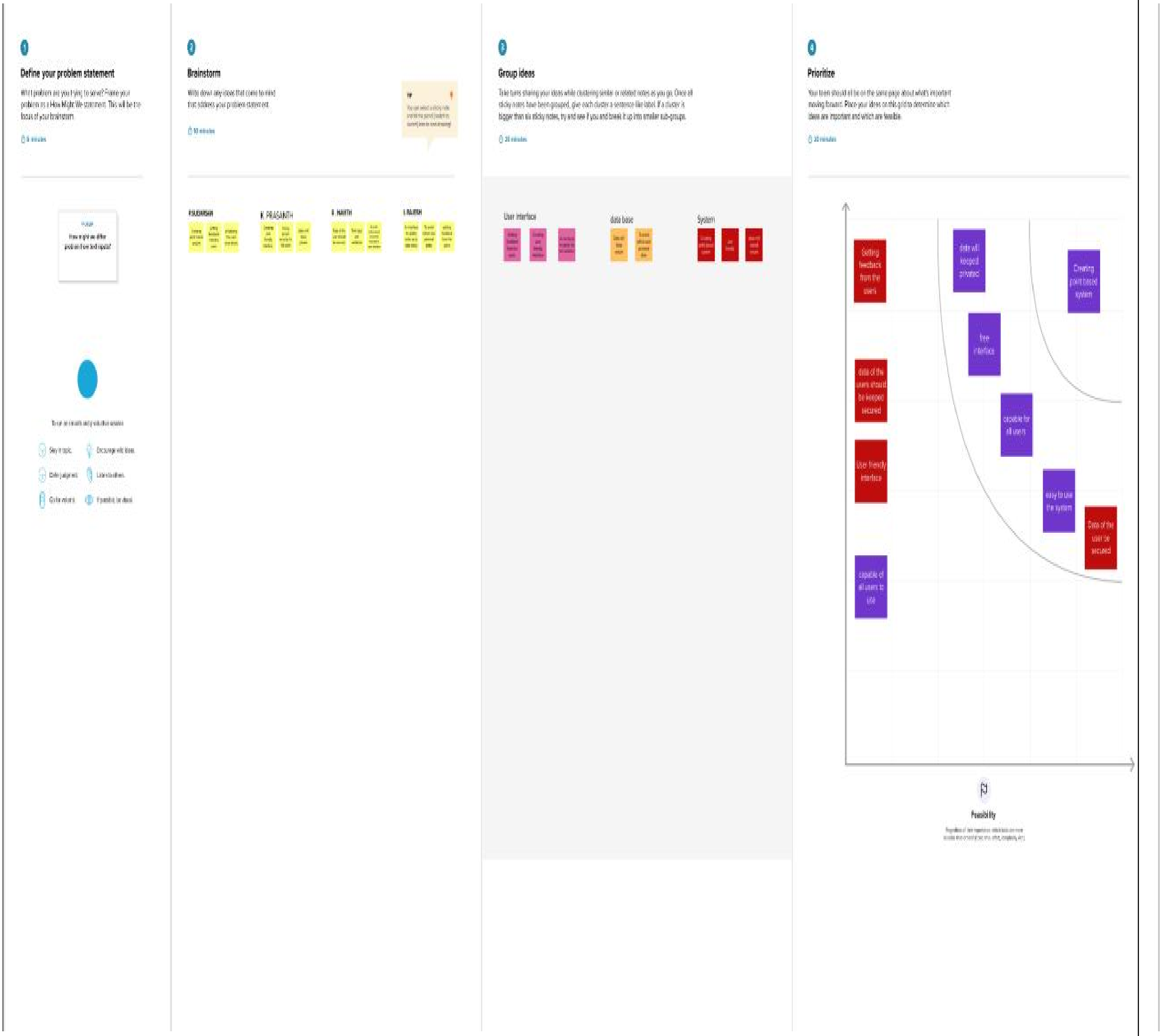
Feels

What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

- in securities
- data stealing from the import text
- not knowing how to use

EMPATHY MAP

2.2 BRAIN STORM



1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

4

Prioritize

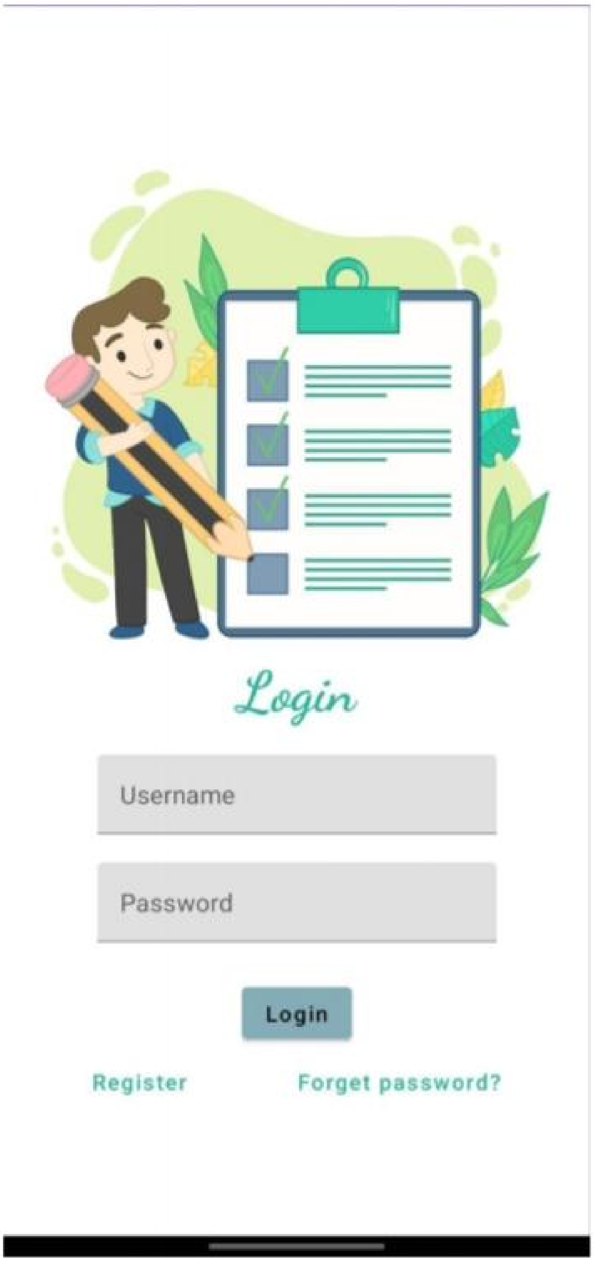
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

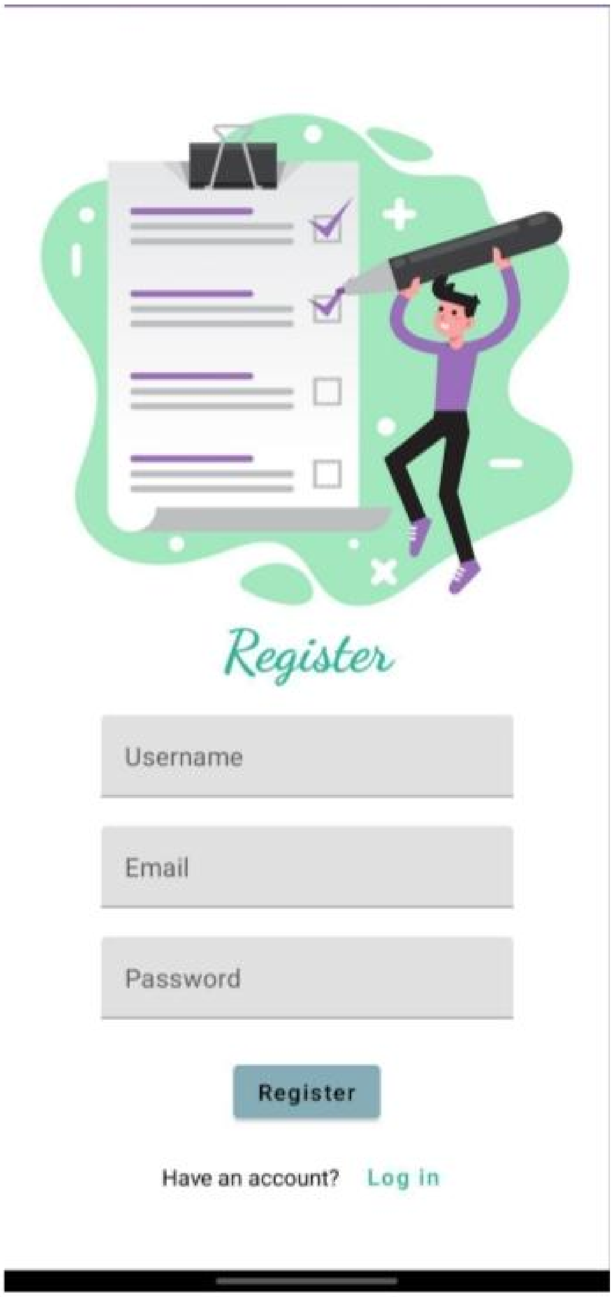
BRAIN STORM

3. RESULTS

OUTPUT



Register Page :



Register

Username

Email

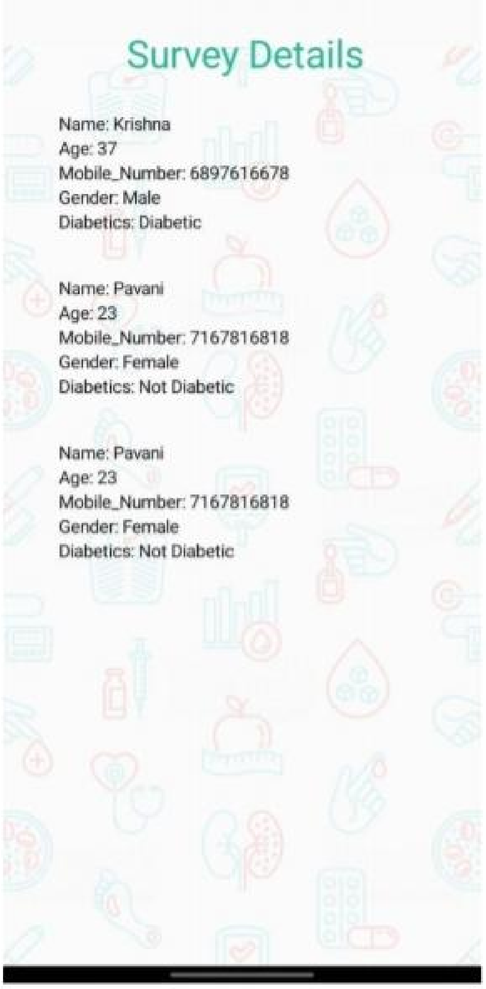
Password

Register

Have an account? Log in

After logging in with Admin Credentials which are hard coded. Password must be "admin" .

Admin page:



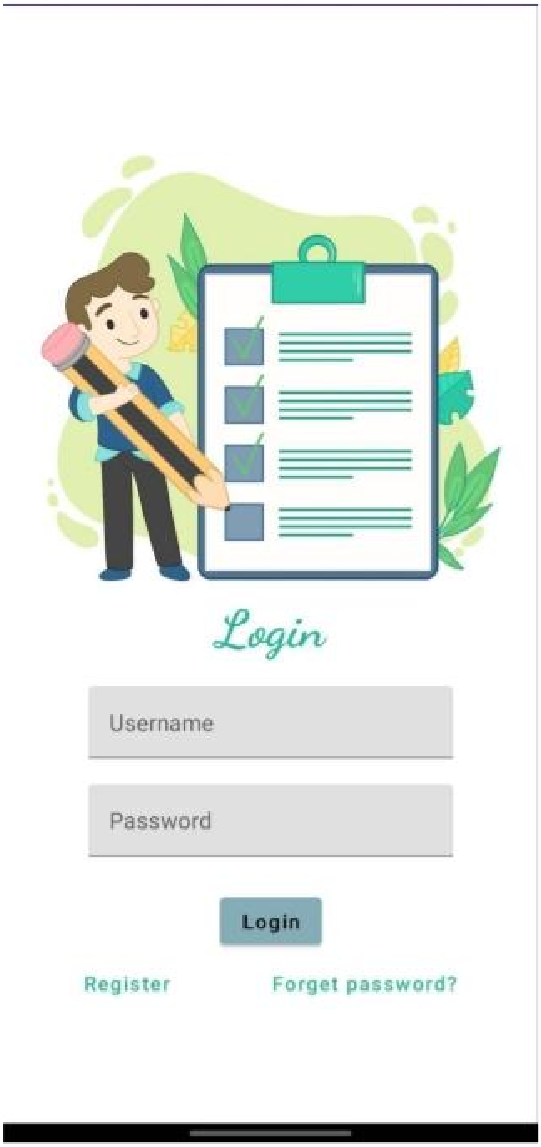
Survey Details

Name: Krishna
Age: 37
Mobile_Number: 6897616678
Gender: Male
Diabetics: Diabetic

Name: Pavani
Age: 23
Mobile_Number: 7167816818
Gender: Female
Diabetics: Not Diabetic

Name: Pavani
Age: 23
Mobile_Number: 7167816818
Gender: Female
Diabetics: Not Diabetic

User Module:
Login Page :

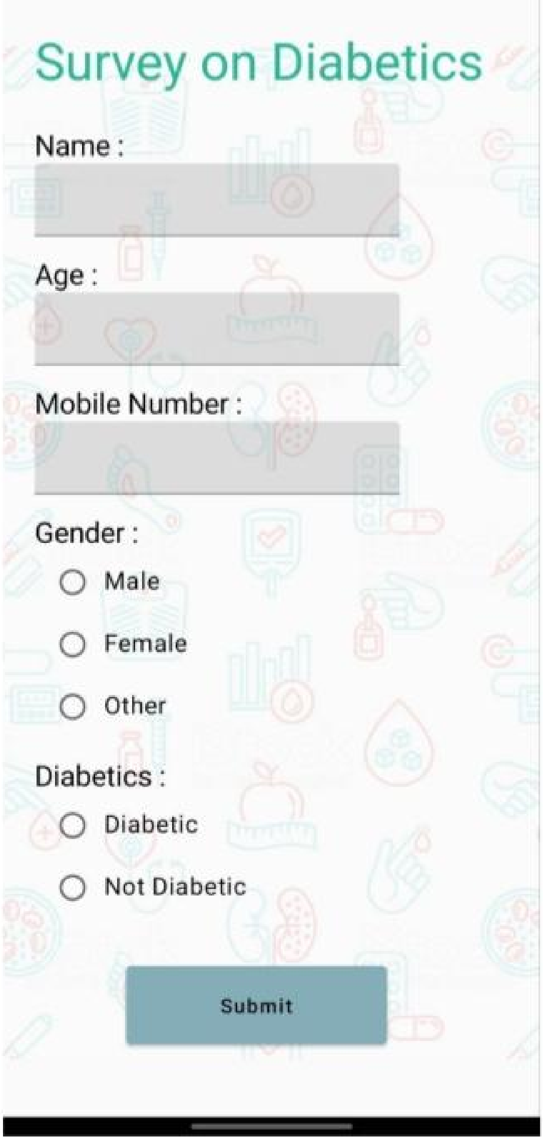


Login

Username

Password

Register Page :



Survey on Diabetics

Name :

Age :

Mobile Number :

Gender :

○ Male

○ Female

○ Other

Diabetics :

○ Diabetic

○ Not Diabetic

4. ADVANTAGES & DISADVANTAGES

4.1 ADVANTAGES

- Cost-efficient. Unlike the written survey, the online version is far cheaper
- Saves time. An online survey is very time efficient.
- Easier with online survey app.
- Accessibility
- Reach and scalability.
- The online survey app is customizable. ...
- More honest. ...
- More objective with an online survey app.

The time it takes for someone to complete an online survey is, on average, about two-thirds shorter than that of other research methods. Since online surveys are being taken automatically, there's no need to wait for paper questionnaires to come back or for a phone interviewer to compile their findings. In addition, with web-based surveys, response time is almost instant.

4.2 DISADVANTAGES

- Inability to Connect With People From Remote Areas. ...
- High Chances of Survey Fraud. ...
- Sampling Issues. ...
- Response Bias. ...
- Survey Fatigue. ...
- Increase in Errors. ...
- A Large Number of Unanswered Questions. ...
- Difficult to Interpret the Sentiments Behind the Answers

Survey fraud is probably the heaviest disadvantage of an online survey. There are people who answer online surveys for the sake of getting the incentive (usually in the form of money) after they have completed the survey, not with a desire to contribute to the advancement of the study. Read more in our blog: Different Types of Survey Bias and How to Avoid Them.

5 APPLICATIONS

In-app surveys are an amazing way to connect with your customers, and therefore a crucial tool for collecting user feedback. These surveys are deployed into apps, enabling you to gather relevant feedback on how your customers view your app as a whole, or how they view the individual features. There's much more to a good survey than simply asking a series of questions. You've also got to take into account your audience's preferences, question types, logic branching, and last but never least, aesthetics. All of these aspects play instrumental roles in helping create surveys that allow you to gain useful, accurate responses, but your ability to create a good survey is only as good as the tools in your arsenal.

It's also worth noting that there are plenty of form apps that offer similar features. Some, like Google Forms, may be included in a suite of software you already use. But most form apps are designed to help you collect specific information, like the names and email addresses of people who'd like to subscribe to your email newsletter. The best survey software, on the other hand, is aimed at large-scale data collection and analysis that uncovers broad trends. Survey apps have become a popular tool for collecting feedback and opinions from a large number of people. With the rise of smartphones and other mobile devices, it has become easier than ever to create and distribute surveys to a wide audience. Survey apps are versatile tools that can be used in various industries and fields to gather valuable insights and data. Here are some areas where survey apps are applied.

Market Research: Survey apps are widely used in market research to gather data on customer preferences, behaviour, and demographics. Companies use this information to improve their products and services and to better understand their customers' needs.

Healthcare: Survey apps are used in healthcare to collect patient feedback on the quality of care they receive. This data can be used to identify areas for improvement and to ensure that patients are receiving the best possible care.

Education: Survey apps are used in education to gather feedback from students and parents on the quality of education they are receiving. This data can be used to identify areas for improvement and to develop better teaching strategies.

6. CONCLUSION

In conclusion, survey apps are versatile tools that can be used in various industries and fields to gather valuable insights and data. With the rise of smartphones and other mobile devices, survey apps have become more accessible and easier to use than ever before. Whether you are a business owner, a healthcare provider, a non-profit organization, or a government agency, survey apps can help you gather the information you need to make informed decisions and improve your services.

Survey apps have become increasingly popular due to their ease of use, accessibility, and ability to collect large amounts of data quickly and efficiently. These apps are designed to help organizations of all types gather information and insights from their target audience or stakeholders. With survey apps, it is possible to design, distribute, and analyze surveys in real-time, providing valuable data that can inform decision-making and help organizations improve their services

One of the key benefits of survey apps is their ability to reach a large number of people in a short amount of time. With traditional paper-based surveys, it can be challenging to collect responses from a wide audience, as it requires time, resources, and physical distribution of the survey. However, with survey apps, it is possible to distribute surveys electronically, which can reach a vast number of people quickly, regardless of their location. This makes survey apps an ideal tool for organizations that need to collect data from a geographically dispersed audience.

Survey apps are also versatile, allowing organizations to create and distribute surveys that meet their specific needs. Whether it is a customer satisfaction survey, an employee engagement survey, or a market research survey, survey apps can be customized to fit the specific needs of the organization. The app's features can include the ability to add images or videos, skip logic, and a variety of question types, such as multiple-choice, open-ended, and ranking questions.

7.FUTURE SCOPE

The use of survey apps has been on the rise in recent years, and this trend is expected to continue in the future. As technology continues to evolve and become more advanced, survey apps are likely to become even more useful and user-friendly. Here are some of the trends that we can expect to see in the future of survey apps:

Increased personalization: As more and more data becomes available, survey apps are likely to become more personalized. This means that surveys will be tailored to the individual, taking into account their preferences and behavior. This will make surveys more engaging and increase response rates.

Integration with other apps: Survey apps are likely to become more integrated with other apps. This will make it easier to collect data from different sources and analyze it more effectively. For example, a survey app could be integrated with a social media app to collect data on user behavior.

Use of AI and machine learning: AI and machine learning are likely to play a big role in the future of survey apps. These technologies will help to analyze data more effectively and provide insights that were not possible before. For example, machine learning algorithms could be used to predict customer behavior based on survey responses.

Improved data security: As more sensitive data is collected through survey apps, it is important to ensure that this data is kept secure. In the future, survey apps are likely to have improved security features to protect user data.

Gamification: Gamification is likely to become more prevalent in survey apps. This means that surveys will be designed to be more fun and engaging, using techniques like rewards and points systems to encourage participation.

In conclusion, the future of survey apps looks bright. With increased personalization, integration with other apps, the use of AI and machine learning, improved data security, and gamification, survey apps are likely to become even more useful and user-friendly. As technology continues to evolve, we can expect to see even more exciting developments in the world of survey apps

8 . APPENDIX

Login

```
package com.example.surveyapplication

import android.content.Context

import android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity


import androidx.activity.compose.setContent


import androidx.compose.foundation.Image


import androidx.compose.foundation.background


import androidx.compose.foundation.layout.*


import androidx.compose.material.*


import androidx.compose.runtime.*


import androidx.compose.ui.Alignment


import androidx.compose.ui.Modifier


import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.surveyapplication.ui.theme.SurveyApplicationTheme


class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        databaseHelper = UserDatabaseHelper(this)

        setContent {



                LoginScreen(this, databaseHelper)



        }

    }

}



@Composable

fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {



    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }


    Column(
        modifier = Modifier.fillMaxSize().background(Color.White),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {


        Image(painterResource(id = R.drawable.survey_login), contentDescription = "")


        Text(
            fontSize = 36.sp,
        fontWeight = FontWeight.ExtraBold,

        fontFamily = FontFamily.Cursive,

        color = Color(0xFF25b897),

        text = "Login"

    )

    Spacer(modifier = Modifier.height(10.dp))



    TextField(

        value = username,

        onValueChange = { username = it },

        label = { Text("Username") },

        modifier = Modifier

            .padding(10.dp)

            .width(280.dp)
```

```
)



TextField(
    value = password,
    onValueChange = { password = it },
    label = { Text("Password") },
    visualTransformation = PasswordVisualTransformation(),
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp)
)



if (error.isNotEmpty()) {
    Text(
        text = error,
        color = MaterialTheme.colors.error,
        modifier = Modifier.padding(vertical = 16.dp)
    )
}


Button(
    onClick = {
        if (username.isNotEmpty() && password.isNotEmpty()) {
            val user = databaseHelper.getUserByUsername(username)
            if (user != null && user.password == password) {
                error = "Successfully log in"
                context.startActivity(
        Intent(
            context,
            MainActivity::class.java
        )
    )
    //onLoginSuccess()
}
if (user != null && user.password == "admin") {
    error = "Successfully log in"
    context.startActivity(
        Intent(
            context,
            AdminActivity::class.java
```

```
                    )
                )
            }
            else {
                error =  "Invalid username or password"
            }


        } else {
            error = "Please fill all fields"
        }
    },
    colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFF84adb8)),
    modifier = Modifier.padding(top = 16.dp)
) {
    Text(text = "Login")

}

Row {

    TextButton(onClick = {context.startActivity(

        Intent(

            context,

            RegisterActivity::class.java

        )

    )}

    )

    { Text(color = Color(0xFF25b897),text = "Register") }

    TextButton(onClick = {

    })
```

```
            {
                Spacer(modifier = Modifier.width(60.dp))
                Text(color = Color(0xFF25b897),text = "Forget password?")
            }
        }
    }
}
private fun startMainPage(context: Context) {
    val intent = Intent(context, MainActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

REGISTER

```
package com.example.surveyapplication

import android.content.Contex

import android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.Image


import androidx.compose.foundation.background


import androidx.compose.foundation.layout.*


import androidx.compose.material.*


import androidx.compose.runtime.*


import androidx.compose.ui.Alignment


import androidx.compose.ui.Modifier


import androidx.compose.ui.graphics.Color


import androidx.compose.ui.layout.ContentScale


import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import androidx.core.content.ContextCompat

import com.example.surveyapplication.ui.theme.SurveyApplicationTheme



class RegisterActivity : ComponentActivity() {

    private lateinit var databaseHelper: UserDatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        databaseHelper = UserDatabaseHelper(this)
        setContent {

            RegistrationScreen(this,databaseHelper)

        }

    }

}

@Composable

fun RegistrationScreen(context: Context, databaseHelper: UserDatabaseHelper) {

    var username by remember { mutableStateOf("") }

    var password by remember { mutableStateOf("") }

    var email by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }


    Column(
        modifier = Modifier.fillMaxSize().background(Color.White),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {

        Image(painterResource(id = R.drawable.survey_signup), contentDescription = "")

        Text(
            fontSize = 36.sp,
            fontWeight = FontWeight.ExtraBold,
        fontFamily = FontFamily.Cursive,

        color = Color(0xFF25b897),

        text = "Register"

    )



    Spacer(modifier = Modifier.height(10.dp))

    TextField(

        value = username,

        onValueChange = { username = it },

        label = { Text("Username") },

        modifier = Modifier

            .padding(10.dp)

            .width(280.dp)
```

```
)

TextField(
    value = email,
    onValueChange = { email = it },
    label = { Text("Email") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp)
)

TextField(
    value = password,
    onValueChange = { password = it },

    label = { Text("Password") },

    visualTransformation = PasswordVisualTransformation(),

    modifier = Modifier

        .padding(10.dp)

        .width(280.dp)

)




if (error.isNotEmpty()) {

    Text(

        text = error,

        color = MaterialTheme.colors.error,

        modifier = Modifier.padding(vertical = 16.dp)
    )
}

Button(
    onClick = {
        if (username.isNotEmpty() && password.isNotEmpty() && email.isNotEmpty()) {
            val user = User(
                id = null,
                firstName = username,
                lastName = null,
                email = email,
                password = password
            )
            databaseHelper.insertUser(user)

            error = "User registered successfully"

            // Start LoginActivity using the current context

            context.startActivity(

                Intent(

                    context,

                    LoginActivity::class.java

                )

            )



        } else {

            error = "Please fill all fields"

        }

    },
    colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFF84adb8)),

    modifier = Modifier.padding(top = 16.dp),



) {

    Text(text = "Register")

}

Spacer(modifier = Modifier.width(10.dp))

Spacer(modifier = Modifier.height(10.dp))



Row() {

    Text(

        modifier = Modifier.padding(top = 14.dp), text = "Have an account?"

    )
    TextButton(onClick = {

        context.startActivity(

            Intent(

                context,

                LoginActivity::class.java

            )

        )

    })



    {

        Spacer(modifier = Modifier.width(10.dp))

        Text( color = Color(0xFF25b897),text = "Log in")

    }

}
```

```
    }

}

private fun startLoginActivity(context: Context) {

    val intent = Intent(context, LoginActivity::class.java)

    ContextCompat.startActivity(context, intent, null)

}
```

Survey database

```
package com.example.surveyapplication

import android.content.Context

import androidx.room.Database

import androidx.room.Room

import androidx.room.RoomDatabase



@Database(entities = [Survey::class], version = 1)
```

```
abstract class SurveyDatabase : RoomDatabase() {

    abstract fun surveyDao(): SurveyDao

    companion object {

        @Volatile
        private var instance: SurveyDatabase? = null

        fun getDatabase(context: Context): SurveyDatabase {
            return instance ?: synchronized(this) {
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    SurveyDatabase::class.java,
                    "user_database"

                ).build()

                instance = newInstance

                newInstance

            }

        }

    }

}
```