A Sleep Tracking App For A Better Night

ABSTRACT:

A project that demonstrates the use of Android Jetpack Compose to build a UI for a sleep tracking app. The app allows users to track their sleep. With the "Sleep Tracker" app, you can assess the quality of sleep they have had in a day. It has been time and again proven that a good quality sleep is pretty essential for effective functioning of both mind and body.

INTRODUCTION:

"Sleep Tracker" application enables you to start the timer when they are in the bed and about to fall asleep. The timer will keep running in the background until it is stopped, whenever the user wakes up. Based on the sleep experience, you can rate your sleep quality. Finally, the app will display an analysis of the kind of sleep, you had the previous night.

INTRODUCTION TO KOTLIN:

The Kotlin programming language is a modern language that gives you more power for your everyday tasks. Kotlin is concise, safe, pragmatic, and focused on interoperability with Java code. It can be used almost everywhere Java is used today for server-side development, Android apps, and much more.

BUILDING APPLICATIONS UI:

Material design is a comprehensive guide for visual, motion, and interaction design across platforms and devices. To use sleep tracking app in your Android app, follow the guidelines defined in the sleep tracking app specification and use the new components and styles available in the app.

Project Workflow:

1. Users register into the application.
2. After registration, user logins into the application.
3. User enters into the main page.
4. User can track the sleep timing and he record the time.

BUILT YOUR APP:

Android is rising in the ranks as one of the most popular operating system to develop for in this era of mobile development. The OS gives developers a quick time to market (TTM), an open source platform that can accommodate changing business requirements, and the ability to deploy on multiple platform.

KOTLIN PROGRAMMING IN NULLABILITY:

A type is only nullable if you explicitly let it hold null. As the error message says, the String data type is a non-nullable type, so you can't reassign the variable to null. To declare nullable variables in Kotlin, you need to add a operator to the end of the type.

KOTLIN FUNDAMENTALS IN LAMBDA EXPRESSION:

Lambda is a function which has no name. Lambda is defined with a curly braces **{}** which takes *variable as a parameter* (if any) and body of function. The *body of function* is written after variable (if any) followed by -> operator.

SYNTAX OF LAMBDA

{ variable -> body_of_function}

DICE APP:

In this section you will create an interactive Dice Roller app that lets users tap a Button composable to roll a dice. The outcome of the roll is shown with an Image. You use Jetpack Compose with Kotlin to build your app layout and then write business logic to handle what happens when the Button composable is tapped.

TIP CALCULATOR:

In this Section, you explore how to use and think about state when you use Compose. To do this, you build a tip-calculator app called Tip Time with built-in Compose UI elements like TEXTFIELD, TEXT and SPACER.

ADVANTAGES:

1.Helps you better understand your sleep patterns in order to improve your sleep patterns you need to understand them.

2.Tracks your different sleep phases there are generally four sleep phases: awake, light, deep and REM.

3. Captures possible sleep interruptions and disruptions.

4.Gently wakes you up at the ideal item.

5.Convenient compared to wearing a sleep tracker device.

FUTURE SCOPE:

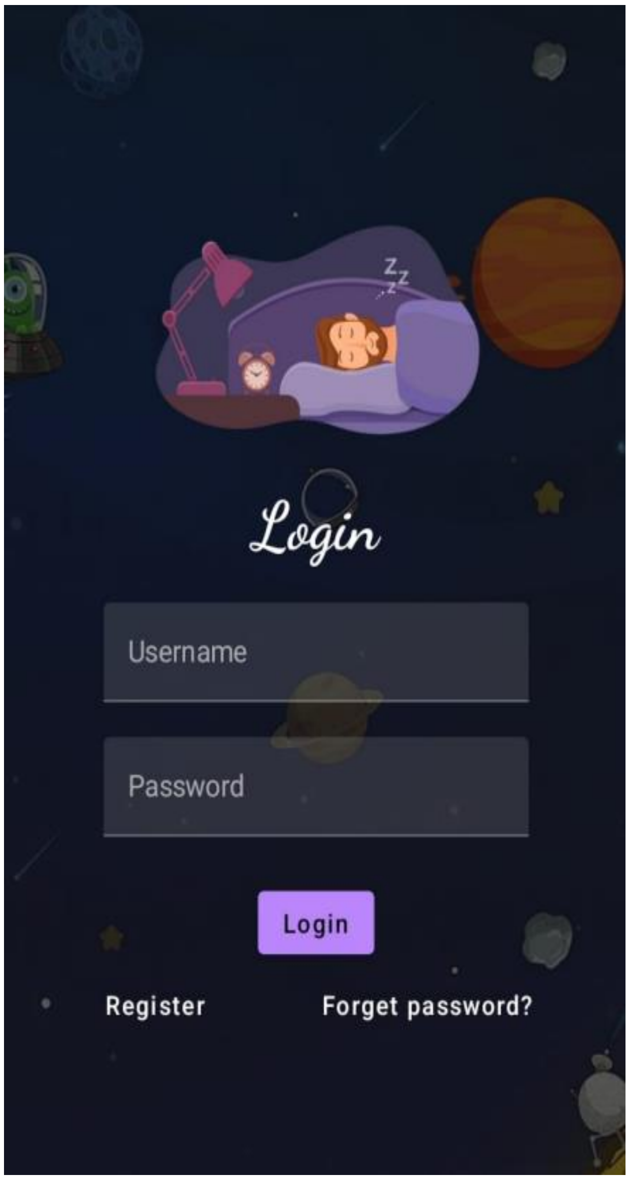
The future scope of sleep tracking apps for Android is quite promising. As technology advances, sleep tracking apps are likely to become more accurate and sophisticated, potentially incorporating features such as wearable sensors, machine learning algorithms and personalized recommendations based on individuals sleep patterns and habits

CODE:

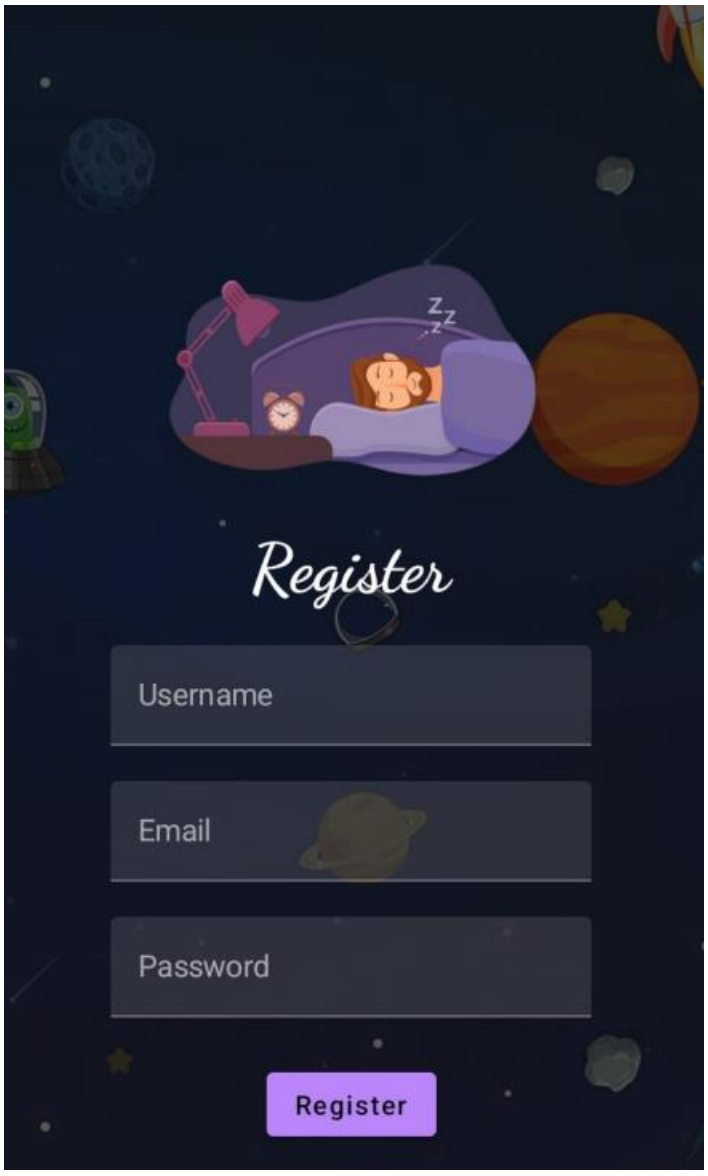
```
class
TrackActivity:

ComponentActivity()
{


                      private lateinit var databaseHelper: TimeLogDatabaseHelper


                          override fun onCreate(savedInstanceState: Bundle?) {

                              super.onCreate(savedInstanceState)


                              databaseHelper = TimeLogDatabaseHelper(this)

                              setContent {

                                  ProjectOneTheme {

                                      // A surface container using the 'background'
                      color from the theme

                                      Surface(

                                          modifier = Modifier.fillMaxSize(),

                                          color = MaterialTheme.colors.background

                                      ) {

                                          //ListListScopeSample(timeLogs)


                                          val data=databaseHelper.getTimeLogs();

                                          Log.d("Sandeep" ,data.toString())

                                          val timeLogs = databaseHelper.getTimeLogs()

                                          ListListScopeSample(timeLogs)

                                      }

                                  }

                              }

                          }

                      }
```

```
@Composable
fun ListListScopeSample(timeLogs:
List<TimeLogDatabaseHelper.TimeLog>) {
    val imageModifier = Modifier
    Image(
        painterResource(id = R.drawable.sleeptracking),
        contentScale = ContentScale.FillHeight,
        contentDescription = "",
        modifier = imageModifier
            .alpha(0.3F),
    )

    Text(text = "Sleep Tracking", modifier = Modifier.padding(top
= 16.dp, start = 106.dp ), color = Color.White, fontSize = 24.sp)
    Spacer(modifier = Modifier.height(30.dp))
    LazyRow(
        modifier = Modifier
            .fillMaxSize()
            .padding(top = 56.dp),

        horizontalArrangement = Arrangement.SpaceBetween
    ){
        item {

            LazyColumn {
                items(timeLogs) { timeLog ->
                    Column(modifier = Modifier.padding(16.dp)) {
                        //Text("ID: ${timeLog.id}")

                        Text("Start time:
${formatDateTime(timeLog.startTime)}")

                        Text("End time: ${timeLog.endTime?.let {
formatDateTime(it) }}")

                    }

                }

            }

        }


        }

    }


private fun formatDateTime(timestamp: Long): String {

    val dateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss",
Locale.getDefault())

    return dateFormat.format(Date(timestamp))

}
```

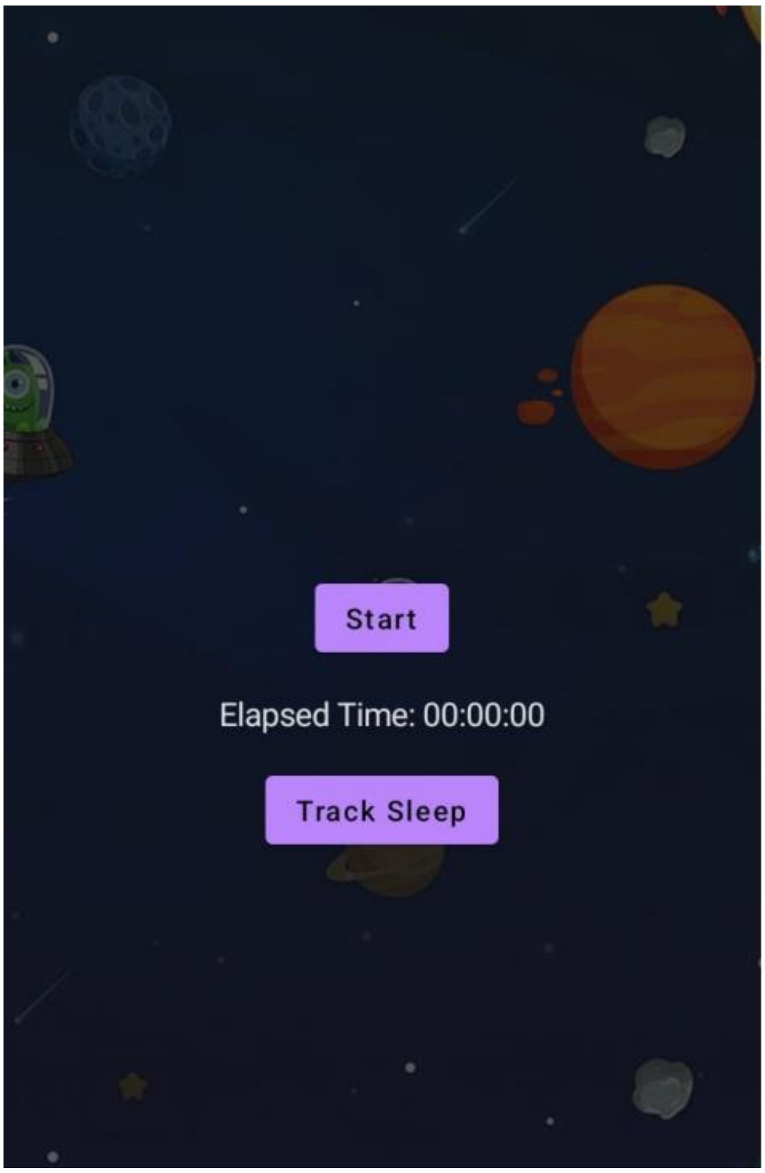
LOGIN PAGE:



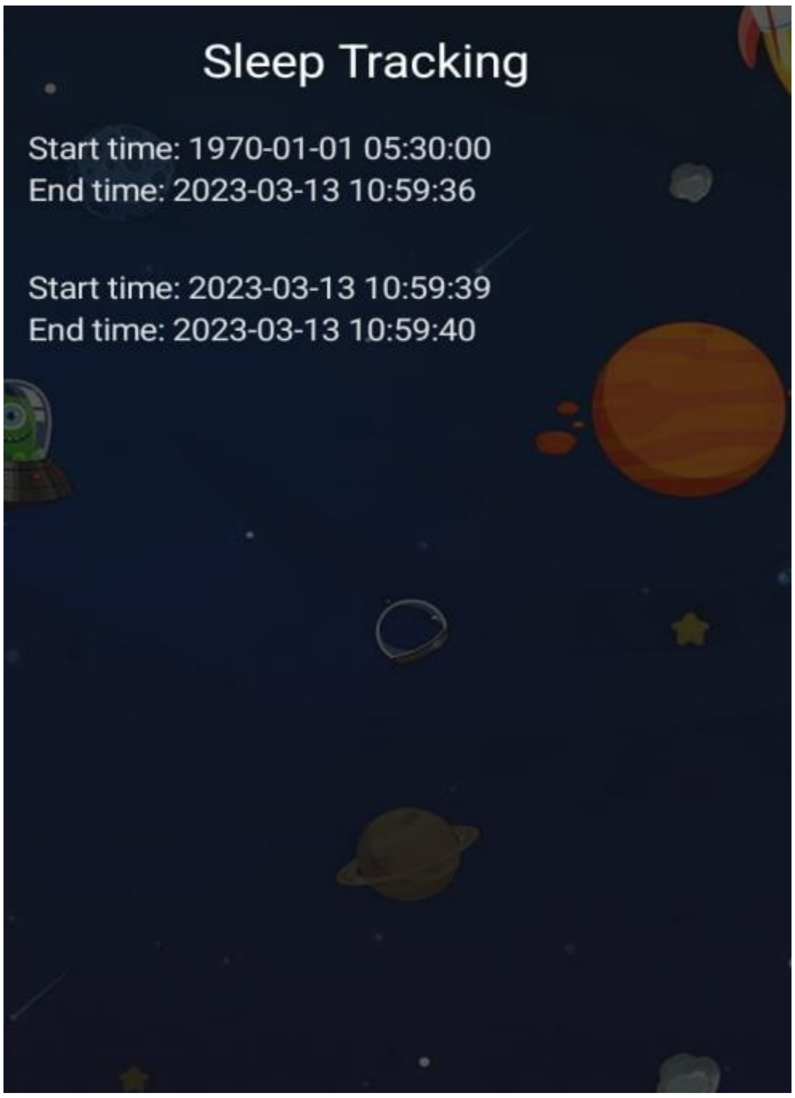
REGISTRATION PAGE:



MAIN PAGE:



SLEEP TARCK PAGE:



Sleep Tracking

Start time: 1970-01-01 05:30:00
End time: 2023-03-13 10:59:36

Start time: 2023-03-13 10:59:39
End time: 2023-03-13 10:59:40