SLEEP TRACKING APP

Team Members Name	NM ID
SUNIL A B	ED8A92F5B230F1A938E4185D43966080
KAMALAKKANNAN P	C2ED0E44C4F9722BB894D96853026081
VIKNESH S	87FD63EA7F035ECA50641D9044D4BF73
VIJAY M	4FAE4D1C0C62F8EA2474C7338E163F81

Project Description:

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.

“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.

Installation of Android Studio:

1. Prerequisites

Ensure your system meets the requirements:

- Windows: 8 GB RAM, 4 GB free disk space, 64-bit Windows 10/11
- macOS: macOS 10.14+, 8 GB RAM, 4 GB free space
- Linux: Ubuntu 20.04+, 8 GB RAM, 4 GB free space

2. Download and Install

Windows:

- Download .exe from [Android Studio](https://developer.android.com/studio).
- Run the installer and choose Standard setup.
- Launch Android Studio and let it download necessary components.

macOS:

- Download .dmg from [Android Studio](https://developer.android.com/studio).
- Drag Android Studio to Applications.
- Launch and follow the setup wizard.

Linux:

- Download .zip from [Android Studio](https://developer.android.com/studio).
- Extract it, then run studio.sh from the bin folder.

3. Set Up SDK

- After installation, Android Studio will prompt you to download the Android SDK and other tools.
- Use the SDK Manager to install required packages and AVD Manager for emulators.

4. Start Development

- Create a new project in Android Studio and choose a template (e.g., Empty Activity).

- Run the app on an emulator or connected Android device.

Steps to be followed:

1. Download the Android Studio installer from the official website: developer.android.com/studio.

2. Run the installer and follow the setup wizard to install Android Studio on your computer.

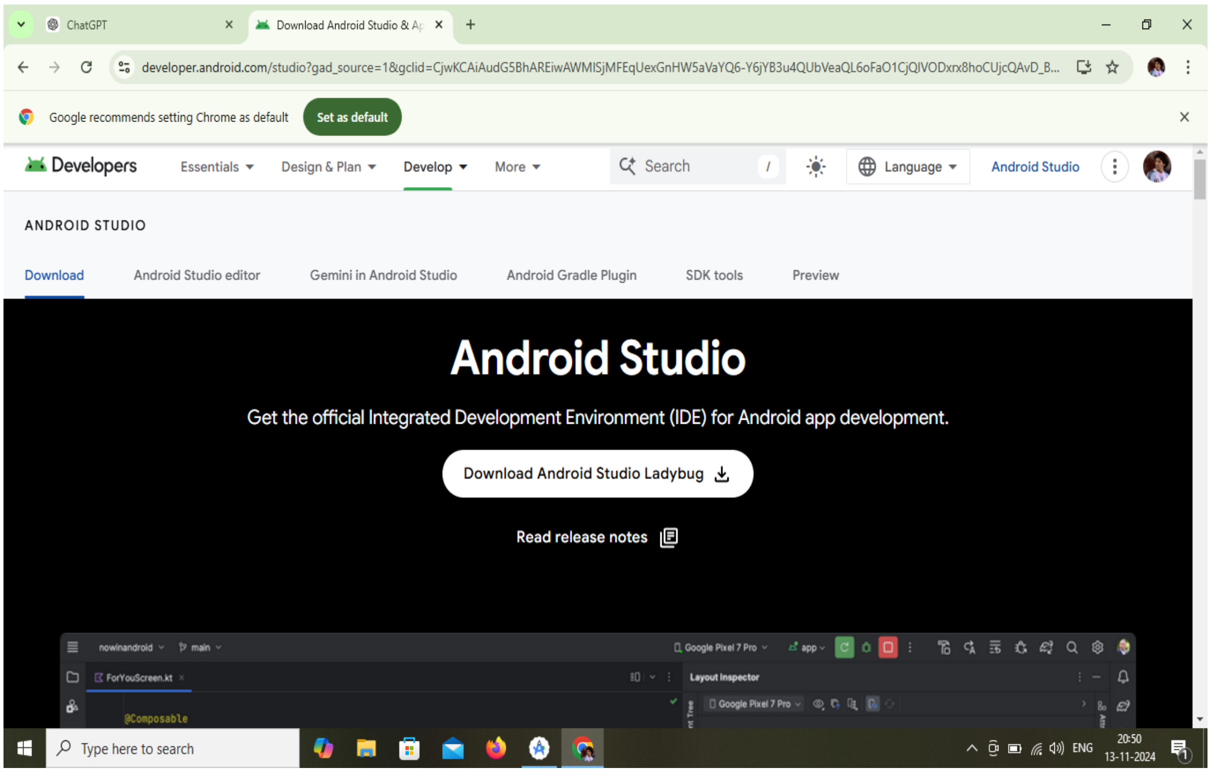
3. Launch Android Studio after installation is complete.

4. On the first run, select "Standard" setup for a quick installation of necessary components.

5. Download Android SDK and other required tools during the setup process.

6. Once everything is installed, start a new project to begin development

Screenshots:

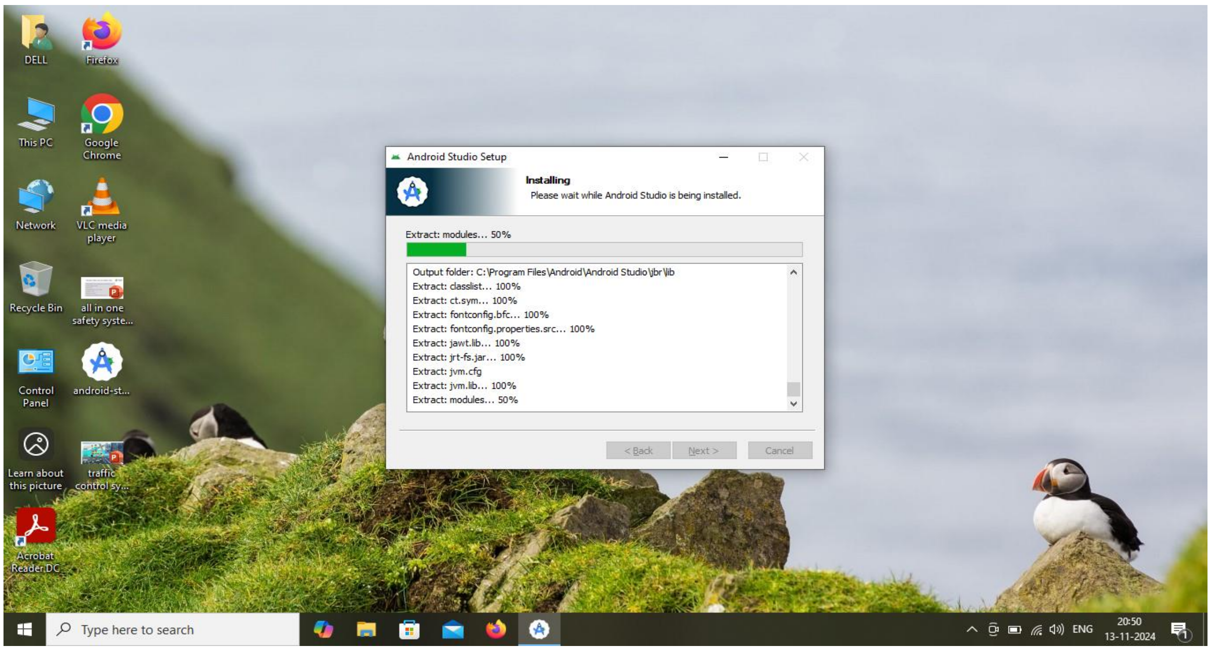

developer.android.com/studio?gad_source=1&gclid=CjwKCAiAudG5BhAREiwAWMlSjMFEqUexGnHW5aVaYQ6-Y6jYB3u4QUbVeaQL6oFaO1CjQlVODxrx8hoCUjcQAvD_B...

Google recommends setting Chrome as default

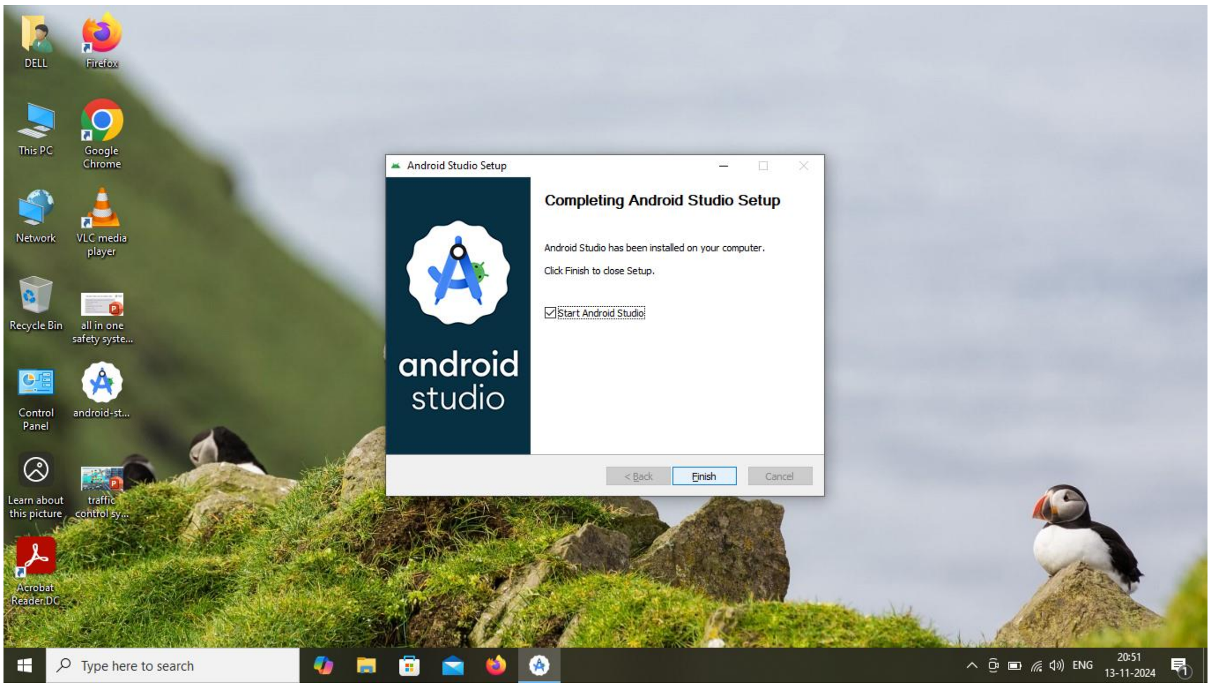
ANDROID STUDIO

Android Studio

Get the official Integrated Development Environment (IDE) for Android app development.



```
Extract: modules... 50%
Output folder: C:\Program Files\Android\Android Studio\jbr\lib
Extract: classlist... 100%
Extract: ct.sym... 100%
Extract: fontconfig.bfc... 100%
Extract: fontconfig.properties.src... 100%
Extract: jawt.lib... 100%
Extract: jrt-fs.jar... 100%
Extract: jvm.cfg
Extract: jvm.lib... 100%
Extract: modules... 50%
```



Completing Android Studio Setup

Android Studio has been installed on your computer.

Click Finish to close Setup.

[✓ Start Android Studio]

Creating a New Project in Android Studio:

1. Open Android Studio

- Launch Android Studio. If you're opening it for the first time, it may take a few moments to initialize.

2. Start a New Project

- On the Welcome Screen, click "Start a new Android Studio project".
  - If you have an existing project open, go to File > New > New Project.

3. Choose a Template

- Android Studio will prompt you to choose a project template. You can choose from various options, like:
  - Empty Activity (for a blank app).
  - Basic Activity (with a toolbar and floating action button).
  - Navigation Drawer Activity (with a side navigation menu).
  - Fullscreen Activity (for apps that use the whole screen).

Choose "Empty Activity" for a simple start.

4. Configure Your Project

- Name: Enter your app's name (e.g., "MyFirstApp").
- Package Name: A unique identifier (usually in reverse domain format, like `com.example.myfirstapp`).
- Save Location: Choose a location on your computer to store the project.
- Language: Choose between Java or Kotlin. (Kotlin is now the preferred language for Android development.)

- Minimum API Level: Select the lowest version of Android your app will support. Android Studio recommends an API level based on your target audience.

5. Finish

- Click Finish to create your project. Android Studio will generate the necessary files and open your new project.

6. Start Coding

- After the project is created, you'll see:
  - MainActivity.java/Kotlin: The main entry point for your app.
  - activity_main.xml: The layout file for the main activity (UI design).

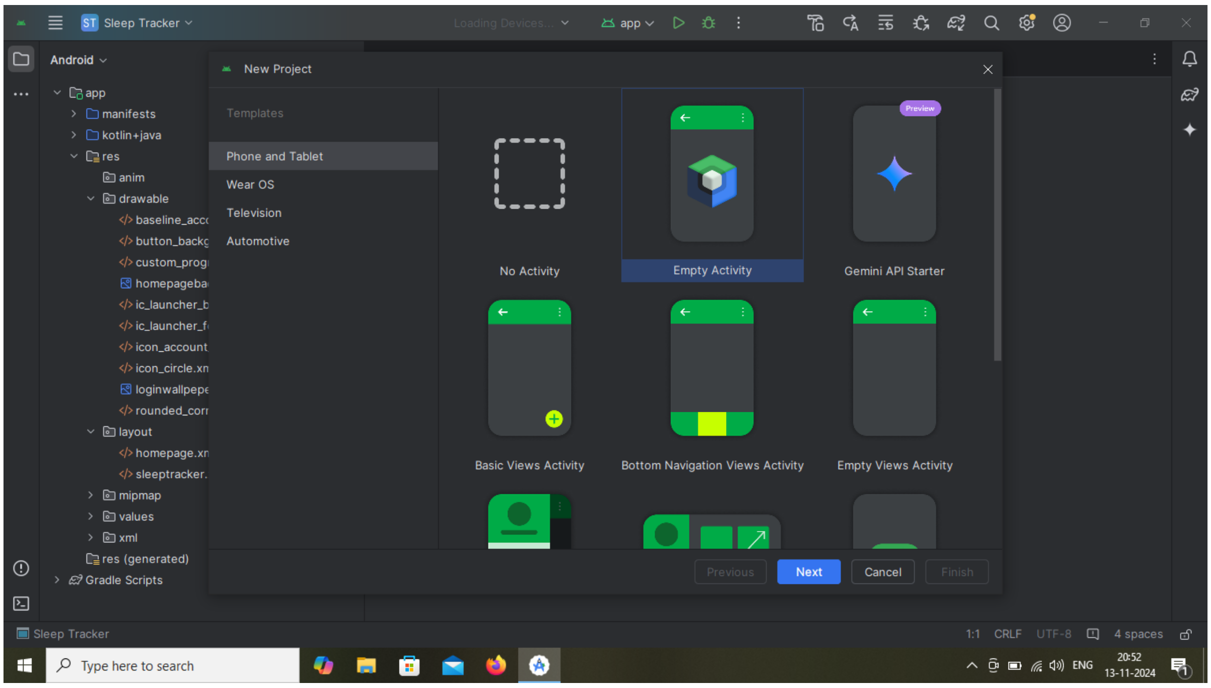
You can now edit the code and design for your new Android app.

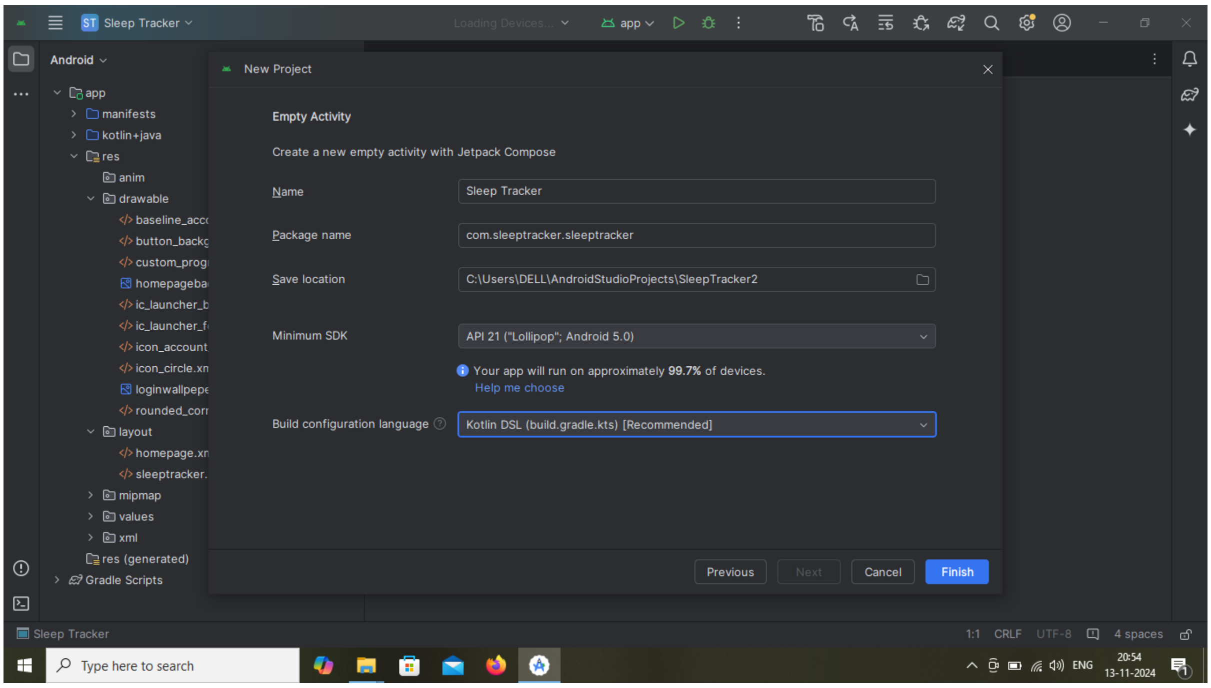
7. Run Your App

- To test your app, you can either:
  - Use an Android Emulator by creating a virtual device (AVD).
  - Connect an Android device via USB and enable developer mode.

Click the green Run button (a play icon) in the top toolbar to build and run your app.

Screenshots:





Empty Activity

Create a new empty activity with Jetpack Compose

Name: Sleep Tracker

Package name: com.sleeptracker.sleeptracker

Save location: C:\Users\DELL\AndroidStudioProjects\SleepTracker2

Minimum SDK: API 21 ("Lollipop"; Android 5.0)

Your app will run on approximately **99.7%** of devices.
Help me choose

Build configuration language: Kotlin DSL (build.gradle.kts) [Recommended]

Creating The Database Classes:

1. User Registration / Profile Setup

Before tracking sleep data, the user must register and set up their profile.

Steps:

- Step 1: User creates an account (sign up).
- Step 2: User inputs preferences (e.g., sleep goals, ideal sleep duration).
- Step 3: User sets up wearable or app integrations (if applicable).

Key Data Points for this Step:

- User ID
- User Preferences (e.g., desired sleep duration, sleep quality score)
- Sleep Environment Preferences (e.g., quiet room, no light, etc.)

2. Tracking a Sleep Session

This is where the sleep data is logged—either automatically by a wearable or manually entered by the user.

Steps:

- Step 1: User starts a sleep session (either manually or automatically).
- Step 2: The app records the start time of sleep.
- Step 3: The app records when the user wakes up, either via alarm or after detecting wakefulness.
- Step 4: The app tracks various sleep metrics: stages of sleep, sleep quality, interruptions, movement, heart rate (if applicable).

Key Data Points for this Step:

- Start Time and End Time of the session.
- Sleep Duration (calculated based on start and end times).
- Sleep Quality (could be self-reported or derived from other metrics).

- Sleep Stages (Light, Deep, REM, Awake).
- Sleep Interruptions (number of times the user wakes up during the night).
- Movement/Restlessness (number of movements detected).
- Heart Rate (optional, from wearables).

3. Recording Sleep Stages

Sleep is typically tracked in stages (Light Sleep, Deep Sleep, REM, and Awake). This step captures that data.

Steps:

- Step 1: The app or wearable detects and records when the user is in each sleep stage.
- Step 2: Each stage's duration is tracked, noting when the user transitions between stages.
- Step 3: The app calculates and aggregates the total time spent in each stage.

Key Data Points for this Step:

- Sleep Stages:
    - Light Sleep
    - Deep Sleep
    - REM Sleep
    - Awake Time
- Duration of Each Stage (how long the user spends in each sleep stage).

4. Data Review and Insights

After the sleep session ends, the app provides insights and metrics based on the sleep data collected.

Steps:

- Step 1: User reviews their sleep session summary (e.g., total sleep duration, quality score, number of interruptions).

- Step 2: App offers insights based on the data (e.g., “You spent too little time in Deep Sleep tonight” or “Your sleep quality improved 10% compared to last night”).
- Step 3: If the user has set sleep goals, the app compares their sleep against these goals.

Key Data Points for this Step:

- Sleep Quality (derived score from different metrics like sleep duration and stages).
- Sleep Duration (how much sleep the user got).
- Sleep Interruptions (how often the user woke up during the night).
- Trends or Suggestions based on the current session (e.g., improvement or decline in sleep stages, time spent in each phase).

5. Aggregating and Analyzing Sleep Data Over Time

This step aggregates sleep data from multiple sessions and provides long-term insights.

Steps:

- Step 1: The app gathers data from previous sleep sessions (over days, weeks, or months).
- Step 2: The app calculates averages (e.g., average sleep duration, average sleep quality).
- Step 3: The app provides a trend or pattern over time, showing if the user’s sleep is improving or declining.
- Step 4: The app suggests adjustments based on trends (e.g., “You’ve been consistently getting less sleep on weekdays. Try setting an earlier bedtime.”).

Key Data Points for this Step:

- Average Sleep Duration (over a week, month, or year).
- Average Sleep Quality (from a quality score or other derived metrics).

- Sleep Consistency (how consistent the user is with their sleep schedule).
- Trends (e.g., “You’re getting 7 hours of sleep per night on average, but your sleep quality has dropped recently”).

6. User Goal Setting and Progress

This step helps users set and track sleep goals and helps them stay motivated to meet their objectives.

Steps:

- Step 1: The user sets a sleep goal (e.g., “I want to get 8 hours of sleep every night” or “I want to improve my sleep quality to 8/10”).
- Step 2: The app tracks the user's progress toward their goal.
- Step 3: The app provides feedback or motivation (e.g., “You’ve hit your sleep goal for the past 5 days in a row!”).

Key Data Points for this Step:

- Sleep Goal (duration or quality goal the user wants to achieve).
- Progress Toward Goal (how close the user is to meeting their goal).
- Achievement or Milestones (e.g., hitting the goal consistently for a week).

7. Notifications and Reminders (Optional)

This feature keeps the user on track with their sleep habits by sending reminders.

Steps:

- Step 1: The app sends reminders to go to bed at a certain time (based on sleep goals or ideal bedtimes).
- Step 2: The app may remind the user of their sleep goals and progress.

- Step 3: The app sends motivational messages or tips (e.g., “Try turning off screens 30 minutes before bed to improve sleep quality”).

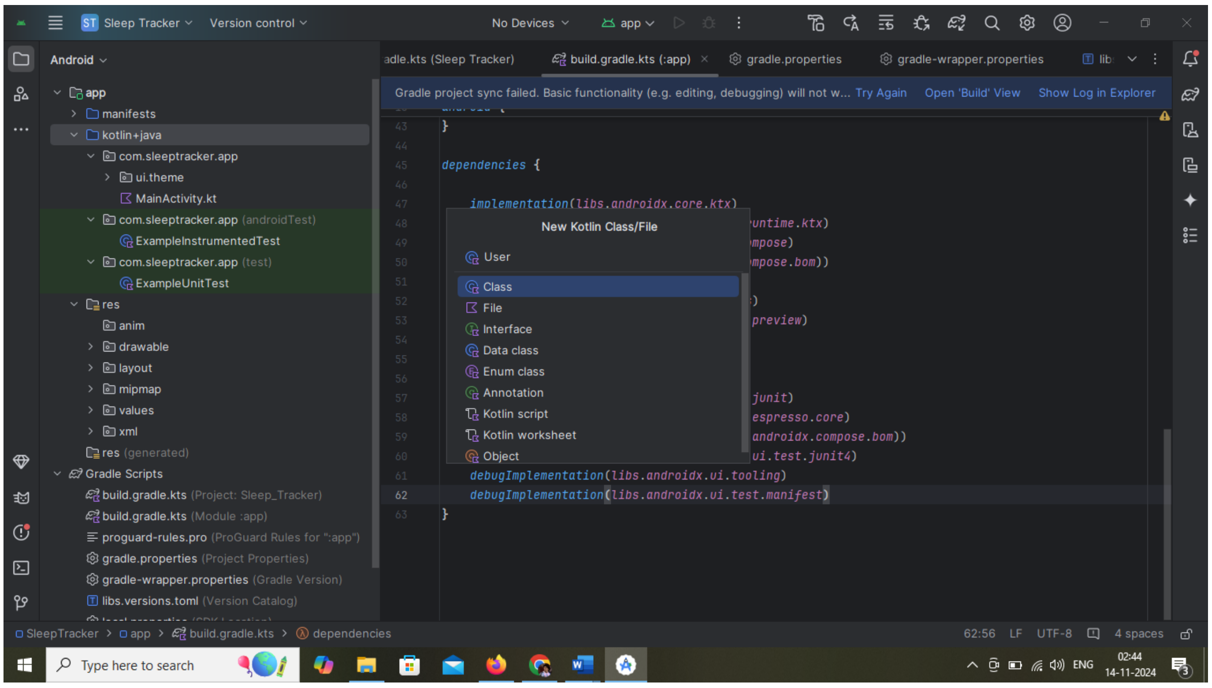
Key Data Points for this Step:

- Notifications Sent (e.g., reminders to sleep, motivational messages).
- User Interaction (whether the user took action, like going to bed earlier).

Recap: Step-by-Step Process in Your Sleep Tracker App

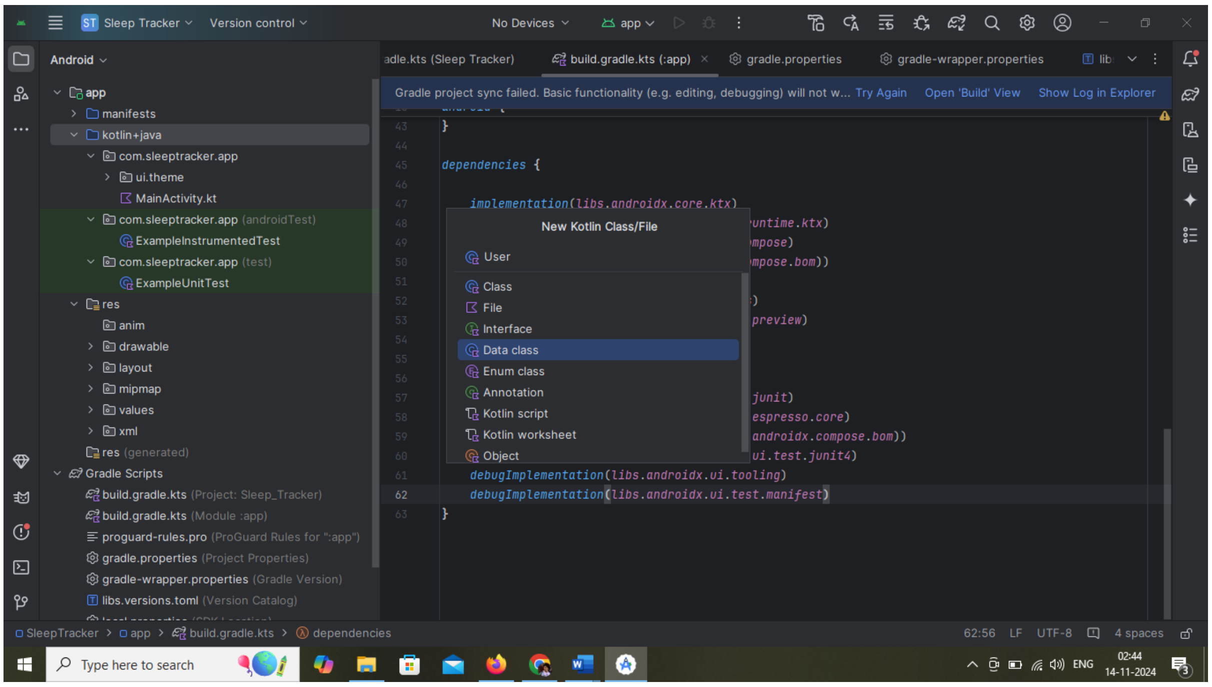
1. User Profile Setup: Register and set up the user’s sleep preferences.
2. Tracking Sleep Session: Log the start and end times, and gather sleep data like quality and interruptions.
3. Recording Sleep Stages: Capture data about sleep stages (light, deep, REM, awake).
4. Data Review and Insights: Review the sleep data, with insights about quality, duration, and trends.
5. Aggregating Data Over Time: Analyze sleep patterns and provide long-term insights.
6. Goal Setting and Progress: Set goals and track progress toward improving sleep habits.
7. Notifications and Reminders: Send reminders to help the user stay on track with their sleep goals.

Screenshots :



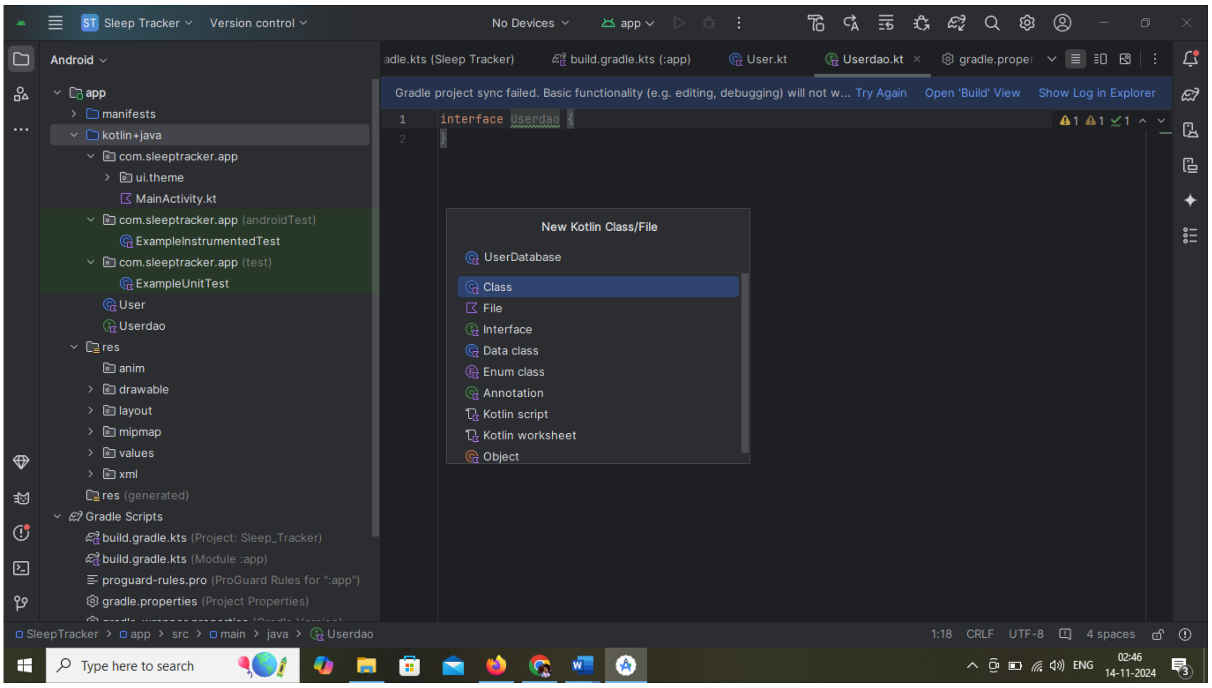
Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not w... Try Again Open 'Build' View Show Log in Explorer

```
}

dependencies {

    implementation(libs.androidx.core.ktx)
                                          untime.ktx)
                                          mpose)
                                          mpose.bom))

                                          )
                                          preview)



                                          junit)
                                          espresso.core)
                                          androidx.compose.bom))
                                          ui.test.junit4)
    debugImplementation(libs.androidx.ui.tooling)
    debugImplementation(libs.androidx.ui.test.manifest)
}
```



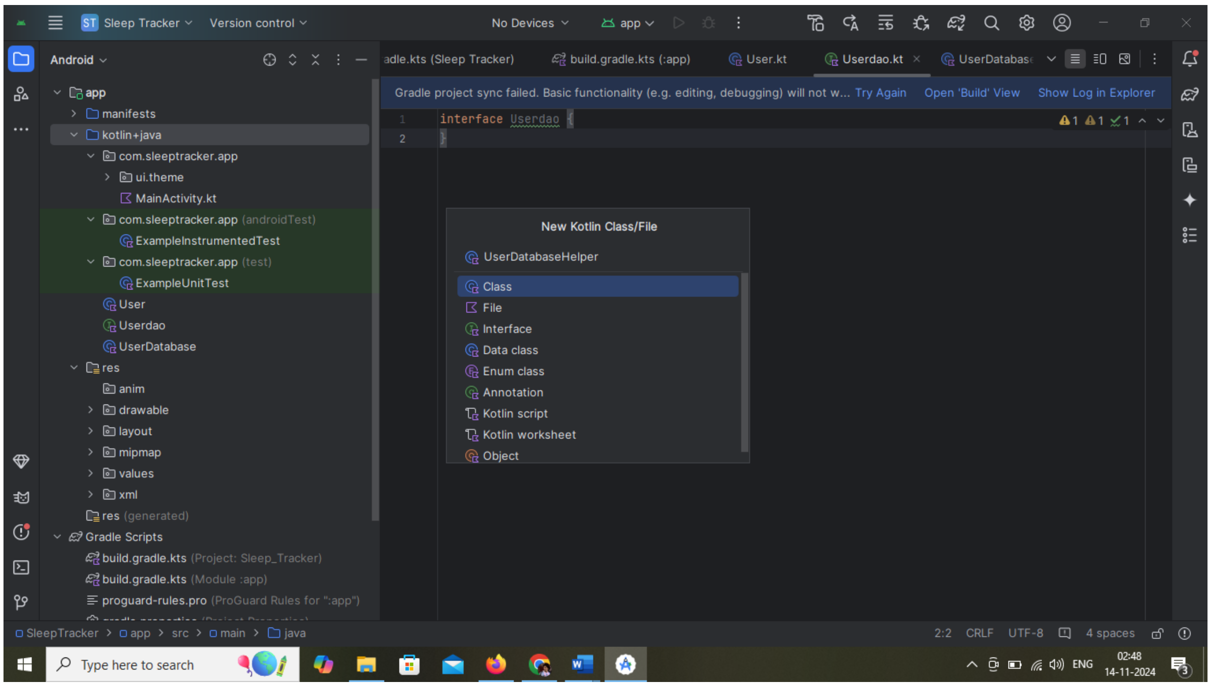
Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not w... Try Again Open 'Build' View Show Log in Explorer

```
}

dependencies {

    implementation(libs.androidx.core.ktx)
                                         untime.ktx)
                                         mpose)
                                         mpose.bom))

                                         )
                                         preview)


                                         junit)
                                         espresso.core)
                                         androidx.compose.bom))
                                         ui.test.junit4)
    debugImplementation(libs.androidx.ui.tooling)
    debugImplementation(libs.androidx.ui.test.manifest)
}
```



Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not w... Try Again Open 'Build' View Show Log in Explorer

```
interface Userdao {
}
```


Sleep Tracker · Version control · No Devices · app

Android

- app
  - manifests
  - kotlin+java
    - com.sleeptracker.app
      - ui.theme
      - MainActivity.kt
    - com.sleeptracker.app (androidTest)
      - ExampleInstrumentedTest
    - com.sleeptracker.app (test)
      - ExampleUnitTest
    - User
    - Userdao
    - UserDatabase
  - res
    - anim
    - drawable
    - layout
    - mipmap
    - values
    - xml
  - res (generated)
- Gradle Scripts
  - build.gradle.kts (Project: Sleep_Tracker)
  - build.gradle.kts (Module :app)
  - proguard-rules.pro (ProGuard Rules for ":app")

adle.kts (Sleep Tracker) · build.gradle.kts (:app) · User.kt · Userdao.kt · UserDatabase

Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not w... Try Again · Open 'Build' View · Show Log in Explorer

```
interface Userdao {
}
```

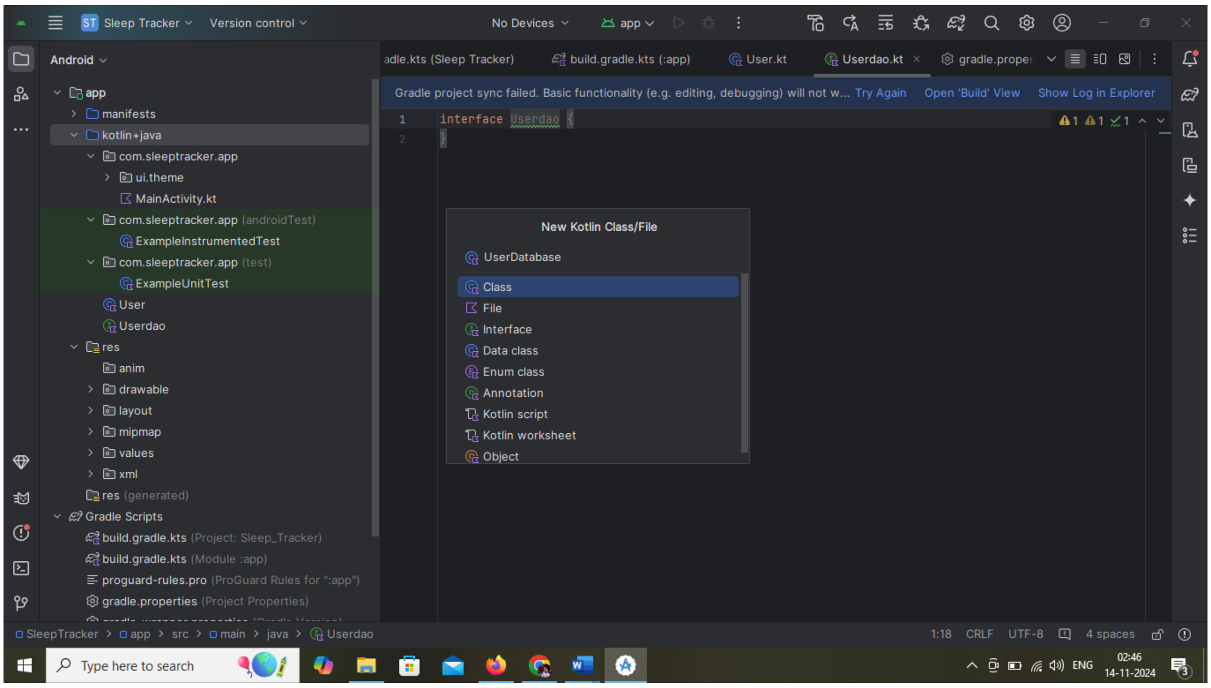
New Kotlin Class/File

UserDatabaseHelper

- Class
- File
- Interface
- Data class
- Enum class
- Annotation
- Kotlin script
- Kotlin worksheet
- Object

SleepTracker > app > src > main > java · 2:2 · CRLF · UTF-8 · 4 spaces

Type here to search · ENG · 02:48 · 14-11-2024


Sleep Tracker · Version control · No Devices · app

Android

- app
  - manifests
  - kotlin+java
    - com.sleeptracker.app
      - ui.theme
      - MainActivity.kt
    - com.sleeptracker.app (androidTest)
      - ExampleInstrumentedTest
    - com.sleeptracker.app (test)
      - ExampleUnitTest
    - User
    - Userdao
  - res
    - anim
    - drawable
    - layout
    - mipmap
    - values
    - xml
  - res (generated)
- Gradle Scripts
  - build.gradle.kts (Project: Sleep_Tracker)
  - build.gradle.kts (Module :app)
  - proguard-rules.pro (ProGuard Rules for ":app")
  - gradle.properties (Project Properties)

adle.kts (Sleep Tracker) · build.gradle.kts (:app) · User.kt · Userdao.kt · gradle.proper

Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not w... Try Again · Open 'Build' View · Show Log in Explorer

```
interface Userdao {
}
```

New Kotlin Class/File

UserDatabase

- Class
- File
- Interface
- Data class
- Enum class
- Annotation
- Kotlin script
- Kotlin worksheet
- Object

SleepTracker > app > src > main > java > Userdao · 1:18 · CRLF · UTF-8 · 4 spaces

Type here to search · ENG · 02:46 · 14-11-2024

Building Application UI And Connecting To Database:

1. Setup Database

- Define Database Tables: Create essential tables for users and sleep sessions. Here's a minimal schema for relational databases (like PostgreSQL, MySQL):
    - Users Table:
        - user_id (Primary Key)
        - email (Unique)
        - password_hash
    - SleepSessions Table:
        - session_id (Primary Key)
        - user_id (Foreign Key)
        - start_time
        - end_time
        - duration
        - sleep_quality

This allows the app to store the user's basic information and track their sleep data.

2. Basic Backend Setup

- Choose a Backend Framework: (e.g., Node.js with Express, Python Flask, or Django)
- Create API Routes: Define just two key routes:
    - POST /api/login: Handle user login.
    - POST /api/sleep-session: Log new sleep data.
- Connect to the Database: Set up your backend to communicate with the database (using an ORM like Sequelize for Node.js or SQLAlchemy for Python).

3. Basic UI Design

Design just the essential screens:

- Login Screen: Allows users to log in (enter email/password).
- Dashboard: Display a summary of the most recent sleep session and a button to log new sleep data.
- Sleep Tracking: Input for starting and ending a sleep session, along with sleep quality rating.

4. Connect UI with Backend (API Calls)

- Login Interaction:
    - Frontend sends user credentials to the backend POST /api/login API.
    - On success, the backend responds with a JWT (JSON Web Token), which the frontend stores (usually in localStorage or sessionStorage).

Sleep Data Logging:

- Frontend captures start time, end time, and sleep quality (this could be simple input fields or integrated with a wearable device).
- Frontend sends this data as a POST request to POST /api/sleep-session to log the session.
- The backend stores this data in the SleepSessions table.

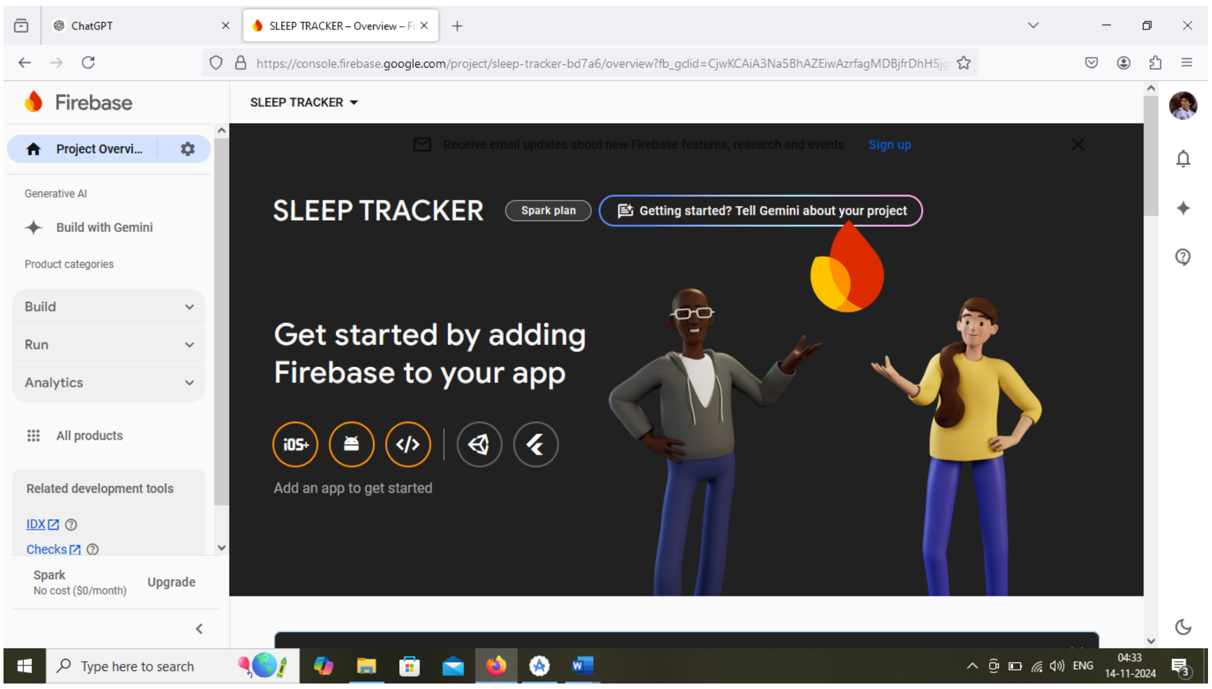
5. Basic Data Retrieval (Optional for Now)

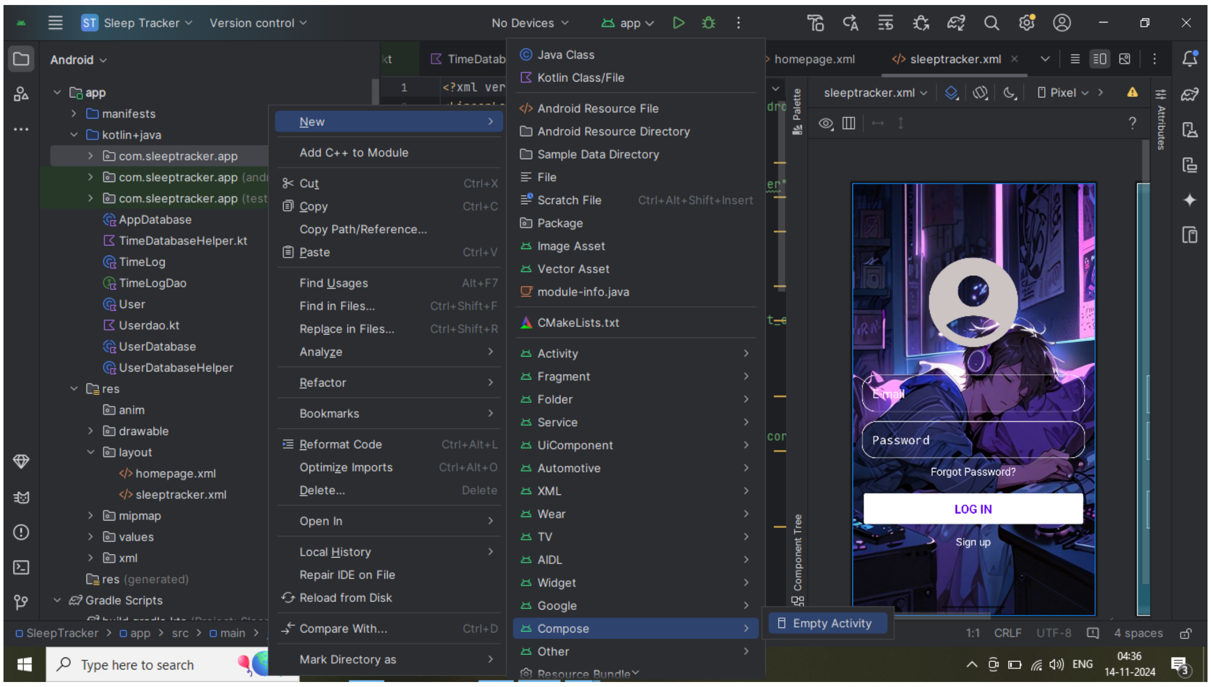
- Retrieve Last Sleep Session (optional step to show user data on dashboard):
    - Frontend sends a GET request to /api/sleep-session/latest (or similar).
    - The backend returns the most recent sleep data for the logged-in user.

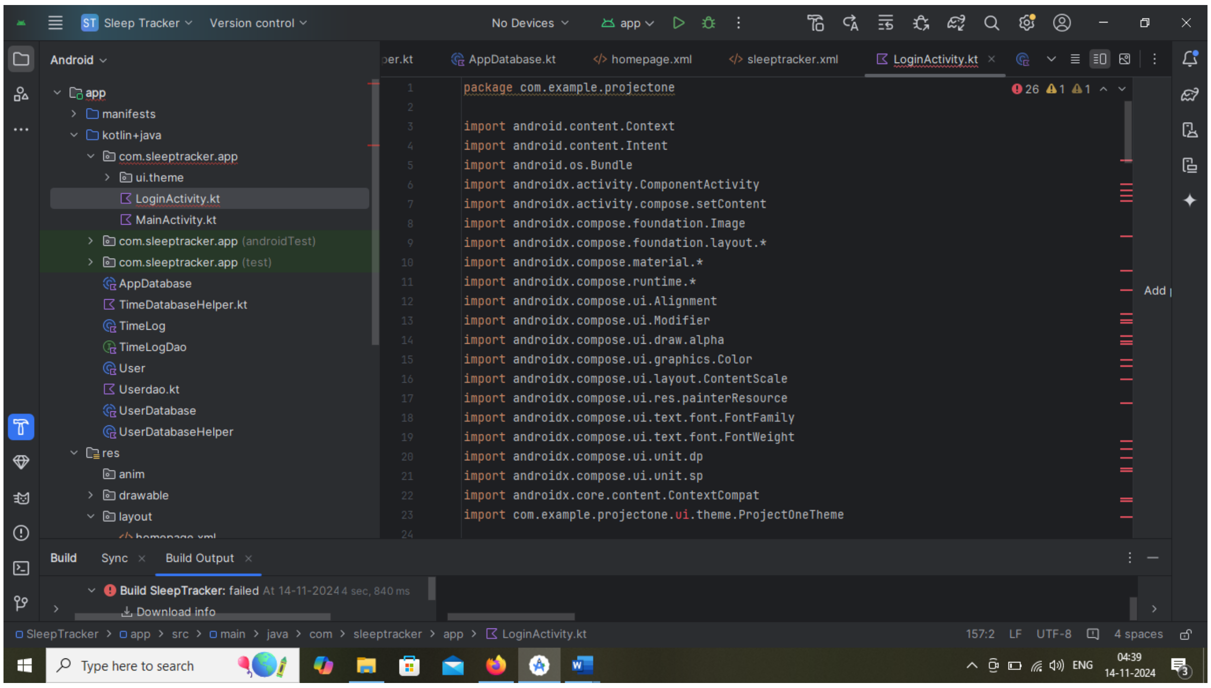
6. Testing

- Test basic login and sleep logging:
  - Ensure POST /api/login correctly authenticates the user.
  - Ensure POST /api/sleep-session correctly stores and logs the sleep data.
- Check data is correctly stored and retrieved from the database:
  - Use Postman or curl to test your API endpoints directly.
  - Confirm the UI displays the correct data (e.g., showing recent sleep session on the dashboard).

Screenshots:


ChatGPT
SLEEP TRACKER – Overview – Fi
https://console.firebase.google.com/project/sleep-tracker-bd7a6/overview?fb_gclid=CjwKCAiA3Na5BhAZEiwAzrfagMDBjfrDhH5j
Firebase
Project Overvi...
Generative AI
Build with Gemini
Product categories
Build
Run
Analytics
All products
Related development tools
IDX
Checks
Spark
No cost ($0/month)
Upgrade
SLEEP TRACKER
Receive email updates about new Firebase features, research and events
Sign up
SLEEP TRACKER
Spark plan
Getting started? Tell Gemini about your project
Get started by adding Firebase to your app
iOS+
Add an app to get started
Type here to search
ENG
04:33
14-11-2024





```
package com.example.projectone

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.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
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.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.projectone.ui.theme.ProjectOneTheme
```

Build SleepTracker: failed At 14-11-2024 4 sec, 840 ms

Modifying AndroidManifest.Xml:

Sure! Here’s a simple breakdown of how to modify your `AndroidManifest.xml` file:

1. Basic Structure of `AndroidManifest.xml`:

xml

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/Theme.MyApp">

        <!-- Main Activity -->
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
```

```
</manifest>
```

2.Common Modifications:

Add Permissions:

To request a permission, such as access to the internet:

xml

```
<uses-permission android:name="android.permission.INTERNET" />
```

Declare a New Activity:

To declare a new activity:

xml

```
<activity android:name=".SecondActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Set the Application Theme:

To set a theme for the app:

xml

```
<application
    android:theme="@style/Theme.AppCompat.Light">
```

Make Sure Your Main Activity is Set:

The main activity should have an `<intent-filter>` with the `MAIN` action and `LAUNCHER` category:

xml

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

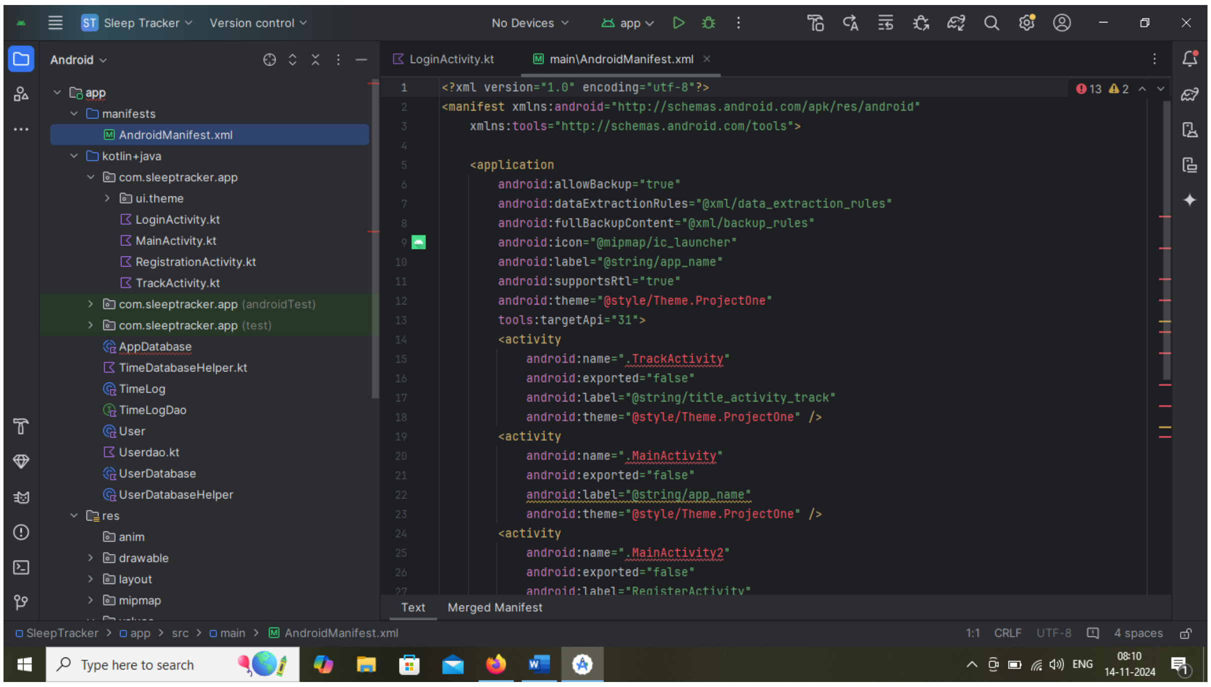
Example with All Common Changes:

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

    <!-- Permissions -->
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/Theme.AppCompat.Light">

        <!-- Main Activity -->
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- Second Activity -->
        <activity android:name=".SecondActivity">
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

Key Points:

- `<uses-permission>`: Requests permissions (e.g., internet access).
- `<activity>`: Declares an activity (e.g., MainActivity or SecondActivity).
- `<intent-filter>`: Specifies how the activity can be launched (e.g., as the main activity or in response to a specific action).

Screenshot:



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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.ProjectOne"
        tools:targetApi="31">
        <activity
            android:name=".TrackActivity"
            android:exported="false"
            android:label="@string/title_activity_track"
            android:theme="@style/Theme.ProjectOne" />
        <activity
            android:name=".MainActivity"
            android:exported="false"
            android:label="@string/app_name"
            android:theme="@style/Theme.ProjectOne" />
        <activity
            android:name=".MainActivity2"
            android:exported="false"
            android:label="RegisterActivity"
```

Running The Application:

To run a sleep tracking app on Android Studio, follow these steps:

1. Set Up Permissions

In `AndroidManifest.xml`, add necessary permissions for sensors and health data (if applicable):

xml

<uses-permission android:name="android.permission.BODY_SENSORS" />

<uses-permission android:name="android.permission.INTERNET" />

```
```

2. Implement Sleep Tracking Features

- Using Sensors (like accelerometer) to detect sleep motion:

```
  java
  SensorManager sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
  Sensor accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
  sensorManager.registerListener(...);
```

- Using Google Fit for sleep data:
  - Add the Fit API dependency:

```
    gradle
    implementation 'com.google.android.gms:play-services-fitness:21.0.1'
```

  - Access sleep data with Fit APIs.

3. Run on Emulator or Device

- Emulator: Create and launch an Android Virtual Device (AVD) from the AVD Manager.
- Device: Enable USB Debugging on your phone and connect it via USB.

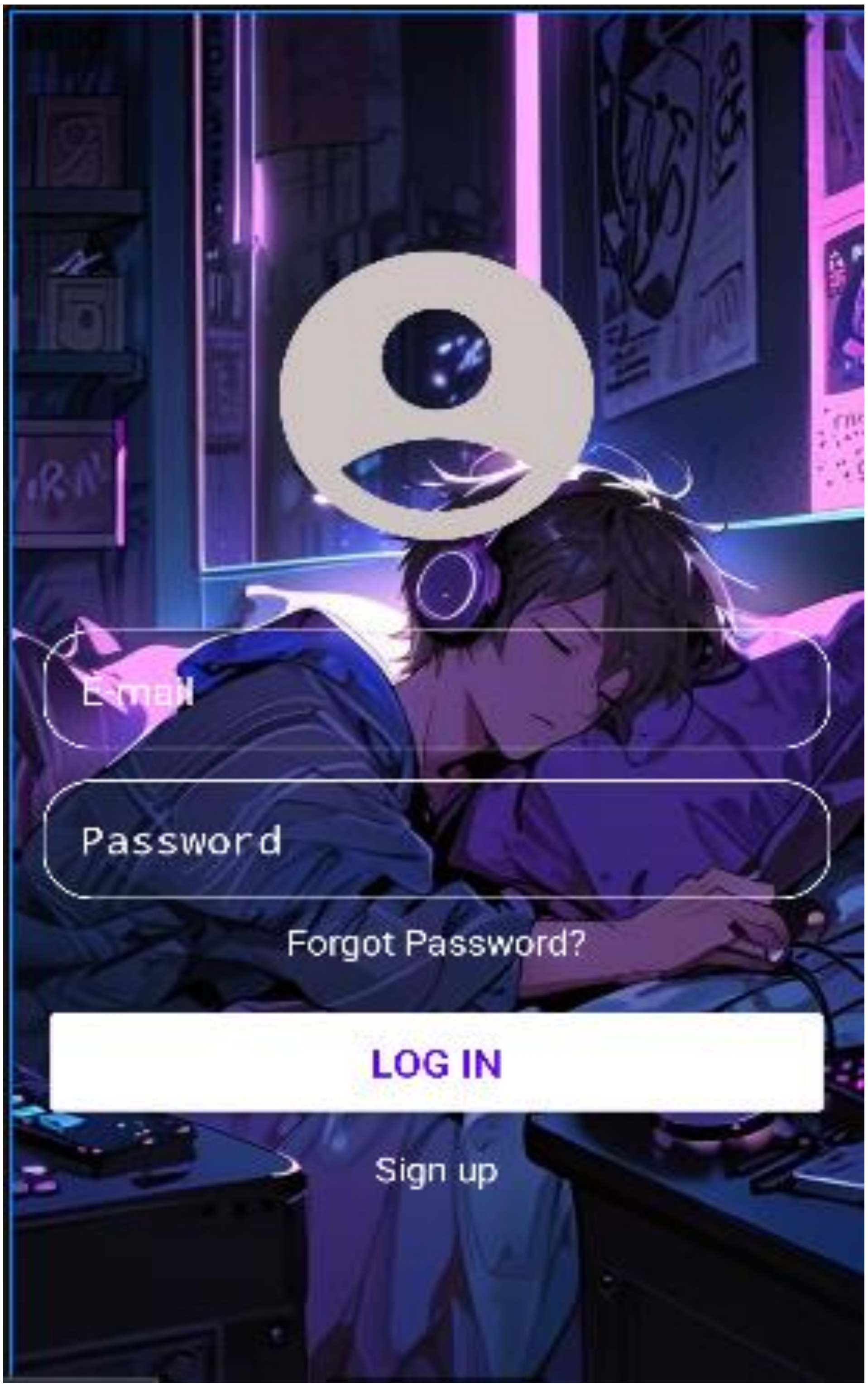
4. Debug and Test

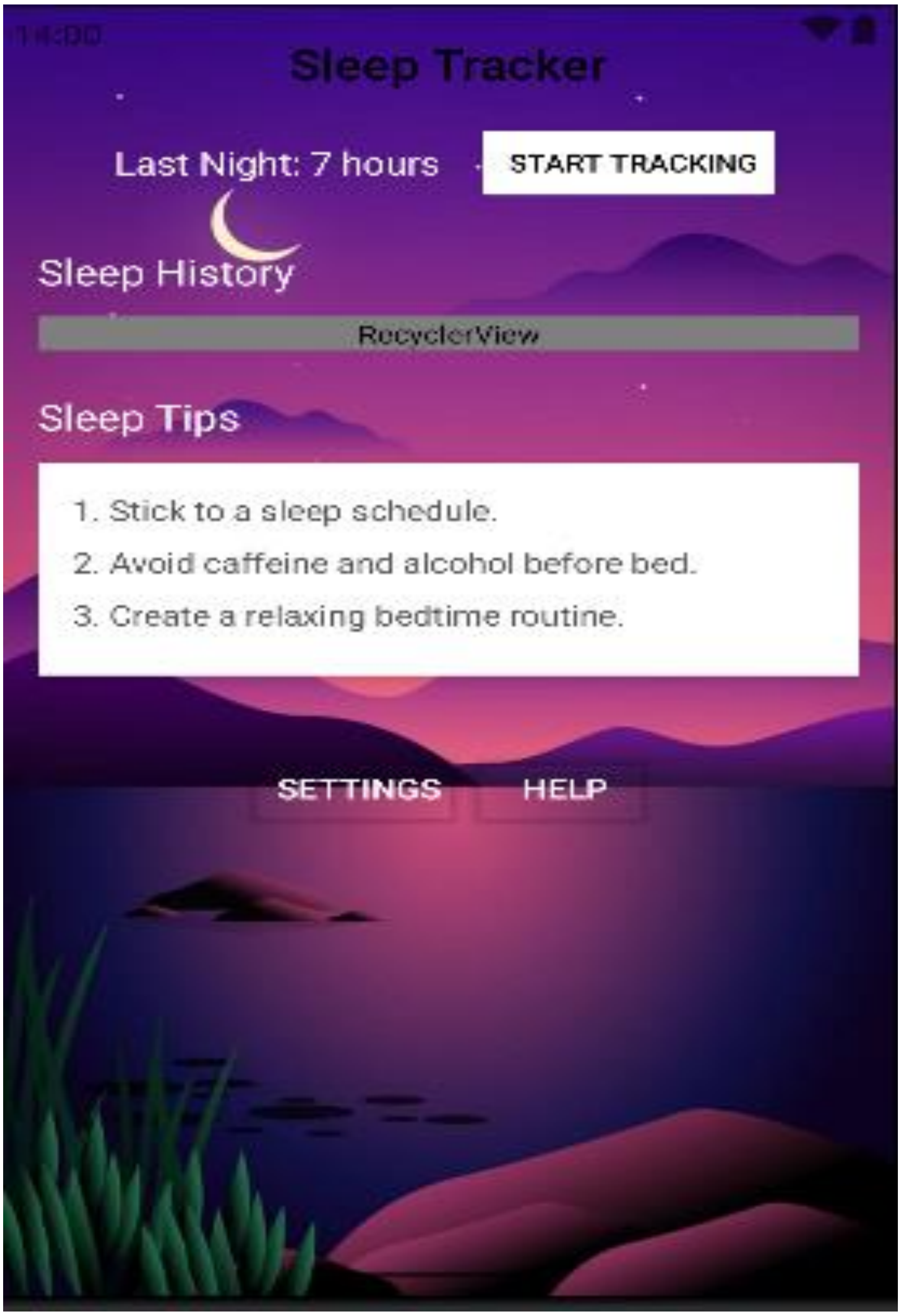
- Use Logcat to monitor sensor data or app behavior while testing.

5. Publish (Optional)

Once tested, build a release APK or app bundle to publish on the Google Play Store.

Screenshots:





1. Stick to a sleep schedule.
2. Avoid caffeine and alcohol before bed.
3. Create a relaxing bedtime routine.