**1.0 Introduction**

The **"ELT Tracking"** application is designed for advanced technology that will be user- friendly and, therefore, offer applicable data regarding users' fitness activities. It would be the best tracking and activity recording application for health-conscious people, fitness fans, and athletes alike. It has features to keep track of physical performance, improve health, and participate in features such as challenges. The application also uses intuitive interfaces, which brings the insights users need to make choices regarding their health and fitness journeys.

**ELT Tracking** offers a range of features, including:

* Running and Cycling Tracking
* BMI Calculator, Calories Counting
* Reminders and alerts.
* Weather Forecasts.
* Leaderboard and Challenges

The app is designed primarily for fitness junkies, competitive sports people, and health- conscious individuals who require an efficient tool for tracking or improving their performances. It would serve the purpose for novices in the fitness adventure and seasoned athletes in the refining of their training. The social dimension like the leaderboards and challenges promotes motivation and generates community amongst users.

The app plans on improving its users further by incorporating smartwatch compatibility. This will afford users access to tracking activities with less effort as well as real-time notifications directly on their wearable devices. Availability of such functions as heart rate measurements, step counts, as well as other possible advanced health metrics, will make it possible to provide a holistic picture of users' fitness data.

# 2.0 Application Overview

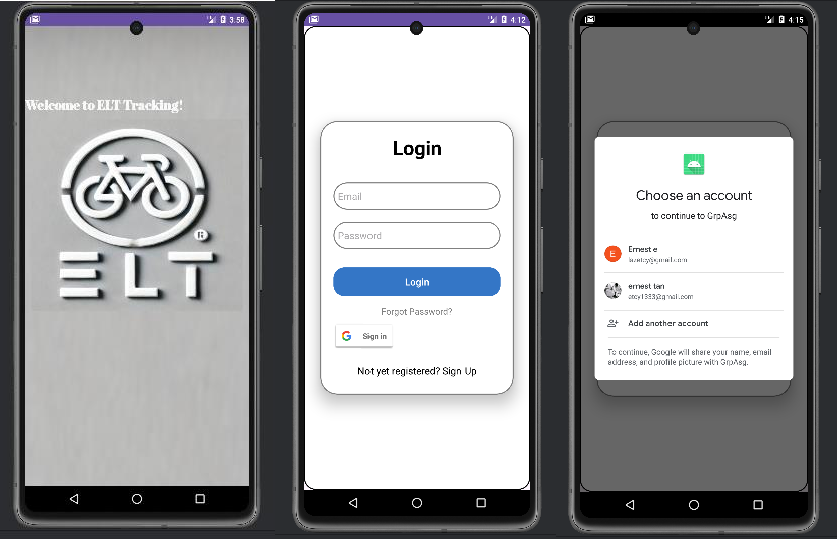
**2.1 Features**

1. **BMI Calculator**: Calculates Body Mass Index based on user input and categorizes results into health ranges (e.g., underweight, normal, overweight). It saves results with timestamps for progress tracking(BmiCalculatorActivity).
2. **Activity Tracking (Running and Cycling)**: Tracks real-time metrics like distance, speed, calories burned, and route mapping using GPS technology. Users can pause, resume, and save sessions, with data stored in Firebase for analysis and leaderboard updates (CyclingActivity, RunningActivity).
3. **Reminders and Alerts**: Enables users to schedule reminders for fitness tasks or maintenance (e.g., "Check Tyre"), with notifications delivered via the app’s alarm system. It also provides real-time weather alerts to ensure safety during outdoor activities (ReminderActivity, WeatherAlertPopup).
4. **Weather Forecasts**: Offers live weather updates and 5-point forecasts to help users plan their activities effectively, taking into account local weather conditions (WeatherActivity).
5. **Leaderboard and Challenges**: Encourages community engagement by allowing users to compete with peers, track progress, and celebrate achievements through challenges and rankings.

# 2.3 User Manuals

**Splash Screen and Log In (FlashScreenActivity.java & LoginActivity.java)**

(Appendix 1, 2)



1

3

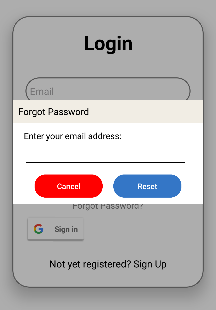
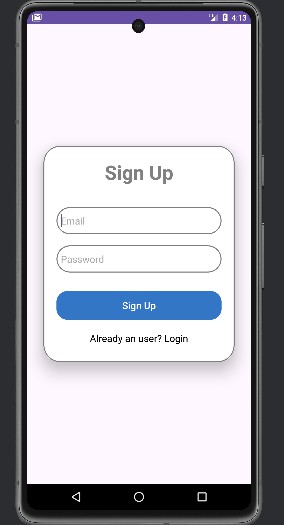
2

1. The FlashScreenActivity features animations, including a fade-in effect for text and a scale-up effect for an image and plays a sound effect at startup to enrich the user experience. This activity also integrates a Firebase Authentication check to determine whether a user is already signed in. Based on the user's authentication status, it transitions seamlessly to either the MainActivity (if the user is signed in) or the LoginActivity (if no user is signed in).
2. The LoginActivity is the central access point for users to authenticate and gain access to the application. It supports traditional email-password authentication and integrates Google Sign-In for seamless user login. The activity validates user inputs, ensuring proper formatting for email addresses and non-empty passwords, and provides error messages for invalid input.
3. Image shows the Google Sign In Option after the google button is clicked. The GoogleSignInClient and GoogleSignInOptions then request user's ID token and email address for authentication. Upon successful sign-in, the user's data, including the ID token, is retrieved from the resulting intent. This ID token is then used to create a Firebase credential through the GoogleAuthProvider. The credential is passed to Firebase's signInWithCredential method.

Using the UserManager.java (Appendix 4) to authenticate the user and manage user information within the application. It store and retrieve information about the currently authenticated user. It also includes methods to set and get the FirebaseUser and accessing GoogleSignInAccount information

# SignUp and Forgot Password (SignUpActivity.java)

(Appendix 3)



2

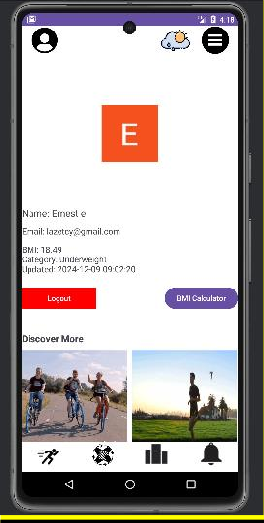
1

1. The SignUpActivity provides functionality for new users to create an account in the application. It allows users to input their email and password, validates these inputs, and uses Firebase Authentication to register the new account. If the input fields are left empty, appropriate error messages are displayed to guide the user.

Upon successful registration, a confirmation message is shown, and the user is redirected to the LoginActivity.

1. The forgot password feature allows users to reset their password via email. This is implemented with a custom dialog for user input and feedback, ensuring an intuitive experience. The activity also includes a "Sign Up" redirect, guiding new users to create an account.

# User Profile (MainActivity)



Community Page

LeaderBoard Page

Running Page

Reminder Page

3.Scrollable Activities

2.Logout

1. User Details

Option Menu (History Page)

User Profile Page

Weather Page

(Appendix 7)

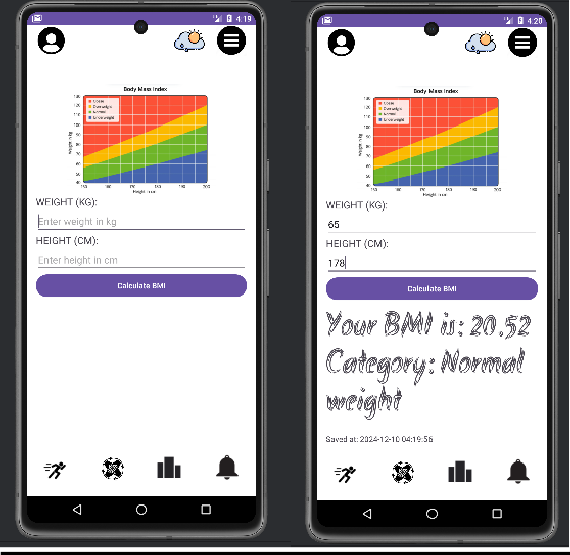
BMI Calculator

1. User Details. Fetch and present key details about the logged-in user, including their name, email, and profile picture. This is achieved using Firebase and Google Sign-In integration. If a user is authenticated via Google, their account details, such as the display name, email, and profile picture URL, are retrieved using the GoogleSignInAccount class. For the profile picture, the app uses the Glide library to load and display the image. Display the user BMI if the user used the BMI calculator activity.
2. The logout button provides users with a straightforward way to securely log out of their account. When the logout button is clicked, the application uses the Google Sign-In API's signOut method to log out any Google-authenticated users. After signing out, the user is redirected to the LoginActivity.
3. The Scrollable Activities display 4 gif image Horizontally. Mainly about what is the attraction in this app.

Other yellow highlighted button will redirect user to the respective page by using either Navigation.java or Cycling Vavigation.java (Appendix 5, 6). Every button clicked, there is a button clicked sound.

# BMI Calculator (BmiCalculatorActivity.java, BmiDatabaseHelper.java)

(Appendix 8, 9)



1.Calculate

Height (user input)

2.Calculate

Weight (user input)

BMI Chart

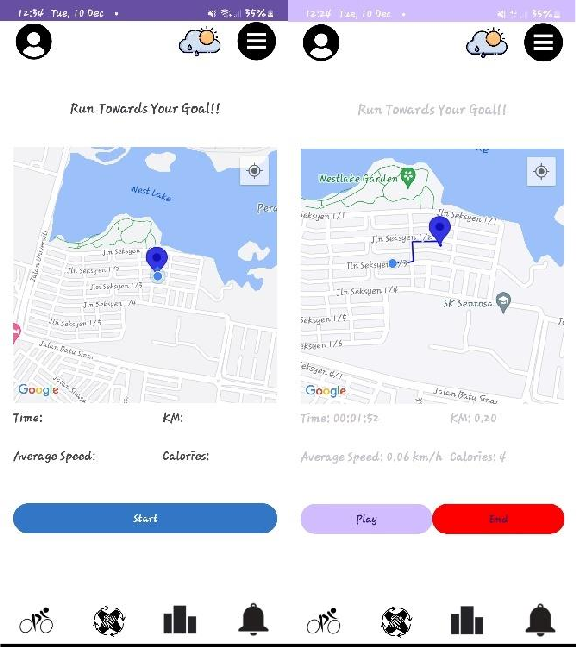
The activity integrates a timestamp feature, recording the date and time of the calculation. This data, including the BMI value, category, and timestamp, is saved locally in an SQLite database using the BmiDatabaseHelper. This ensures users can view their BMI history.

1. When the "Calculate" button is clicked, it calculates the BMI then categorized into standard categories such as "Underweight," "Normal weight," "Overweight," or "Obese,".
2. Display user calculated results and save to database.

# Running and Cycling

1. **Running Activity (RunningActivity.java)**

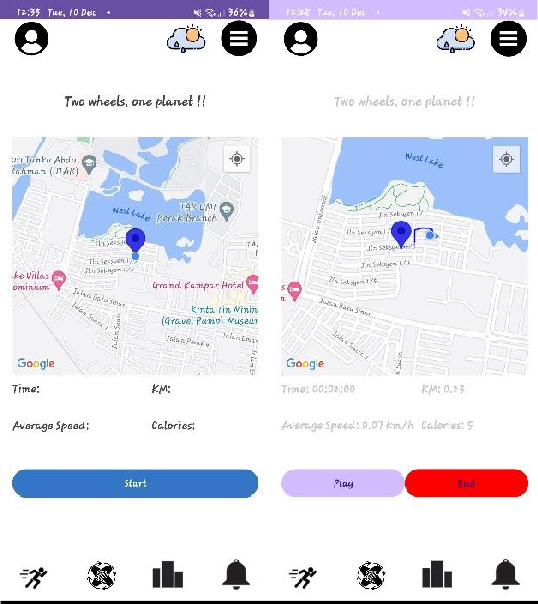
(Appendix 10)



Cycling Page

# Cycling Page (CyclingAvtivity.java)

(Appendix 11)



1. The RunningActivity is a fitness tracking feature that records and monitors running sessions in real-time. It provides functionalities for tracking distance, duration, average speed, and calories burned during a run. Users can start, pause, resume, or stop a session with intuitive button controls. The activity integrates Google Maps to visually display the user’s running route, marked with a polyline that updates dynamically as the user moves.
2. The CyclingActivity is similar to the RunningActivity but tailored for cycling sessions. It tracks real-time metrics such as distance traveled, average speed, elapsed time, and calories burned. The activity uses Google Maps to display the cycling route, with a polyline that updates dynamically based on the user’s location. Users can start, pause, resume, and stop their tracking session using dedicated controls.

Both RunningActivity and CyclingActivity integrate Firebase for session data storage but store in different activity type in the firebase. Blue track will appear as user goes.

# Weather Page (WeatherActivity.java)

(Appendix 15)



1

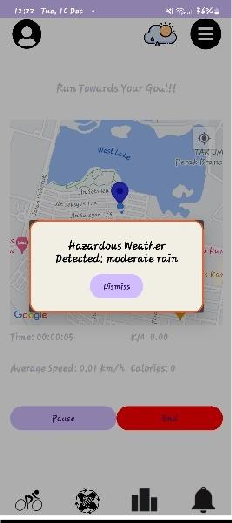
The WeatherActivity is designed to provide users with real-time weather updates and forecasts. It integrates location-based services to fetch weather data dynamically. By using the device’s GPS through the Fused Location Provider Client, the activity determines the user's current latitude and longitude, retrieving precise, localized weather information via the OpenWeatherMap API.

It also presents comprehensive weather details, including the current temperature, weather conditions (e.g., sunny or rainy), atmospheric humidity, wind speed, and a "feels- like" temperature that adjusts for environmental factors.

1. 5-point weather forecast, showing weather predictions at three-hour intervals. (in 24hours based). Each forecast entry provides the predicted temperature, weather condition, and a corresponding icon, such as a sun for sunny conditions or a cloud with rain for rainy forecasts.

# Weather Alert Activity (WeatherAlertPopup.java)

(Appendix 16)



The Weather alert activity monitor weather conditions and notify users about hazardous weather in real-time. It leverages the OpenWeatherMap API to fetch weather details based on the user's geographic coordinates (latitude and longitude) and presents warnings in the form of a popup dialog if adverse conditions are detected. After the dismiss button clicked, the weather alert will not pop up in **15 minutes** time interval to prevent too frequent pop up.

# History Page (HistoryActivity.java, HistoryEntry.java, HistoryAdapter.java)

(Appendix 12, 13, 14)

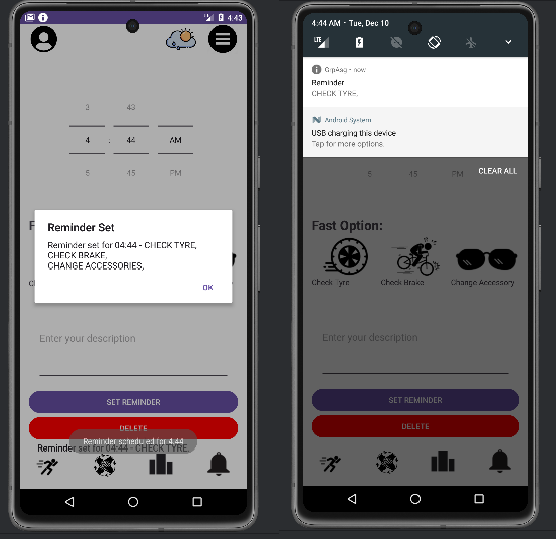
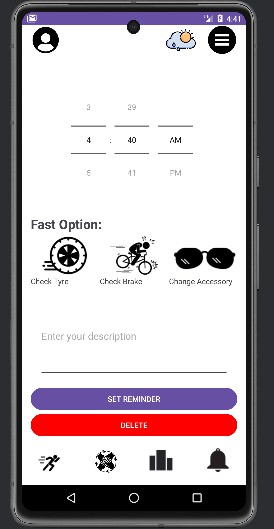


1

This activity retrieves data from Firebase, where users' running and cycling records are stored. It uses a RecyclerView to display this information, providing a smooth scrolling interface for users to explore their previous activities. Each entry in the history includes details such as distance covered, time spent, calories burned, and the date of the activity.

1. Users can toggle between viewing running and cycling history. **Reminder Page (ReminderActivity.java, ReminderAdapter.java, ReminderBroadcastReceiver.java)**

(Appendix 17, 18, 19)



1

Reminder History

Notifications shows

The ReminderActivity allows users to create, manage, and view reminders for their activities. It features an interactive interface with tools for setting custom reminders using a TimePicker and description input.

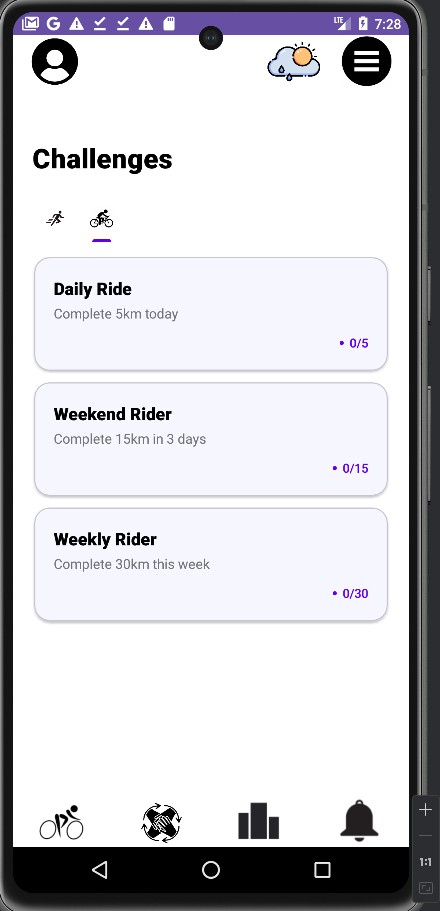
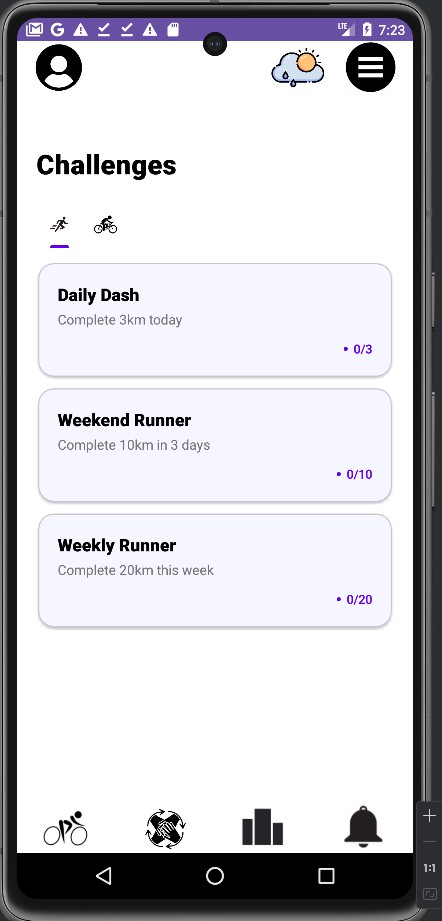
The ReminderAdapter is responsible for rendering the list of reminders in the RecyclerView within the ReminderActivity.

The ReminderBroadcastReceiver handles the notifications for scheduled reminders. When a reminder's alarm triggers, this broadcast receiver creates and displays a notification with the reminder's details.

1. The Fast option allow user to input words such as “Check tyre, Check brake and Change accessories” easily by just clicking on it.

# Challenges(ChallengesActivity.java, ChallengersAdapter.java, Challenge.java, ChallengersProgress.java)

(appendix 20,21,22,23)

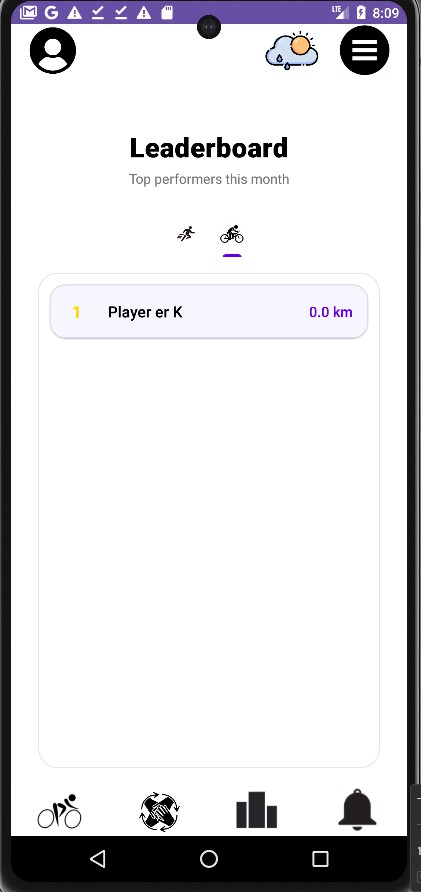
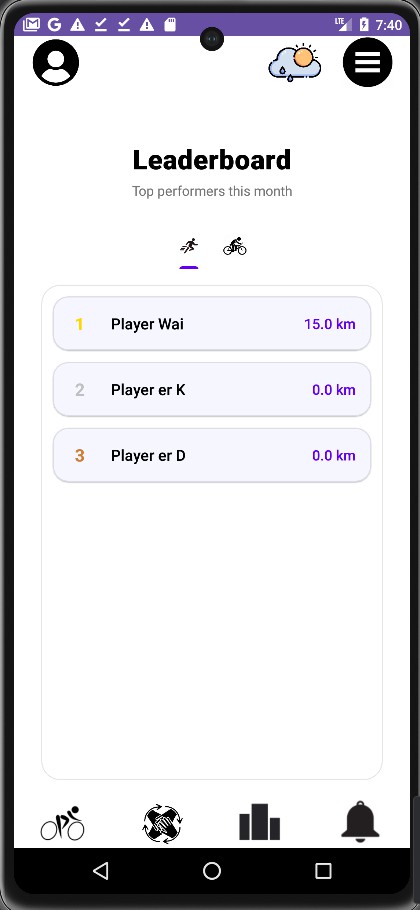


The challenges activity allows users to view challenges that are available to complete in two different activities, running and cycling. The challenges are retrieved from the firebase and save in it so users still view their ongoing challenges when using another device, after they have logged in with their accounts.

1. The are each three challenges available in one of the two activities, including daily, three days and weekly challenges, with running as the default one. Users who prefer cycling can click on the cycling icon underneath the title “challenges” to switch to the three challenges offered for cycling.

# Leaderboard(LeaderboardActivity.java, LeaderboardAdapter.java, LeaderboardEntry.java)

(Appendix 24,25,26)

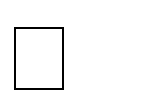


Leaderboard shows the top users that have the most distance travelled(km), the leaderboard is split into two, one for running and another for cycling to ensure fairness of competing among the users with running as the default. The leaderboard refreshes once per month to give opportunities for users that start using the app later than others

1. Users can view the names of the leaderboard are started with player, and their total distances travelled. Users can also click on the cycling icon above the leaderboard to change and view the leaderboard for cyclists.

# ​Technical Details

* 1. **Development Tools**
     + **Android Studio:** The app was developed using **Android Studio**, the official Integrated Development Environment (IDE) for Android development. It provides a robust set of tools for building, testing, and debugging Android applications.
     + **Programming Language:** The application was written in **Java**, which is the preferred language for modern Android development due to its conciseness, safety features, and interoperability with Kotlin.
     + **Minimum SDK Version:** The app supports Android versions starting from **API level 24 (Android 7.0 Nougat)** to ensure compatibility with most devices in use today.
     + **Gradle:** Used for managing dependencies and building the application efficiently.
  2. **Libraries and Dependencies**

1. **Google Maps API**: Integrates interactive maps for tracking activities such as running and cycling [4].
   * Dependency: com.google.android.gms:play-services-maps:18.1.0
2. **Google Location Services**: Provides location data for real-time activity tracking and route mapping [5].
   * Dependency: com.google.android.gms:play-services-location:21.0.
   * ​
3. **Firebase Realtime Database**: Stores user activity history and leaderboard data [2].
   * Dependency: com.google.firebase:firebase-database:20.0.0
4. **Firebase Storage**: Manages file uploads and storage, such as user profile images or activity-related data [3].
   * Dependency: com.google.firebase:firebase-storage:20.2.1
5. **OkHttp**: Handles network requests for weather updates and other APIs [8].
   * Dependency: com.squareup.okhttp3:okhttp:4.10.0
6. **Volley Library**: Facilitates efficient API calls, such as fetching weather data [7].
   * Dependency: com.android.volley:volley:1.2.1
7. **Firebase Authentication**: Supports user authentication through email and Google Sign-In [1].
   * Dependency: com.google.firebase:firebase-auth:22.1.1
8. **Firebase Analytics**: Tracks app usage statistics and user behavior for better insights .
   * Dependency: com.google.firebase:firebase-analytics
9. **Glide**: Loads and displays images, such as user profile pictures or weather icons [6].
   * Dependency: com.github.bumptech.glide:glide:4.15.1
10. **AndroidX AppCompat**: Provides backward-compatible features and modern UI components [10].
    * Dependency: androidx.appcompat:appcompat:1.7.0
11. **Material Design Components**: Implements Material Design UI for buttons, navigation bars, and dialogs [9].
    * Dependency: com.google.android.material:material:1.12.0
12. **Custom GButton Library**: Adds visually appealing button components to enhance UI interactivity [13].
    * Dependency: com.github.TutorialsAndroid:GButton:v1.0.19
13. **AndroidX ConstraintLayout**: Enables flexible and responsive layouts for modern app designs [11].
    * Dependency: androidx.constraintlayout:constraintlayout:2.2.0

# 4.0 Implementations

1. **User Authentication with Google Sign-In**

To enable users to securely sign in and track their fitness progress across devices, the app integrates **Google Sign-In** for authentication.

* + **Add Google Sign-In dependency in build.gradle:** implementation 'com.google.android.gms:play-services-auth:22.1.1' implementation 'com.google.android.gms:play-services-auth:21.2.0'

# Code for Google Sign-In integration:

val gOptions = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT\_SIGN\_IN)

.requestIdToken("370438433891-eii12nk3dtto3c53e3pe3b4pbrro5ohv.apps.googleusercontent.com")

.requestEmail().build();

gClient = GoogleSignIn.getClient(this, gOptions); googleBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent signInIntent = gClient.getSignInIntent(); activityResultLauncher.launch(signInIntent);

}

});

ActivityResultLauncher<Intent> activityResultLauncher = registerForActivityResult( new ActivityResultContracts.StartActivityForResult(),

result -> {

if (result.getResultCode() == Activity.RESULT\_OK) { Intent data = result.getData();

Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data); try {

GoogleSignInAccount account = task.getResult(ApiException.class);

// Successfully signed in

String idToken = account.getIdToken(); // Use this for Firebase Authentication

Toast.makeText(LoginActivity.this, "Sign-In Successful", Toast.LENGTH\_SHORT).show();

// Continue to MainActivity finish();

Intent intent = new Intent(LoginActivity.this, MainActivity.class); startActivity(intent);

} catch (ApiException e) {

Toast.makeText(LoginActivity.this, "Something went wrong: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

}

}

});

# Fetching Activity Data from Google Fit

The app retrieves fitness data such as steps taken and calories burned from **Google Maps using the Google Maps API**.

* + **Add Google Maps API dependency in build.gradle:** implementation ‘com.google.android.gms:play-services-maps:18.1.0’ implementation ‘com.google.android.gms:play-services-location:21.0.1’

# Storing and Retrieving Data Using Room Database

The app uses **Room** for local database storage, where user profiles, activity logs, and meal entries are saved.

# Add Room dependencies in build.gradle:

1. **Displaying Data in RecyclerView**

The app uses a **RecyclerView** to display the user's activity logs, meal entries, or workout progress.

# Layout for RecyclerView in XML:

Challenges Faced

# Google Fit Integration:

1. One of the major challenges was correctly handling the permissions required for accessing Google Fit data. This was solved by checking for permissions during the first app launch and asking the user for permission dynamically.

# Handling Large Data Sets:

As the app tracks activity logs and meal entries, large amounts of data can accumulate over time. To optimize performance, we implemented **paging** in RecyclerView and lazy loading in database queries, reducing memory usage and improving load times.

# Background Syncing:

Ensuring that data syncing happens in the background without affecting user experience was another challenge. We used **WorkManager** to handle periodic data synchronization with Google Fit, ensuring that data is updated even when the app is not in the foreground.

# ​Future Enhancements

The ELT Tracking App is already a strong tool for keeping track of fitness and tying the community together but proves to be even stronger in terms of what more it can offer to users in the future. One such probably the most awaited feature is the smartwatch integration. The app would provide real-time access to heart rate, step count, and oxygen saturation metrics right on the wrist by enabling compatibility with leading wearable devices. It would also allow users to control the app's features, such as starting and pausing activity tracking, receiving notifications, and monitoring weather updates, without needing their phone. Such a convenience would empower them to remain focused on their fitness goals.

The app will also have one more major enhancement in terms of personalized insights and recommendations. The incorporation of data analytics via machine learning may allow it to analyse users' historical data and provide personalized suggestions regarding workouts, dietary plans, or even rests days. For example, the runner who often excels at later hours could receive a minor nudge to run that little bit faster in the evening, turning the app into a digital fitness coach tailored to suit one's needs and progress, thus making it easier for everyone to achieve their needs.

Finally, user involvement will be taken to a different level by community-driven challenges and rewards. Monthly fitness challenges of this app will comb individual involvement and group participation. The gamification elements like badges, leaderboards, and even real-world rewards such as discounts on fitness gear will help keep users encouraged. To further build and foster this sense of community, the app will add group challenges that allow users to form teams and compete collectively. These improvements are not just going to make the app interactive but also build support networks that energize users to keep up their healthy journeys. All these current developments indicate that the application is growing with the time, users, and challenges in fitness.

# ​Conclusion

The ELT Tracking App is one of the most comprehensive and intelligently designed applications that meet both the fitness and health-oriented user's needs. The app boasts features like activity tracking for running and cycling, a BMI calculator, weather alerts, reminders, community-driven leader boards, all allowing the user to make his or her fitness lives a journey for themselves. User-friendly interface, the robust technology integrated into it, and real-time analytics add to that seamless experience to users of all fitness levels. This app propels its users to be active within the achieved by being aware of what they do and encouraging community involvement.

The application strategy uses modern technology to talk about Firebase being the data storage for the apps, Google Maps for showing the route pathways, and APIs for weather updates. It thus makes it a one-stop fitness ecosystem where users can be on track with their progress, timely reminders push them, and challenges receive from contributing members keep them motivated. There are even safety and precautionary measures, such as stating the weather alert, which implies the focus on the user rather than just mere functionality. The integration of such functions makes the application much easier to use and helps turn it into a society-friendly assistant to fitness and person wellbeing.

While these enhancements-over-informed smartwatch integration, personalized insights, and community features-poised to complete the app's evolution with respect to current user behavior, the intention is to push further into really expanding the capabilities of this versatile instrument for a fitness program and much more. The innovative concept behind the ELT Tracking App, and its continuing futuristic roadmap, will definitely make it a potential asset to users in their quest to lead healthier, more active lives. This app is not a mere tracker of activities but a partner in the journey to sustained fitness and well-being.

# References

1. Google LLC, "Firebase Authentication," [Online]. Available: https://firebase.google.com/products/auth. [Accessed: 10-Dec-2024].
2. Google LLC, "Firebase Realtime Database," [Online]. Available: https://firebase.google.com/products/realtime-database. [Accessed: 10-Dec-2024].
3. Google LLC, "Firebase Storage," [Online]. Available: https://firebase.google.com/products/storage. [Accessed: 10-Dec-2024].
4. Google LLC, "Google Maps Platform," [Online]. Available: https://developers.google.com/maps. [Accessed: 10-Dec-2024].
5. Google LLC, "Google Play services: Location," [Online]. Available: https://developers.google.com/android/reference/com/google/android/gms/location/packa ge-summary. [Accessed: 10-Dec-2024].
6. BumpTech, "Glide: An image loading and caching library for Android," [Online]. Available: [https://github.com/bumptech/glide.](https://github.com/bumptech/glide) [Accessed: 10-Dec-2024].
7. Google LLC, "Volley: HTTP library for Android," [Online]. Available: [https://developer.android.com/training/volley.](https://developer.android.com/training/volley) [Accessed: 10-Dec-2024].
8. Square, Inc., "OkHttp: An HTTP & HTTP/2 client for Android and Java applications," [Online]. Available: https://square.github.io/okhttp/. [Accessed: 10-Dec- 2024].
9. Google LLC, "Material Design Components," [Online]. Available: https://material.io/develop/android. [Accessed: 10-Dec-2024].
10. Google LLC, "AndroidX AppCompat," [Online]. Available: [https://developer.android.com/jetpack/androidx/releases/appcompat.](https://developer.android.com/jetpack/androidx/releases/appcompat) [Accessed: 10-Dec- 2024].
11. Google LLC, "ConstraintLayout," [Online]. Available: [https://developer.android.com/jetpack/androidx/releases/constraintlayout.](https://developer.android.com/jetpack/androidx/releases/constraintlayout) [Accessed: 10- Dec-2024].
12. OpenWeather, "OpenWeatherMap: Weather API," [Online]. Available: https://openweathermap.org/api. [Accessed: 10-Dec-2024].
13. TutorialsAndroid, "GButton: A customizable button library for Android," [Online]. Available: [https://github.com/TutorialsAndroid/GButton.](https://github.com/TutorialsAndroid/GButton) [Accessed: 10-Dec-2024].