Kingdom of Saudi Arabia	المملكة العربية السعودية
Ministry of Education	وزارة التعليم
Qassim University	جامعة القصيم
College of Computer	كلية الحاسب
Information Technology Dept.	قسم تقنية المعلومات

IT342 | Course Project

StopWatch: Time Management App

Students:

Ali Al-Dhlaan	431107785
Abdulmjeed Aldawish	431109432
Waleed Alharbi	431108129

Group8

Section:4578

Date:3/5/2025

Supervisor:

Dr. Muhammad Ali Hamouda

Table of Content

Abstract

StopWatch is a feature-rich Android application designed to provide users with an intuitive and efficient way to measure time. The app combines a stopwatch with lap functionality and a countdown timer, catering to athletes, fitness enthusiasts, and professionals who require precise time tracking. Developed using **Java** and **XML** in **Android Studio**, the app ensures smooth performance with real-time updates. Key features include:

- **Stopwatch** with start, pause, reset, and lap recording.
- **Countdown Timer** with customizable hours, minutes, and seconds.
- **User Authentication** (login/signup) for personalized sessions.
- **Clean UI** with responsive controls and easy navigation.

This report details the development process, system architecture, challenges faced, and future improvements for the StopWatch app.

---

1 .Introduction

1.1 Background

In various fields, such as sports, wellness, cooking, and productivity, proper time measuring tools are crucial. Though numerous apps provide stopwatch and timer features, they come with extra functionality that is not necessary and takes away from the user experience. StopWatch was .developed to provide a simple, yet effective and light solution for precise time measuring

1.2 Objective

The overall purpose of StopWatch is to:

•Provide an extremely responsive stopwatch with lap capture.

•Provide a variable countdown timer for various applications.

•Provide easy navigation with few distractions.

•Securely authenticate individual use.

1.3 Relevance

Unlike bulky timer software, StopWatch is focused on performance and efficiency and therefore best fits:

•Competitors and Trainers (monitoring lap durations during practice sessions).

•Fitness Enthusiasts (balancing exercise and rest periods).

•Students & Professionals (organizing study sessions or work).

1.4 Target Audience

•People need a fast and precise stopwatch/timer.

•Users of minimalist design who prefer no ads or clutter.

2 .Project Objectives

The primary goals of this program were:

1 .Design a complete functional stopwatch with:

- Start, stop, and resume controls.
- Lap recording and clear laps functionality.

2 .Add a countdown timer with:

- Customizable time entry (HH:MM:SS).
- tVisual indication when time elapses.

3 .Implement user authentication (login/signup) with Firebase.

4 .Design an intuitive UI with:

- Clear button designs.
- Seamless switching between timer and stopwatch modes.

5 .Deliver real-time performance with minimal lag.

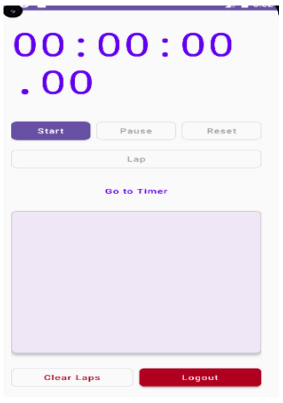
---

3. App Description and Features

3.1 Stopwatch Functionality

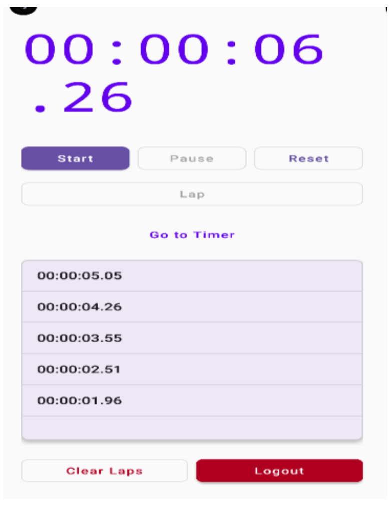
Insert Function (Lap Recording)

- **What data is added**: Lap times (formatted as HH:MM:SS.ms).
- **User action**: Tapping the **"Lap"** button records the current time.



Read Function (Viewing Laps)

- **How data is retrieved**: Laps are stored in a RecyclerView and dynamically updated.



00:00:05.05
00:00:04.26
00:00:03.55
00:00:02.51
00:00:01.96

Update Function (Resetting Laps)

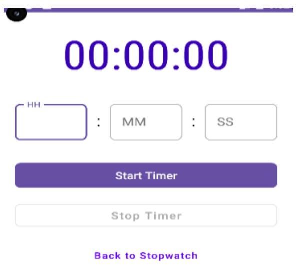
- **User action**: Pressing **"Clear Laps"** removes all recorded times.

Delete Function (Resetting Stopwatch)

- **User action**: Pressing **"Reset"** sets the stopwatch to 00:00:00.00.
- Viewing Laps picture shows the reset state.

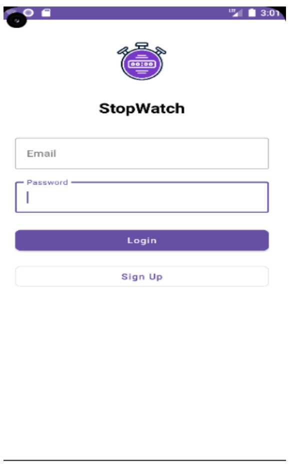
3.2 Timer Functionality

- **User Input**: Time is set via numeric input (HH, MM, SS).
- **Controls**:
  - **Start Timer**: Begins countdown.
  - **Stop Timer**: Pauses the countdown.



.3.3 User Authentication

- **Login/Signup**: Users can create accounts or log in
- **Logout**: Securely ends the session.



4. Technologies Used

Technology	Purpose
Java	Backend logic for stopwatch, timer, and UI interactions.
XML	Frontend layout design (buttons, text views, etc.).
Android Studio	Primary IDE for development and debugging.
Firebase	User authentication (login/signup).
RecyclerView	Efficiently displays lap times.

5. System Design

5.1 Architecture

- **Model-View-Controller (MVC)**:
    - **Model**: Handles time calculations (Java).
    - **View**: UI elements (XML layouts).
    - **Controller**: Manages user inputs (button clicks).

5.2 Database Structure (Firebase)

- **Users Collection**: Stores email and password (hashed).
- **No local storage**: Laps are temporary (cleared on reset).

5.3 UI Flow Diagram

1. **Login/Signup** → **Stopwatch/Timer Screen** → **Lap Management** → **Logout**.

---

6. User Interface (UI) and User Experience (UX)

- **Color Scheme**: Dark background with bright accent buttons for visibility.
- **Navigation**:
    - Tab-based switching between stopwatch and timer.
    - Clear labels for all actions.
- **Responsive Design**: Adapts to different screen sizes.

---

7. Testing and Debugging

7.1 Testing Methods

- **Manual Testing**: Verified button responsiveness, lap accuracy, and timer precision.
- **Edge Cases**:
    - Timer behavior at 00:00:00.
    - Lap recording during pause.

7.2 Bugs Fixed

- **Issue**: Timer sometimes skipped seconds.
  Solution: Used Handler for millisecond precision.

- **Issue**: Laps overflowed on small screens.
  Solution: Implemented RecyclerView scrolling.

---

8. Challenges & Solutions

Challenge	Solution
Accurate millisecond tracking	Used System.currentTimeMillis() for precision.
User session persistence	Integrated Firebase Auth.
UI lag during lap updates	Optimized RecyclerView adapter.

---

9. Future Improvements

1. **Cloud Sync**: Save laps/times to Firebase for cross-device access.
2. **Voice Control**: "Start/Stop" via voice commands.
3. **Themes**: Customizable color schemes.
4. **Notifications**: Alert when timer finishes.

---

10. Conclusion

The StopWatch project succeeded in its primary objectives by delivering a functional and user-friendly Android application for precise time measurement. Some of the key accomplishments include the implementation of a high-accuracy stopwatch with lap recording, a customizable countdown timer by the user, and Firebase-based secure authentication. Through this project, we could acquire practical experience in Java and XML coding, real-time UI updating, and Firebase integration, and apply MVC architecture for reusable code structuring.

Although the app is solid as is, there are some aspects that can be enhanced in subsequent releases. Improved data persistence by syncing lap history to the cloud would enhance usabilty across devices. Voice commands would potentially make the app more accessible, and customizable themes would enhance personalization. Overall, this project laid a good foundation in mobile development, highlighting both successful strategies and room for improvement.

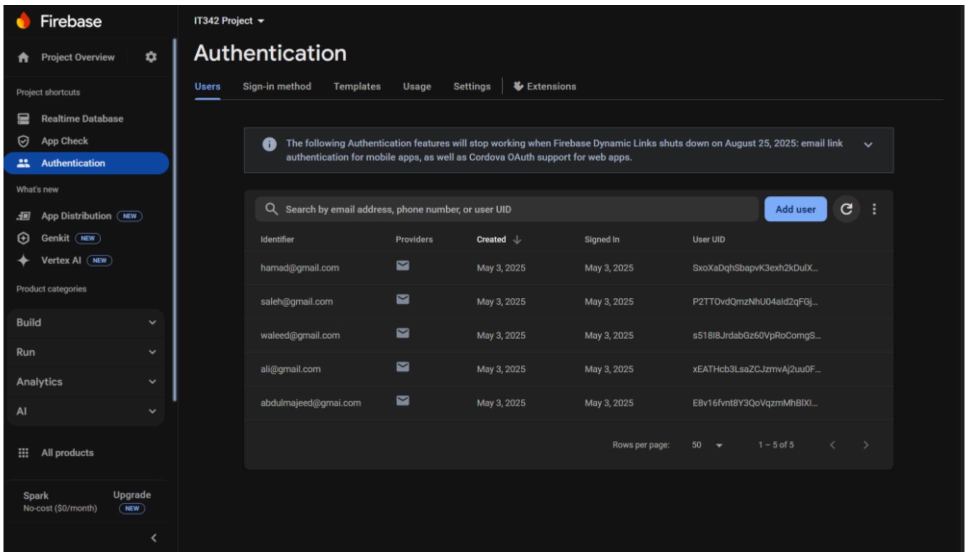
11. References

1. **Android Developer Documentation**
    - Google. (2023). *Android Developers*. Retrieved from https://developer.android.com
2. **Firebase Authentication**
    - Google. (2023). *Firebase Authentication*. Retrieved from https://firebase.google.com/docs/auth
3. **Material Design Guidelines**
    - Google. (2023). *Material Design for Android*. Retrieved from https://material.io/develop/android
4. **Java Concurrency & Performance Optimization**
    - Oracle. (2023). *Java Documentation*. Retrieved from https://docs.oracle.com/en/java/
5. **Mobile App Testing Strategies**
    - Microsoft. (2023). *Mobile App Testing Best Practices*. Retrieved from https://learn.microsoft.com/en-us/appcenter/test-cloud/

Appendix :

Screenshot 1: Firebase Authentication - User list in Firebase. This screenshot shows the users authenticated in the Firebase Authentication system. Each user has an associated email, creation date, and unique User ID. This is essential for managing and verifying users in the StopWatch app

.



Authentication

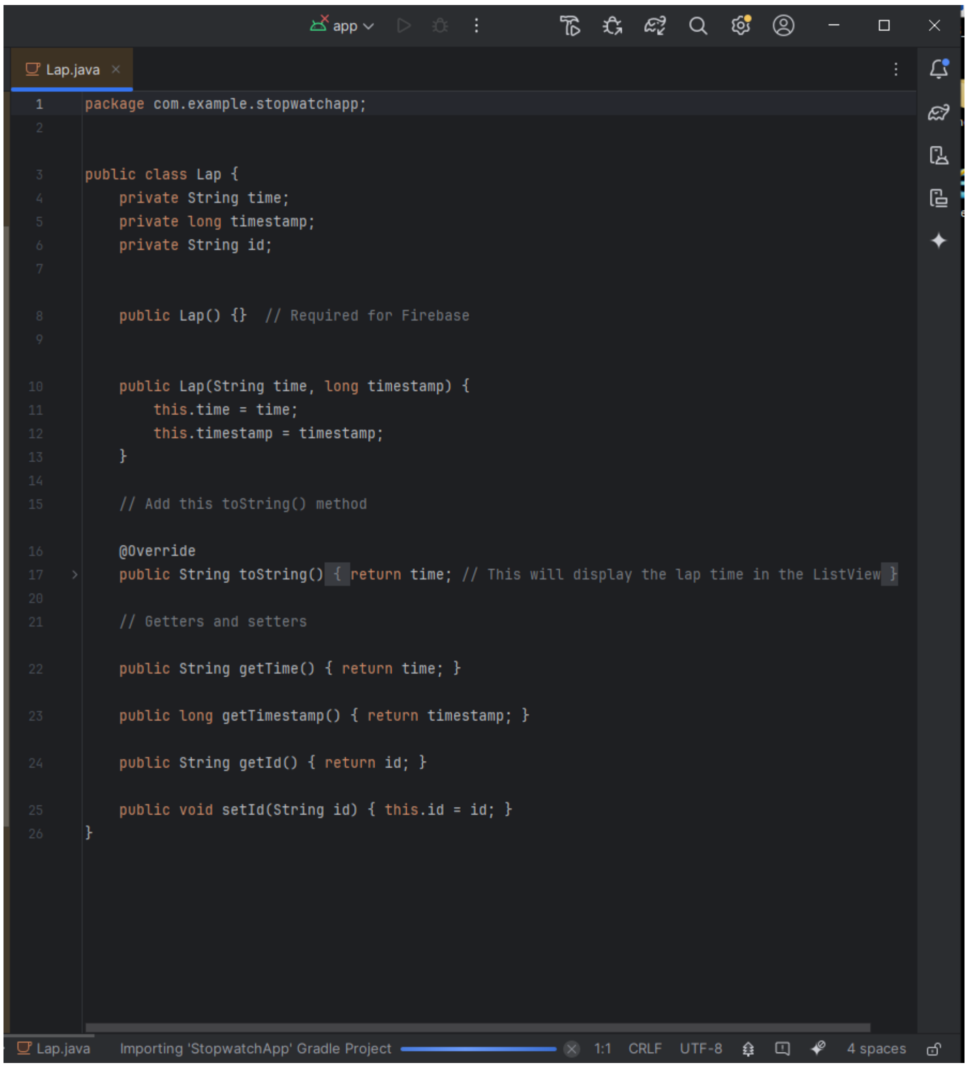
The following Authentication features will stop working when Firebase Dynamic Links shuts down on August 25, 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.

Search by email address, phone number, or user UID

Identifier	Providers	Created	Signed In	User UID
hamad@gmail.com	✉	May 3, 2025	May 3, 2025	SxoXaDqhSbapvK3exh2kDulX...
saleh@gmail.com	✉	May 3, 2025	May 3, 2025	P2TTOvdQmzNhU04ald2qFGj...
waleed@gmail.com	✉	May 3, 2025	May 3, 2025	s518I8JrdabGz60VpRoComgS...
ali@gmail.com	✉	May 3, 2025	May 3, 2025	xEATHcb3LsaZCJzmvAj2uu0F...
abdulmajeed@gmai.com	✉	May 3, 2025	May 3, 2025	E8v16fvnt8Y3QoVqzmMhBIXI...

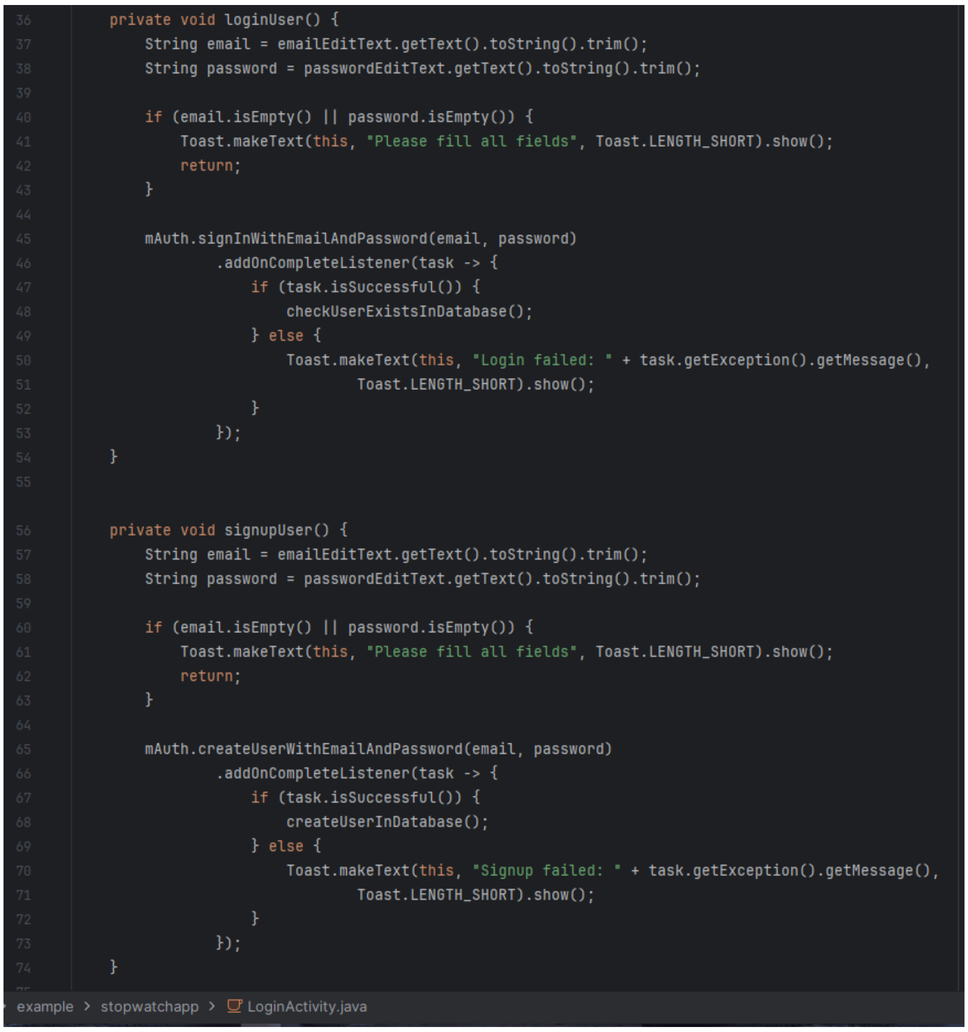
Rows per page: 50 1 – 5 of 5

Screenshot 2: Java Class - Lap.java class for managing lap data in the StopWatch app. This class defines the properties and methods for handling lap times, including a timestamp and lap time. It is an important part of the app's functionality for tracking and displaying laps.



```java
package com.example.stopwatchapp;


public class Lap {
    private String time;
    private long timestamp;
    private String id;


    public Lap() {}  // Required for Firebase


    public Lap(String time, long timestamp) {
        this.time = time;
        this.timestamp = timestamp;
    }

    // Add this toString() method

    @Override
    public String toString() { return time; // This will display the lap time in the ListView }

    // Getters and setters

    public String getTime() { return time; }

    public long getTimestamp() { return timestamp; }

    public String getId() { return id; }

    public void setId(String id) { this.id = id; }
}
```

Screenshot 3: Firebase Authentication code snippet - Handles user login and signup functionality in Firebase. This code connects the app with Firebase Authentication, enabling users to log in or sign up with their email and password. It also manages success and failure scenarios for login and signup actions.



```java
    private void loginUser() {
        String email = emailEditText.getText().toString().trim();
        String password = passwordEditText.getText().toString().trim();

        if (email.isEmpty() || password.isEmpty()) {
            Toast.makeText(this, "Please fill all fields", Toast.LENGTH_SHORT).show();
            return;
        }

        mAuth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        checkUserExistsInDatabase();
                    } else {
                        Toast.makeText(this, "Login failed: " + task.getException().getMessage(),
                                Toast.LENGTH_SHORT).show();
                    }
                });
    }


    private void signupUser() {
        String email = emailEditText.getText().toString().trim();
        String password = passwordEditText.getText().toString().trim();

        if (email.isEmpty() || password.isEmpty()) {
            Toast.makeText(this, "Please fill all fields", Toast.LENGTH_SHORT).show();
            return;
        }

        mAuth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        createUserInDatabase();
                    } else {
                        Toast.makeText(this, "Signup failed: " + task.getException().getMessage(),
                                Toast.LENGTH_SHORT).show();
                    }
                });
    }
```