VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590018

A Java for Mobile Application Development Mini Project Report on

"Athletic Posture Enhancement Application"

Submitted in partial fulfillment of the requirements for the VI semester and award of the degree of Bachelor of Engineering in AI & ML of Visvesvaraya Technological University, Belagavi

Submitted by:

Vishnu Kashyap - 1RN20AI062

Under the Guidance of:

Dr. Rama Satish K V
Associate Professor
Department of AI & ML

ESTD : 2001
An Institute with a Difference

Department of AI & ML
RNS Institute of Technology
Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560 098
2022-2023

RNS Institute of Technology

Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560098

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

(NBA Accredited for academic years 2018-19, 2019-20, 2020-22)

ESTD : 2001

An Institute with a Difference

CERTIFICATE

Certified that the mini project work entitled **"Athletic Posture Enhancement Application"** has been successfully carried out by **Vishnu Kashyap** bearing USN **"1RN20AI062"**, bonafide student of **"RNS Institute of Technology"** in partial fulfillment of the requirements for the 6th semester of **"Bachelor of Engineering in Artificial Intelligence and Machine Learning Engineering of Visvesvaraya Technological University"**, Belagavi, during academic year 2022- 2023. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the **Mobile Application Development** laboratory requirements of 6th semester BE in AI and ML.

Signature of the Guide	Signature of the HoD	Signature of the Principal
Dr. Rama Satish K V Associate Professor Dept. of AI and ML RNSIT, Bengaluru	**Dr. Harsha S** Head, Dept. of AI and ML RNSIT, Bengaluru	**Dr. H S Ramesh Babu** Principal RNSIT, Bengaluru

Name & Signature

Examiner 1:

Examiner 2:

Acknowledgement

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project work. We would like to take this opportunity to thank them all.

We are grateful to **Management**, **Dr. M K Venkatesha**, Director and **Dr. H S Ramesh Babu** Principal, RNSIT, Bangalore, for their support towards completing this mini project.

We would like to thank **Dr. Harsha S**, Head, Department of Artificial Intelligence and Machine Learning Engineering, RNSIT, Bangalore, for his valuable suggestions and expert advice.

We deeply express our sincere gratitude to our guide **Dr. Rama Satish K V**, Associate Professor, Department of AI & ML, RNSIT, Bangalore, for his able guidance, regular source of encouragement and assistance throughout this project.

We would like to thank all the teaching and non-teaching staff of the Department of Artificial Intelligence and Machine Learning Engineering, RNSIT, Bengaluru for their constant support and encouragement.

Vishnu Kashyap

1RN20AI062

Abstract

The posture of an athlete is always an important characteristic when it comes to their mobility, agility, physical fitness and strength. The body language of an athlete or any other human being is always assessed by perfect posture. As such comes the essential requirement of Good Posture Practices (GPP).

This application caters the key requirement of the analysis of an athlete's posture without the requirement of any additional computational hardware or daily visits to the physician but at the comfort of one's own smartphone running Android OS.

The widespread use of the freeware Android OS had enabled several up-and-coming developers such as myself to utilize the design and infrastructure supported by Android in creating custom built applications. The Athletic Posture Enhancement application is one such customized program which runs on the Pose Detection Module presented by the Google ML Kit.

This app promotes the idea of self-assessment with features such as camera access to both the front and rear cameras which displays the human body structure as a silhouette of nodal body joints which is accompanied by a value indicating the accuracy of the posture ranging from 0.5 to 1.0

The application is also secured by a Login/Register infrastructure supported by Google Firebase which allows access only to the registered users to use the ML features provided. The application also features GPP, a video tutorial of how to work the pose detection and a Calendar Event Access which enables users to set up a time for analyzing their posture daily.

Programming languages such as Java, XML (eXtensible Markup Language) and Firebase authentication procedures are the backbone of the application. All in all, this application provides users an access to an ML algorithm built in an easily portable and compact Application.

Index

List of Figures

Chapter 1

Introduction

1.1 Overview of Mobile Application Development

Mobile application development refers to the process of creating software applications that are specifically designed to run on mobile devices such as smartphones and tablets. It involves a combination of programming languages, frameworks, and tools to develop applications for various mobile operating systems like Android and iOS.

The process of mobile application development typically includes several stages:

- **Planning and Conceptualization**: This involves defining the app's purpose, target audience, and core features. It also includes market research, competitor analysis, and creating a solid plan for development.
- **Design**: In this stage, the app's user interface (UI) and user experience (UX) are designed. It includes creating wireframes, mockups, and prototypes to visualize the app's layout, navigation, and overall design aesthetics.
- **Development**: Developers write the actual code to bring the app to life. They utilize programming languages like Java or Kotlin for Android development and Swift or Objective-C for iOS development. Frameworks and development tools like Android Studio and Xcode are often used to streamline the development process.
- **Testing**: This phase involves rigorous testing of the application to ensure it functions correctly, is free of bugs and glitches, and provides a smooth user experience. Testing may include functional testing, performance testing, usability testing, and compatibility testing across different devices.
- **Deployment**: Once the app is thoroughly tested and deemed ready for release, it is submitted to the respective app stores (such as Google Play Store or Apple App Store) for approval. The app is then made available for download and installation by users.
- **Maintenance and Updates**: After the app is launched, it requires ongoing maintenance and updates. This includes fixing bugs, optimizing performance, adding new features, and ensuring compatibility with new OS versions or devices.

Mobile application development can be done using native development, where separate codebases are created for each platform, or through cross-platform development, where a single codebase is used to build applications that can run on multiple platforms.

The demand for mobile applications continues to grow as smartphones become an integral part of our daily lives. Businesses and individuals alike leverage mobile app development to provide services, engage with customers, streamline processes, and enhance user experiences.

1.2 History of Mobile Application Development

The history of mobile application development dates back to the early 2000s when mobile phones started evolving into more than just devices for making calls and sending text messages. Here's a brief overview of the key milestones in the history of mobile application development:

- ❖ **Pre-Smartphone Era**: Before the emergence of smartphones, mobile applications were limited to basic functionalities like calculators, calendars, and games. These applications were mostly built using programming languages specific to each mobile platform.

- ❖ **Introduction of App Stores**: In 2008, Apple introduced the App Store for its newly launched iPhone, creating a centralized marketplace for mobile applications. This move revolutionized the industry by providing developers with a platform to distribute and monetize their applications.

- ❖ **Rise of Android**: Google's Android operating system gained popularity as an open-source alternative to iOS. Android Market (later renamed Google Play Store) was launched in 2008, enabling developers to reach a wider user base.

- ❖ **Expansion of App Ecosystem**: With the increasing popularity of smartphones, app development flourished. Social media, productivity, gaming, and entertainment apps became widespread, catering to diverse user demands.

- ❖ **Advancements in Cross-Platform Development**: Cross-platform development frameworks like PhoneGap (now Apache Cordova), Xamarin, and React Native emerged, allowing developers to write code once and deploy it on multiple platforms, reducing development time and costs.

- **Integration of Advanced Technologies**: Mobile app development witnessed the integration of advanced technologies like augmented reality (AR), virtual reality (VR), artificial intelligence (AI), and machine learning (ML), expanding the possibilities for app functionalities and user experiences.

Additionally, the history of mobile application development has witnessed significant shifts in design principles and development practices. As users became more demanding and sophisticated, app developers began prioritizing intuitive user interfaces, seamless navigation, and personalized experiences. The advent of cloud computing and improved internet connectivity played a vital role in shaping mobile app development. It facilitated the storage and retrieval of data from remote servers, enabling real-time synchronization and enabling apps to leverage the power of the cloud for enhanced functionality and data management.

1.3 Applications of Mobile Application Development

Mobile applications, or apps, have a wide range of uses and have become an integral part of our daily lives. They offer convenience, accessibility, and functionality that cater to various needs and interests. Let's explore the different uses of mobile applications:

- **Communication and Social Networking**:
  Mobile apps have transformed the way we communicate and connect with others. Social networking apps like Facebook, Instagram, and Twitter provide platforms for sharing updates, photos, and videos with friends and followers. Messaging apps like WhatsApp, WeChat, and Telegram enable instant messaging, voice calls, and video calls, facilitating seamless communication across distances. These apps have revolutionized social interactions, allowing people to stay connected regardless of geographical barriers.

- **E-commerce and Online Shopping**:
  Mobile apps have revolutionized the retail industry by providing users with a convenient and seamless online shopping experience. E-commerce apps such as Amazon, eBay, and

Alibaba allow users to browse and purchase products from the comfort of their smartphones. These apps offer features like personalized recommendations, secure payment options, order tracking, and easy returns, enhancing the overall shopping experience. Mobile payment apps like Apple Pay, Google Pay, and PayPal enable secure and convenient transactions, further facilitating online shopping.

- **Entertainment and Media Streaming**:

Mobile applications offer a plethora of options for users to consume media content and indulge in entertainment. Video streaming apps like Netflix, YouTube, and Disney+ provide on-demand access to movies, TV shows, and documentaries. Music streaming apps like Spotify, Apple Music, and SoundCloud offer vast libraries of songs and personalized playlists. Gaming apps have also gained immense popularity, with titles like Candy Crush, PUBG Mobile, and Fortnite captivating millions of users worldwide. These apps provide entertainment and leisure on the go, catering to various interests and preferences.

- **Travel and Navigation**:

Mobile apps have transformed the way we travel and navigate our surroundings. Travel apps like Booking.com, Airbnb, and TripAdvisor facilitate hotel bookings, flight reservations, and travel planning. Navigation apps like Google Maps, Waze, and Apple Maps provide real-time directions, traffic updates, and points of interest, ensuring smooth navigation in unfamiliar locations. Additionally, ride-sharing apps like Uber and Lyft have revolutionized the transportation industry, offering convenient and cost-effective alternatives to traditional taxis.

- **Productivity and Business Tools**:

Mobile applications have become essential tools for enhancing productivity and streamlining business processes. Productivity apps like Microsoft Office Suite, Google Docs, and Evernote enable users to create, edit, and collaborate on documents, spreadsheets, and presentations. Task management apps like Trello, Asana, and Todoist help individuals and teams organize and prioritize their work. Communication and collaboration apps like Slack and Microsoft Teams facilitate seamless teamwork and communication, particularly in remote work environments. Furthermore, mobile banking apps, project management apps, and CRM apps have simplified financial management, project coordination, and customer relationship management for businesses.

- **Health and Fitness**:

Mobile apps have had a significant impact on the health and fitness industry, empowering individuals to track and manage their well-being. Fitness tracking apps like Fitbit, Strava, and MyFitnessPal monitor physical activity, count steps, and track workouts. Health apps provide resources for meditation, sleep tracking, and stress management. Telemedicine apps facilitate remote consultations with healthcare professionals, providing access to medical advice and virtual healthcare services.

- **Education and Learning**:

Mobile applications have transformed the education sector by providing access to knowledge and learning resources on mobile devices. Educational apps like Duolingo, Khan Academy, and Coursera offer courses, tutorials, and quizzes on various subjects. Language learning apps facilitate language acquisition, while coding apps teach programming skills. E-book apps provide a digital library of books, making reading more accessible. These apps enable individuals to learn at their own pace and expand their knowledge in diverse fields.

- **Utility and Tools**:

Mobile applications offer various utility and tool-based functionalities. Weather apps provide real-time weather updates and forecasts. Note-taking apps like Evernote and Google Keep help users capture and organize their thoughts and ideas. Language translation apps facilitate real-time translation of text and speech. Barcode scanning apps allow users to scan and compare prices of products. These utility apps cater to specific needs and tasks, making daily life more convenient and efficient.

Chapter 2

Introduction to Android

Android Studio is the official integrated development environment (IDE) for Android app development. It provides a comprehensive set of tools, including a code editor, layout editor, performance analysis, emulator, and build system. With seamless integration with the Android SDK, it enables developers to create, test, and deploy high-quality Android applications. Android Studio supports multiple programming languages, such as Java and Kotlin, and offers features like version control integration and instant app deployment. It is a powerful and constantly evolving tool that empowers developers to build innovative and feature-rich Android apps.

2.1 Android Versions

The development of the Android operating system was started in 2003 by Android, Inc. Later on, it was purchased by Google in 2005. The beta version of Android OS was released on November 5, 2007, while the **software development kit (SDK)** was released on November 12, 2007.

The first Android mobile that was publicly released with Android 1.0 was the T-Mobile G1 (aka HTC Dream) in October 2008.

Google announced in August 2019 that they were ending the confectionery scheme, and they use numerical ordering for future Android versions. The first Android version which was released under the numerical order format was Android 10.

Figure 2.1 Android Versions

Figure 2.1 depicts all the android operating system versions released thus far. We are currently in a period where Android 13 is on the brink of release for all smartphones.

2.2 Android Architecture

Android software contains an open-source Linux Kernel having collection of number of C/C++ libraries which are exposed through an application framework service.

Among all the components Linux Kernel provides main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide platform for running an android application. Figure 2.2 shows a diagrammatic representation of the layers within an Android OS.

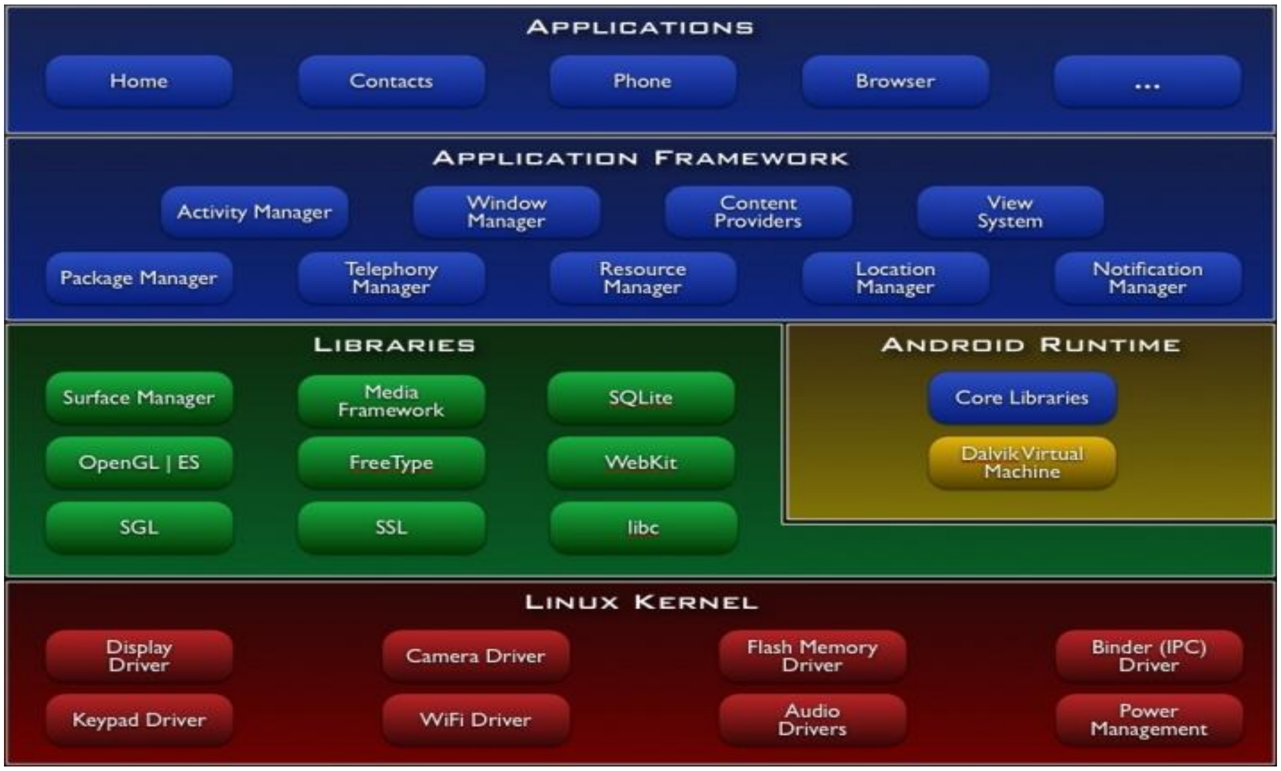


Figure 2.2 Android Architecture

- **Applications**:

Applications is the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc. and third-party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only.

- **Application framework:**

Application Framework provides several important classes which are used to create an Android application. It provides a generic abstraction for hardware access and also helps in managing the user interface with application resources.

- **Application runtime**:

Android Runtime environment is one of the most important parts of Android. It contains components like core libraries and the Dalvik virtual machine (DVM). Mainly, it provides the base for the application framework and powers our application with the help of the core libraries.

Like Java Virtual Machine (JVM), Dalvik Virtual Machine (DVM) is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently. SQLite provides database support and FreeType provides font support.

Linux Kernel is heart of the android architecture. It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime.

2.3 Android Studio Installation

To install Android Studio, follow these steps:

- Download Android Studio: Go to the official Android Studio website (https://developer.android.com/studio) and click on the "Download" button. Choose the appropriate version for your operating system (Windows, macOS, or Linux).
- Choose Installation Options: The installer will present you with various installation options. You can select the components you want to install, such as the Android SDK, virtual devices, and additional tools. Make your desired selections and click "Next."
- Android Virtual Device (AVD) Setup: After the installation, you will be prompted to set up an Android Virtual Device (AVD) for testing your apps on the emulator. You can create or import an existing AVD or skip this step for now.
- Setup Android SDK: On the first launch, Android Studio will prompt you to import settings from a previous installation or set up a new Android SDK. Choose the option to set up a new Android SDK.
- SDK Components Setup: Android Studio will guide you through the process of downloading and installing the necessary SDK components for app development. Follow the on-screen instructions and wait for the components to be downloaded and installed.
- Configuration and Updates: Once the SDK components are installed, you can configure additional settings and check for updates. Android Studio will provide options to set up a theme, enable usage statistics reporting, and more. Customize the settings according to your preferences.
- Start Developing: After completing the setup process, Android Studio is ready for use. You can create new projects and begin developing applications.

Chapter 3

Resource Requirements

3.1 Hardware Requirements

The Hardware requirements are very minimal and the program can be run on most of the machines. The following are the hardware requirements:

For developers:	For testers:
Processor: AMD Ryzen 5 7600 or better	Processor: Intel i5 6th generation or above
RAM: 32GB DDR5	RAM: 8GB DDR4 or higher
Storage Space: 40GB	Storage Space: 40GB

3.2 Software Requirements

The software requirements are description of features and functionalities of the system. The following are the software requirements:

- Operating System: Windows 8.1 or above
- IDE: Android Studio
- Framework: Microsoft .Net Framework v4.0 or above
- Android OS: 4.4 KitKat or above.

The same applies to both developers and testers.

Chapter 4

Introduction to the Project

4.1 Overview of the Project

Athletic Posture Enhancement Application (APE APP) is the project designed to cater the posture analysis of an athlete to improve their agility and mobility. It is an easily accessible, compact and portable application which enables users to self-assess their own posture as well as posture of somebody else.

This project is backed by the Pose Detection module included in the Google ML Kit which is a pre trained Machine Learning model which quickly captures the object fed into the algorithm through the camera display. Additionally, the project is also secured by Google Firebase's login/registration authentication systems.

4.2 Aim of the Project

The aim of the APE APP is to provide any user (primarily focussing on athletes) the feature to enhance their own posture at the comfort of their own device. This also ensures proper exposure for interested users on how Machine Learning Algorithms are taking over the application space to suit every human requirement, now available on our very own smart phones. The application also renders this data and enables the users to store and track their daily posture changes.

Additionally, the project is secured and authenticated by a Login/Register module only ensuring registered access to this software along with appropriate application permissions to open the camera, read storage spaces and read calendar events.

All these features collectively ensure a basic requirement for Good Posture Practices along with a complex ML code to suit this requirement with an attractive UI design.

Chapter 5

Implementation

5.1 How to use the Application:

It is very easy to use the application and the following steps are followed:

- Install the app via wireless debugging on your android device’s developer options
- You will be redirected to the login page. If your details don’t already exist in the database [i.e., if you are a new user], you will have to enter your details to register.
- The regular expression expects a signup email of the form username@email.com.
- The password is expected to have at least 6 characters.
- Once the registration process has been completed successfully, you will be redirected to the login page. Entering the same credentials will redirect you to the dashboard that consists of six buttons.
- The button options are the core of this project. The first and second button gives the user access to the front and rear camera respectively backed up by the Pose Detection Module. There is also an additional feature of “click” which stores the screen shot of the posture silhouette in the Internal Storage of a device.
- The third button leads you to a scrolling activity consisting of information of good posture practices. Each of these images also have an onClick action which leads to their webpages for more information on them.
- The fourth button is a video demonstration on how to use the application with respect to pose detection and saving the posture data with the first two button options.
- The fifth button leads to setting up a Calendar Event which by default sets up a Posture Analysis event five minutes later from the current time and of the duration of another five minutes to completely block time for Posture Analysis with this app.
- The sixth button is the final button which caters the Logout option and redirects the user to the landing page while logging out of the dashboard.

5.2 Code:

MainActivity.java:

```
public class MainActivity extends AppCompatActivity {
    Button btn;
    TextView txt1,txt2;
    ImageView im1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_main);
        btn = findViewById(R.id.getStartedButton);
        txt1 = findViewById(R.id.titleTextView);
        txt2 = findViewById(R.id.descriptionTextView);
        im1 = findViewById(R.id.logoImageView);

        btn.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View view){
                 Animation anim
                 =AnimationUtils.loadAnimation(MainActivity.this,R.anim.bounce);
                 //ANIMATION BEGINS HERE
                 btn.startAnimation(anim);
                 Intent i = new Intent(MainActivity.this, login.class );
                 startActivity(i);
            }
        });
    }
}
```

register.java:

```
public class Register extends AppCompatActivity {
    private EditText emailTextView, passwordTextView;
    private Button Btn;
    private FirebaseAuth mAuth;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        // taking FirebaseAuth instance
        mAuth = FirebaseAuth.getInstance();
        // initialising all views through id defined above
        emailTextView = findViewById(R.id.email);
        passwordTextView = findViewById(R.id.passwd);
        Btn = findViewById(R.id.btnregister);
        // Set on Click Listener on Registration button
        Btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v)
            {
                registerNewUser();
            }
        });
    }
    private void registerNewUser()
    {
        String email, password;
        email = emailTextView.getText().toString();
        password = passwordTextView.getText().toString();
        if (TextUtils.isEmpty(email)) {
            Toast.makeText(getApplicationContext(),
                           "Please enter email!!",
                           Toast.LENGTH_LONG)
                    .show();
            return;
        }
        if (TextUtils.isEmpty(password)) {
            Toast.makeText(getApplicationContext(),
                           "Please enter password!!",
                           Toast.LENGTH_LONG)
                    .show();
            return;
        }
        mAuth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task)
                    {
                        if (task.isSuccessful()) {
                            Toast.makeText(getApplicationContext(),
                                           "Registration successful!",
                                           Toast.LENGTH_LONG)
                                    .show();

                            Intent intent
                                    = new Intent(Register.this,
                                    login.class);
                            startActivity(intent);
                        }
                        else {

                            // Registration failed
                            Toast.makeText(
                                           getApplicationContext(),
                                           "Registration failed!!"
                                                   + " Please try again later",
                                           Toast.LENGTH_LONG)
                                    .show();

                        }
                    }
                });
    }
}
```

login.java:

```
public class login extends AppCompatActivity {

    private EditText emailTextView, passwordTextView;
    private Button Btn,Btn2;
    private FirebaseAuth mAuth;
    @Override
```

```
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login2);
    mAuth = FirebaseAuth.getInstance();
    emailTextView = findViewById(R.id.email);
    passwordTextView = findViewById(R.id.password);
    Btn = findViewById(R.id.login);
    Btn2 = findViewById(R.id.register);
    Btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v)
        {
            loginUserAccount();
        }
    });
    Btn2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent i = new Intent(login.this, MainActivity2.class );
            startActivity(i);
        }
    });
}
private void loginUserAccount()
{
    String email, password;
    email = emailTextView.getText().toString();
    password = passwordTextView.getText().toString();
    if (TextUtils.isEmpty(email)) {
        Toast.makeText(getApplicationContext(),
                        "Please enter email!!",
                        Toast.LENGTH_LONG)
                .show();
        return;
    }

    if (TextUtils.isEmpty(password)) {
        Toast.makeText(getApplicationContext(),
                        "Please enter password!!",
                        Toast.LENGTH_LONG)
                .show();
        return;
    }
    mAuth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(
                    new OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(
                                @NonNull Task<AuthResult> task)
                        {
                            if (task.isSuccessful()) {
                                Toast.makeText(getApplicationContext(),
                                                "Login successful!!",
                                                Toast.LENGTH_LONG)
                                        .show();
                                Intent intent
                                        = new Intent(login.this,
                                        Dashboard.class);
```

```
                                  startActivity(intent);
                              }
                              else {
                                   Toast.makeText(getApplicationContext(),
                                                  "Login failed!!",
                                                  Toast.LENGTH_LONG)
                                          .show();
                              }
                          }
                      });
    }
}
```

ML.java:

```
public class Dashboard extends AppCompatActivity{
    ImageButton btnz,btnz2,btnz3,btnz4,btnz5;
    private FirebaseAuth mAuth;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);
        btnz = findViewById(R.id.imageButton3);
        btnz2 = findViewById(R.id.imageButton5);
        btnz3 = findViewById(R.id.imageButton);
        btnz4 = findViewById(R.id.imageButton2);
        btnz5 = findViewById(R.id.imageButton6);
        btnz.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(Dashboard.this,PoseDetectionActivity.class );
startActivity(i);
            }
        });
        btnz2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(Dashboard.this,
PoseDetectionActivityFRONT.class );
                startActivity(i);
            }
        });
        btnz3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(Dashboard.this, ImageScrolling.class);
                startActivity(i);
            }
        });
        btnz4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(Dashboard.this, video.class);
                startActivity(i);
            }
        });
        btnz5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText(getApplicationContext(),
                                "Logged out successfully!",
                                Toast.LENGTH_LONG)
                        .show();
                Intent intent = new Intent(Dashboard.this, MainActivity.class);
startActivity(intent); finish();
            }
        });
    }
    public void calendarEvent(View view) {
        Calendar calendarEvent = Calendar.getInstance();
        Intent i = new Intent(Intent.ACTION_EDIT);
        i.setType("vnd.android.cursor.item/event");
        i.putExtra("beginTime", calendarEvent.getTimeInMillis()+ 60 * 5 * 1000);
        i.putExtra("allDay", false);
        i.putExtra("rule", "FREQ=DAILY");
        i.putExtra("endTime", calendarEvent.getTimeInMillis() + 60 * 10 * 1000);
        i.putExtra("title", "Posture Assessment Reminder");
        startActivity(i);
    }
}
```

PoseDetectionActivity.java:

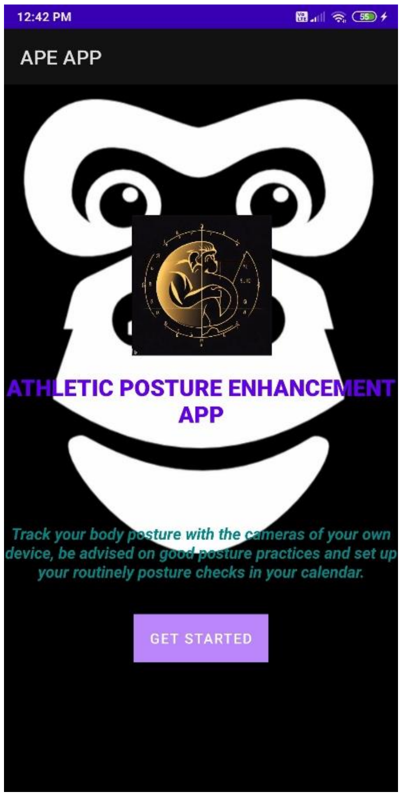
```
public class PoseDetectionActivity extends MLVideoHelperActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
    @Override
    protected VisionBaseProcessor setProcessor() {
        AccuratePoseDetectorOptions options = new AccuratePoseDetectorOptions.Builder()
                .setDetectorMode(AccuratePoseDetectorOptions.STREAM_MODE)
                .build();
        return new PoseDetectorProcessor(
                options, true, false, false, false, true, this, graphicOverlay);
    }
}
```

Video.java:

```
public class video extends AppCompatActivity {
    private VideoView videoView;
       String videoUrl = "
       https://ik.imagekit.io/50cyxejkg/mad_project_Pose.mp4?updatedAt=1689060416159";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_video);
        videoView = findViewById(R.id.videoView);
        Uri uri = Uri.parse(videoUrl);
        videoView.setVideoURI(uri);
        MediaController mediaController = new MediaController(this);
        mediaController.setAnchorView(videoView);
        mediaController.setMediaPlayer(videoView);
        videoView.setMediaController(mediaController);
        videoView.start();
    }
```

Chapter 6

Results


12:42 PM

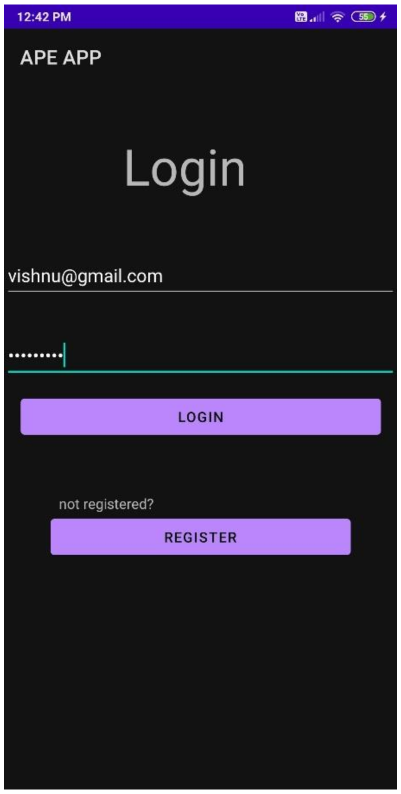
APE APP

ATHLETIC POSTURE ENHANCEMENT APP

Track your body posture with the cameras of your own device, be advised on good posture practices and set up your routinely posture checks in your calendar.

GET STARTED

Fig 6.1


12:42 PM

APE APP

Login

vishnu@gmail.com

•••••••••

LOGIN

not registered?

REGISTER

Fig 6.2

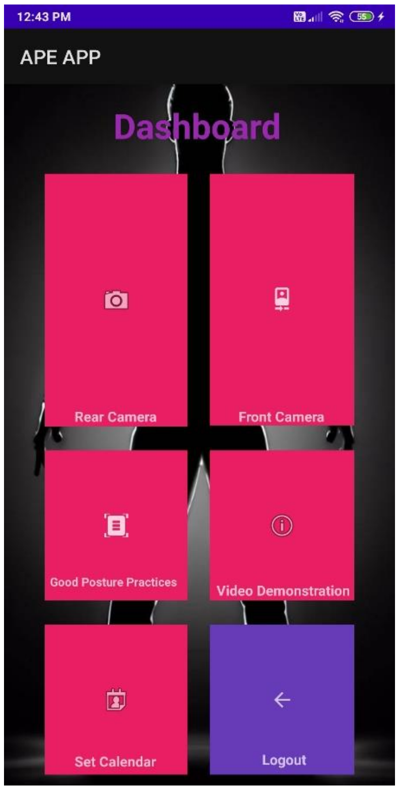


Fig 6.3

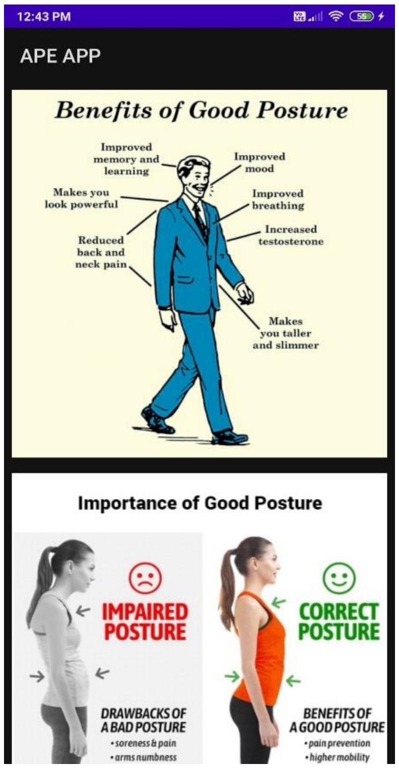
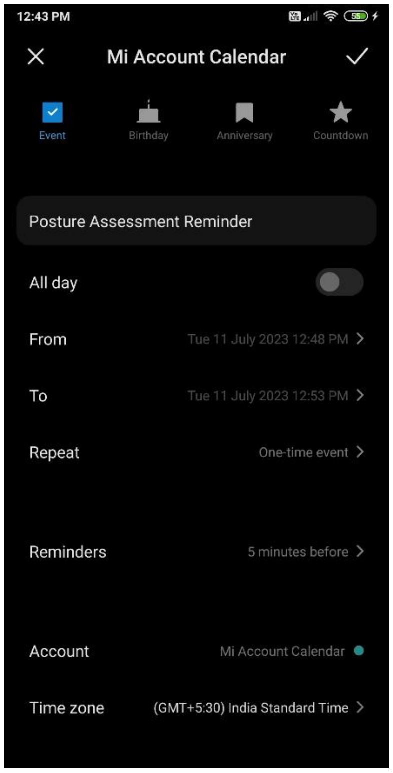


Fig 6.4


12:43 PM

Mi Account Calendar

Event | Birthday | Anniversary | Countdown

Posture Assessment Reminder

All day

From Tue 11 July 2023 12:48 PM

To Tue 11 July 2023 12:53 PM

Repeat One-time event

Reminders 5 minutes before

Account Mi Account Calendar

Time zone (GMT+5:30) India Standard Time

Fig 6.5

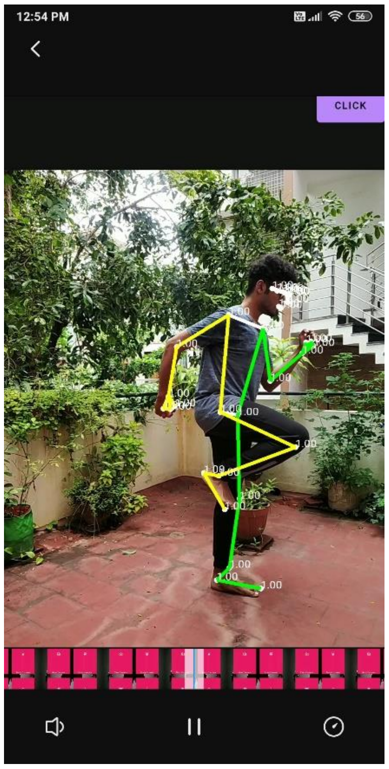


Fig 6.6

- **Fig 6.1: Main Activity –** The startup activity briefs about what the application does along with a Get Started Button with an animation leading to the Login Page.

- **Fig 6.2: Login Page –** This activity is powered by Google Firebase which accepts a valid user login email ID along with the password. If the user isn't registered then the Register button will lead them to the Register Page which accepts new users.

- **Fig 6.3: Dashboard –** This is the landing activity of the application and provides the user with six buttons. The first two buttons direct the user to the usage of Front and Rear Camera respectively providing them access to the camera display and the ML algorithm displays the Pose silhouette over the human object representing pose enhancement aspect of the project. Also, the third button "Good Posture Practices" leads to the PosePractice activity, whereas the fourth button "Video Demonstration" leads to a video presentation in Video View which shows how the front and rear camera access for Pose Detection works. Finally, the fifth button leads to calendar event and the sixth button Logs out the user.

- **Fig 6.4: PosePractice –** This activity is a Scroll View which depicts all the necessary GPP and with a click on these Images will lead to their webpages.

- **Fig 6.5: Calendar –** This activity is a Calendar Event setting activity which leads user to setting up a Pose Assessment reminder five minutes later by default with a default duration of five minutes again for proper pose assessment.

- **Fig 6.6 FrontCamera –** This activity is a sample screenshot of how the front camera button displays the pose detection with a human subject.

Chapter 7

Conclusion and Future Enhancements

7.1 Conclusion

In conclusion, Atheletic Posture Enhancement Application is a comprehensive Android application developed to assist athletes and general user to posture enhancement. With its user-friendly interface and integration of Firebase for secure login and signup, the app provides a personalized and reliable experience. By offering concise details about nodal posture points the users are informed of where to improve their posture and how.

Additionally, the app incorporates Google ML Kit's Pose Detection which is a pre trained model designed to assess posture and the screen shot option, the "click" button is another feature which enables daily tracking of pose data.

This application also comes with a user friendly demonstaration, a scroll page on good posture practices and a calendar event setting feature which serves as a reminder for the user to assess their posture on a daily basis.

Overall, the APE APP caters the human body requirement on a smart phone with precise information and crisp data handling to improve human posture for athletes.

7.2 Future Enhancements

In addition to the mentioned features, the app can be further enhanced with the following additions:

- Custom logs for every athlete user
- Timer option for clicking pose
- Option to switch to wide optic camera
- More secure database infrastructure
- General Layouts can be more User attaractive

References

[1] Beginning Android Programming with Android Studios, 4th Edition, J F DiMarzio

[2] https://stackoverflow.com/

[3] https://developer.android.com/reference/android/app/Application

[4] https://www.tutorialspoint.com/java/index.html

[5] https://openai.com/blog/chatgpt

[6] https://www.youtube.com/

[7] Dawn Griffiths & David Griffiths, Head First Android Development, O‟Reilly, 1 stEdition, 2015