


Raina Narayan

MyFitness App

Android Studio Software:

Current version: Android Studio Meerkat Feature Drop | 2024.3.2 AI-243.25659.59.2432.13423653 April 29, 2025

Emulator:

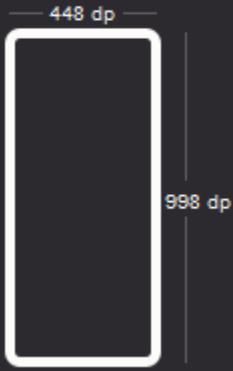
 Pixel 9 Pro XL
Android 16.0 ("Baklava") | x86_64

Pixel 9 Pro XL

Device Info Paired Devices

Summary

API level	36
Resolution (px)	1344 × 2992
Resolution (dp)	448 × 998
Density	480 dpi
ABI list	x86_64
Available storage	
Size on disk	10.6 GB



Test account credentials (all lowercase):

- Username: test
- Password: test

```
password: "test"
username: "test"
```

Contents

Sign Up, Login, and Helper Class.....	3
Profile Activity	4
Edit Profile Activity	5
Home Menu	8
Calendar	9
Meal Planner	10
Workouts	11
Pre-workout	12
Stretches	13
Post-workout	14
Exercises.....	15
Glutes, Core, Legs, Full Body	16

Sign Up, Login, and Helper Class

In my Sign Up Activity, I have created a sign-up menu where users can enter their name, email, username, and password to register. When the user clicks on the sign-up button, it will process all the input values and store them in a Helper Class object. Then I connect to Firebase Realtime Database and save the user data under a child sub class and name it after the entered username. After the data is successfully stored into the database, I display a confirmation message using Toast prompt and redirect the user to the Login Activity screen so the users can log in. I have also included a login text view below the signup button for users who already created an account. When the user clicks the text view, it will direct them to the login screen.

In Login Activity, I implemented a login screen where users can enter their username and password. When the user clicks on the login button, it checks if both fields are not empty and the queries the Firebase RealTime Database to see if a user entered an existing username. Then, it compares the entered password with the one stored in the database once the username matches. I have also implemented a Shared Preference API to save the username and navigate to Profile Activity, passing along the user's name, email, and username for future use. I wanted the information stored on each Activity, so when the user would like to view their profile settings, then the data will not be lost. However, if the username or password does not exist, there is an error message displayed. Similarly to the Sign Up page, I have added a text view which redirects the users to the sign-up page if they did not create an account.

```
public void checkUser() { // validating user method
    String userUsername = loginUsername.getText().toString().trim();
    String userPassword = loginPassword.getText().toString().trim();

    DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "users");
    Query checkUserDatabase = reference.orderByChild( path: "username").equalTo(userUsername);

    checkUserDatabase.addListenerForSingleValueEvent(new ValueEventListener() {
        2 usages
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            if (snapshot.exists()){
                loginUsername.setError(null);
                String passwordFromDB = snapshot.child(userUsername).child( path: "password").getValue(String.class);

                if (passwordFromDB != null && passwordFromDB.equals(userPassword)) {
                    loginUsername.setError(null);

                    //pass the data using intent
                    String nameFromDB = snapshot.child(userUsername).child( path: "name").getValue(String.class);
                    String emailFromDB = snapshot.child(userUsername).child( path: "email").getValue(String.class);
                    String usernameFromDB = snapshot.child(userUsername).child( path: "username").getValue(String.class);

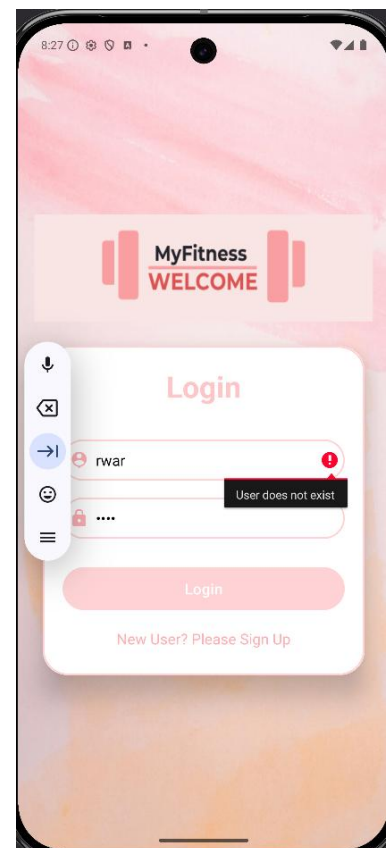
                    //added after second submission
                    SharedPreferences sharedPrefs = getSharedPreferences( name: "user_data", MODE_PRIVATE);
                    sharedPrefs.edit().putString("username", usernameFromDB).apply();

                    Intent intent = new Intent( packageContext: LoginActivity.this, ProfileActivity.class);

                    intent.putExtra( name: "name", nameFromDB);
                    intent.putExtra( name: "email", emailFromDB);
                    intent.putExtra( name: "username", usernameFromDB);

                    startActivity(intent);
                }
            } else {
                loginUsername.setError("User does not exist");
            }

            loginUsername.requestFocus();
        }
    })
}
```



Profile Activity

In my Profile Activity, the user's profile information like name, email, username, weight, height, and fitness goal is displayed on the screen. I implemented SharedPreferences to retrieve and use it to load the user's data from the Firebase Realtime Database, so when the user exits the profile page and later decides to come back to see their information, I am able to store the existing data and display exactly how the user logs in after. I also implemented a method called `showUserDataFromFirebase` that retrieves the user's profile data and display them onto the text view field on the screen to ensure the user can see their latest profile information when the activity loads. In `passUserData` method, the data is sent to Edit Profile Activity using `intent.putExtra` so when the Edit Profile Activity starts, it has all the existing user data already filled and ready to be edited. This allows users to update their profile easily with their existing information already loaded on the screen.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_profile);

    //getting references to the UI controls
    profileName = findViewById(R.id.profileName);
    profileEmail = findViewById(R.id.profileEmail);
    profileUsername = findViewById(R.id.profileUsername);
    homeButton = findViewById(R.id.home_button);
    editButton = findViewById(R.id.edit_button);
    profileWeight = findViewById(R.id.profileWeight);
    profileHeight = findViewById(R.id.profileHeight);
    profileGoal = findViewById(R.id.profileGoal);

    SharedPreferences sharedPrefs = getSharedPreferences("user_data", MODE_PRIVATE);
    username = sharedPrefs.getString("username", null);

    if (username != null) {
        showUserDataFromFirebase(username);
    }
    username = getIntent().getStringExtra("username");

    homeButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(packageContext, ProfileActivity.this, MainActivity.class);
            intent.putExtra("username", username);
            startActivity(intent);
            finish();
        }
    });
}
```

```
Usage:
public void showUserDataFromFirebase(String username) {
    DatabaseReference reference = FirebaseDatabase.getInstance().getReference("users").child(username);
    reference.addListenerForSingleValueEvent(new ValueEventListener() {
        2 usages
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            if (snapshot.exists()) {
                String nameUser = snapshot.child("name").getValue(String.class);
                String emailUser = snapshot.child("email").getValue(String.class);
                String weightUser = snapshot.child("weight").getValue(String.class);
                String heightUser = snapshot.child("height").getValue(String.class);
                String goalUser = snapshot.child("goal").getValue(String.class);

                profileName.setText(nameUser);
                profileEmail.setText(emailUser);
                profileWeight.setText(weightUser);
                profileHeight.setText(heightUser);
                profileGoal.setText(goalUser);
                profileUsername.setText(username);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
        }
    });
}
```

```

1 usage
public void passUserData(){
    String userUsername = profileUsername.getText().toString().trim();

    DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "users");
    Query checkUserData = reference.orderByChild( path: "username").equalTo(userUsername);

    checkUserData.addListenerForSingleValueEvent(new ValueEventListener() {
        2 usages
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            if(snapshot.exists()){
                String nameFromDB = snapshot.child(userUsername).child( path: "name").getValue(String.class);
                String emailFromDB = snapshot.child(userUsername).child( path: "email").getValue(String.class);
                String usernameFromDB = snapshot.child(userUsername).child( path: "username").getValue(String.class);

                String weightFromDB = snapshot.child(userUsername).child( path: "weight").getValue(String.class);
                String heightFromDB = snapshot.child(userUsername).child( path: "height").getValue(String.class);
                String goalFromDB = snapshot.child(userUsername).child( path: "goal").getValue(String.class);

                Intent intent = new Intent( packageContext: ProfileActivity.this, EditProfileActivity.class);

                intent.putExtra( name: "name", nameFromDB);
                intent.putExtra( name: "email", emailFromDB);
                intent.putExtra( name: "username", usernameFromDB);

                intent.putExtra( name: "weight", weightFromDB);
                intent.putExtra( name: "height", heightFromDB);
                intent.putExtra( name: "goal", goalFromDB);

                startActivity(intent);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
        }
    });
}

```

Edit Profile Activity

In this activity, user can update their profile information except their username. When the activity starts, the user data is retrieved from the Intent that was passed from the Profile Activity and it is displayed on the corresponding edit text fields so the user can see and edit their existing information. The save button checks if any input values have changed comparing them to `nameChanged`, `emailChanged`, `setEditText` methods using `if` and `else` conditions. If there are changes, it will update the database with the new values and a toast will be prompted displaying confirmation of the update. If nothing was changed, a toast will be prompted again to notify users that no updates were made. After save, the user will be navigated back to the Profile Activity, passing along the username so the updated profile can be displayed. In the `setEditText` method, I used a Boolean method and `if` conditions so that if the user has changed any of their details. If any of the fields are different or null, it will update the value to the database using the user's username as the key. I set the boolean `inputChange` to true whenever a change is made and if there are no changes then it is set to false.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$SenableEdgeToEdge: this);
    setContentView(R.layout.activity_edit_profile);

    reference = FirebaseDatabase.getInstance().getReference( path: "users");

    //getting references to the UI controls
    editName = findViewById(R.id.editName);
    editEmail = findViewById(R.id.editEmail);

    usernameTV = findViewById(R.id.usernameTV);
    editWeight = findViewById(R.id.editWeight);
    editHeight = findViewById(R.id.editHeight);
    editGoal = findViewById(R.id.editGoal);
    saveBtn = findViewById(R.id.save_button);

    showData();

    setSaveBtn();
}

```

```

1 usage
public void setSaveBtn(){
    saveBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(nameChanged() || emailChanged() || setEditText()){
                Toast.makeText( context: EditProfileActivity.this, text: " Your profile is saved!",Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText( context: EditProfileActivity.this, text: " You have made no changes to your profile!",Toast.LENGTH_SHORT).show();
            }

            Intent intent = new Intent( packageContext: EditProfileActivity.this, ProfileActivity.class);
            intent.putExtra( name: "username", username); // data is passed to profileact class
            startActivity(intent);
            finish();
        }
    });
}

```

```

public boolean setEditText(){
    String newWeight = editWeight.getText().toString().trim();
    String newHeight = editHeight.getText().toString().trim();
    String newGoal = editGoal.getText().toString().trim();

    boolean inputChanged = false;

    if (weight == null || !weight.equals(newWeight)) {
        reference.child(username).child( pathString: "weight").setValue(newWeight);
        inputChanged = true;
    }

    if (height == null || !height.equals(newHeight)) {
        reference.child(username).child( pathString: "height").setValue(newHeight);
        inputChanged = true;
    }

    if (goal == null || !goal.equals(newGoal)) {
        reference.child(username).child( pathString: "goal").setValue(newGoal);
        inputChanged = true;
    }

    return inputChanged;
}

```

```

public void showData(){
    Intent intent = getIntent();

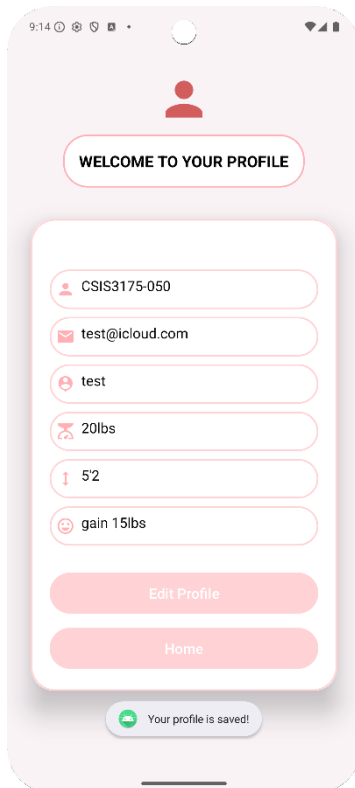
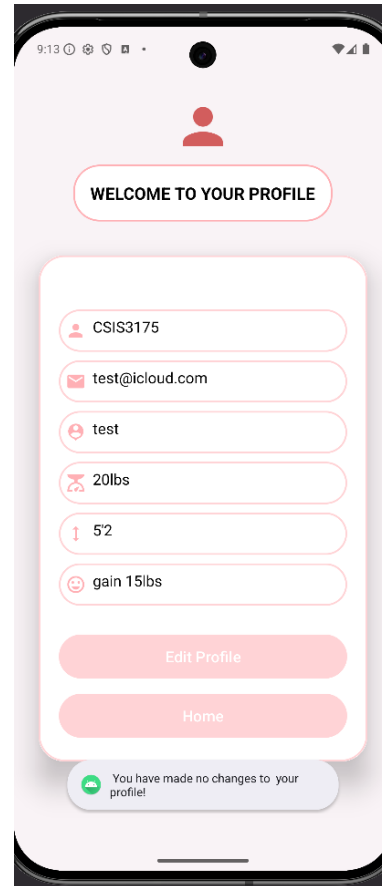
    name = intent.getStringExtra( name: "name");
    email = intent.getStringExtra( name: "email");
    username = intent.getStringExtra( name: "username");

    weight = intent.getStringExtra( name: "weight");
    height = intent.getStringExtra( name: "height");
    goal = intent.getStringExtra( name: "goal");

    if (name != null)
        editName.setText(name);
    if (email != null)
        editEmail.setText(email);
    if (username != null )
        usernameTV.setText(username);

    if (weight != null)
        editWeight.setText(weight);
    if (height != null)
        editHeight.setText(height);
    if (goal != null)
        editGoal.setText(goal);
}

```



```

https://myfitnessapp-8abff-default-rtdb.firebaseio.com/
└─ users
  └─ MyFitness
    └─ hi
      └─ test
        email: "test@icloud.com"
        goal: "gain 15lbs"
        height: "5'2"
        name: "CSIS3175-050"
        password: "test"
        username: "test"
        weight: "20lbs"

```

Home Menu

In Main Activity, I added the Shared Preference again to retrieve and use it to load the user's data from the Firebase Realtime Database so when the user comes back to the Home Menu and goes to the Profile Activity, it will display their current information because without this, I was not able to display the information from the Firebase Database whenever I would go to another activity like Calendar and go back to the Profile Activity, all the input fields were empty. I found implementing Shared Preference was easier than implementing individually to store and display the user's profile information.

```
public class MainActivity extends AppCompatActivity {  
    2 usages  
    TextView calenderTextView, mealplanTextView, workoutsTextView, editprofileTextView;  
    5 usages  
    String username;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);  
        setContentView(R.layout.activity_main);  
  
        calenderTextView = findViewById(R.id.calender_textview);  
        mealplanTextView = findViewById(R.id.mealplan_textview);  
        workoutsTextView = findViewById(R.id.workouts_textview);  
        editprofileTextView = findViewById(R.id.editprofile_textview);  
        |  
        SharedPreferences sharedPrefs = getSharedPreferences( name: "user_data", MODE_PRIVATE);  
        username = sharedPrefs.getString( key: "username", defValue: null);  
  
        calenderTextView.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                Intent intent = new Intent( packageContext: MainActivity.this, CalendarActivity.class);  
                intent.putExtra( name: "username", username);  
                startActivity(intent);  
            }  
        });  
    }  
};
```


Calendar

Event.java

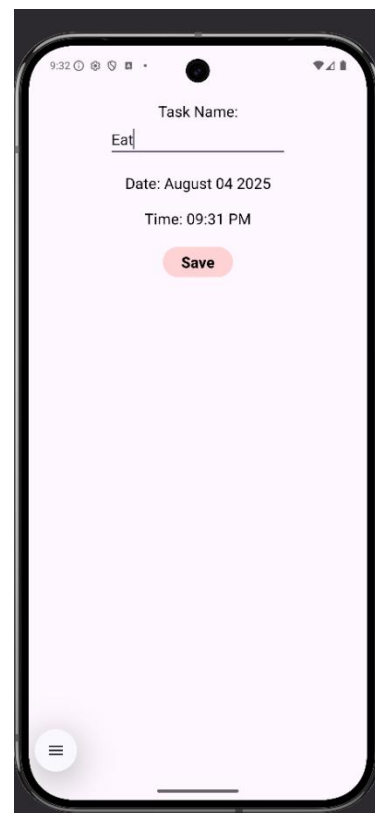
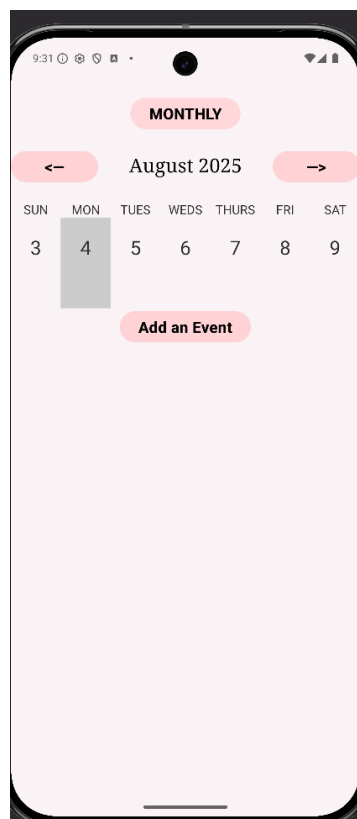
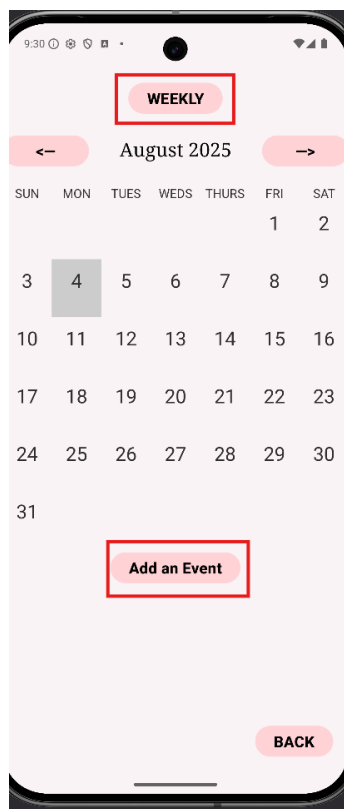
EventAdapter.java

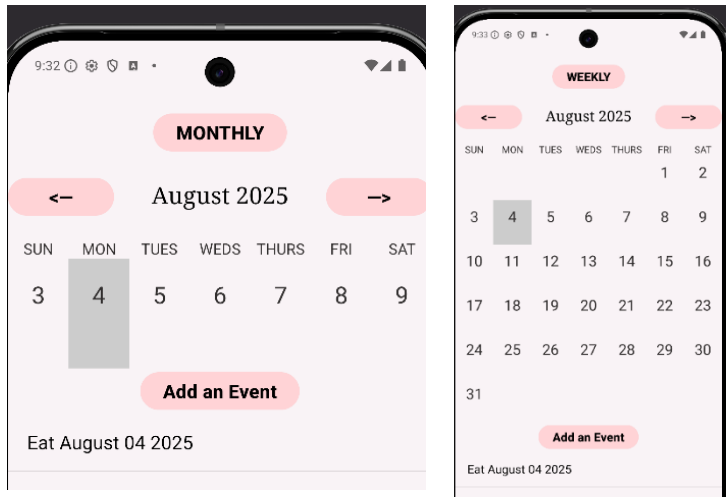
EventEditActivity.java

WeekViewActivity.java

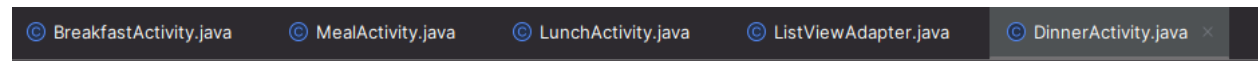
I have implemented a Week View Activity that displays a weekly calendar view that lets users browse through the days of the current week and view their schedule events for each day. To show the seven days of the selected week, I used RecyclerView in a grid layout and populated it with a custom Calendar Adapter. The top of the screen displays the current month and year using the Helper method. When the user clicks on a specific day, the app updates the selected date and reloads the week view accordingly. I have added a back button that allows the users to go back to the Home Menu.

Using a custom Event Adapter, it displays a list of events for the current selected date. If the user clicks the next or previous week buttons, it will shift the date forward or backward by a week and refresh the views. To track a user's progress, I add a new event button that opens the Event Edit Activity in both Weekly View and Monthly View so users can add their task and the date along with the current time are displayed. There is also a monthly view button that takes the users to the full calendar view in Calendar Activity. I have added an onResume method that refreshed the event list to make sure any new events show up as soon as the user returns to the weekly view.

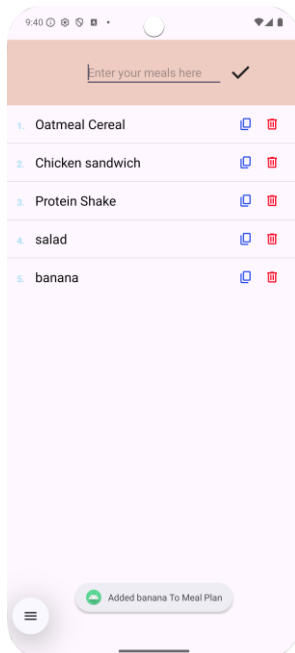




Meal Planner

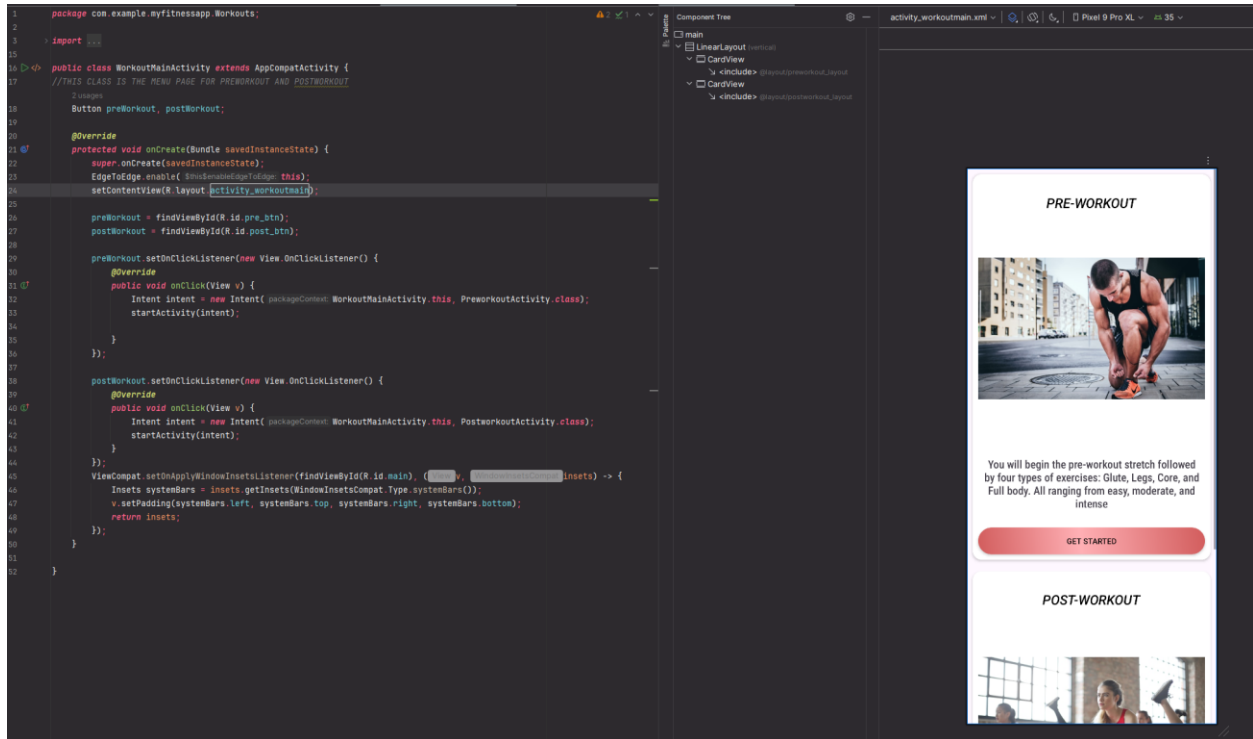


In all three of my Activity, users can view, add, and remove their meals. I have already prefilled in a list of examples for the users to see how it looks like prior to adding. A List View Adapter is implemented to display the meals in a list. When a user enters a new meal in the input field and clicks on the checkmark icon, it will first check if the input is empty and if its not then it will add the meal to the list and clears the input box. There a toast confirming it was added.

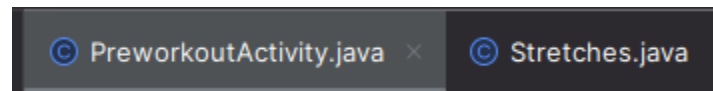


Workouts

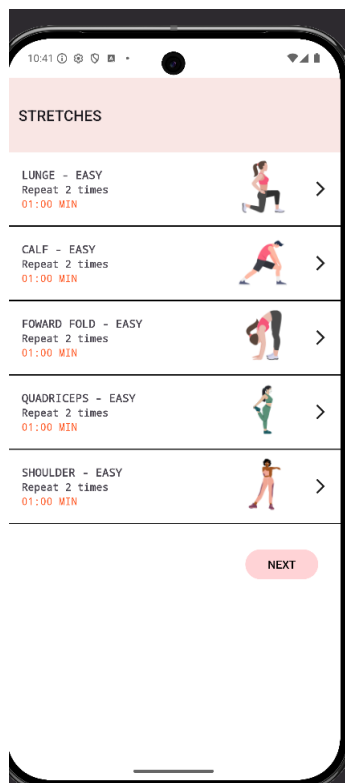
In the WorkoutMain Activity, pre-workout and post-workout options are displayed for the users to choose from. If the users chooses the pre-workout option, it will redirect the user to the pre-workout stretches. I have implemented a get started button that redirects the user to the specific page. All exercise and workouts will display ranging from easy, moderate to intense with the same number of repetitions and timing. For all exercises and stretches, I have created individual string resources for the instructions on how each exercise and stretch works.



Pre-workout



In my Prewriteout Activity, I used the same coding structure as in the Stretches Activity. Users can view and interact with five stretching exercises before starting a workout. I have implemented an array of view IDs to keep track of each stretch pose, and when a user taps on any of them, the app detects which icon was clicked and opens the Stretches Activity, passing a value to indicate which stretch to display. A next button is added so when it is clicked, it takes the user to the ExerciseMain Activity to start their main workout. Each stretch displays the title, number of sets, and the time along side the icon image of the stretches. In landscape view, the users are able to scroll through the stretches



Stretches

In the Stretches activity, the value gets retrieved from the previous screen to figure out which stretch to display for example lunges or forward fold and loads that specific layout onto the screen for the users to interact with. Each stretch has a timer and a start/pause button. When the user taps on the start button, the countdown begins and it displays on the button on top. Users are able to pause the timer if needed. When the timer finishes, it will redirect the user to the next stretch. Once the users is on the last stretch, it will loop back to the first stretch. Users are also able to scroll through stretches in landscape mode. This activity helps guide sures through each stretch, one at a time with a built-in timer. Each stretch displays the image, timer, start/pause button, and alongside instructions on how to do the stretch. In landscape view, the users are able to scroll through the stretches

```
public class Stretches extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_stretches);

        FrameLayout frameLayout = findViewById(R.id.stretch_container);
        LayoutInflater inflater = getLayoutInflater();

        Intent intent = getIntent();
        buttonvalue = intent.getStringExtra("value");

        // int intValue = Integer.parseInt(buttonvalue); //argument 'buttonvalue' might be null
        int intValue = 1;
        try {
            if (buttonvalue != null) {
                intValue = Integer.parseInt(buttonvalue);
            }
        } catch (NumberFormatException e) {
            Log.e(TAG, "Error in CoreMain class", new IllegalArgumentException("Invalid buttonvalue: " + buttonvalue, e));
        }

        View layout = null;
        try {
            if (intValue == 1) {
                layout = inflater.inflate(R.layout.activity_lunge, frameLayout, attachToRoot: false);
            } else if (intValue == 2) {
                layout = inflater.inflate(R.layout.activity_calif, frameLayout, attachToRoot: false);
            } else if (intValue == 3) {
                layout = inflater.inflate(R.layout.activity_forwardfold, frameLayout, attachToRoot: false);
            } else if (intValue == 4) {
                layout = inflater.inflate(R.layout.activity_quadriiceps, frameLayout, attachToRoot: false);
            } else if (intValue == 5) {
                layout = inflater.inflate(R.layout.activity_shoulder, frameLayout, attachToRoot: false);
            }
        } catch (Exception e) {
            Log.e(TAG, "Error loading poses", e);
            finish();
        }

        if (layout != null) {
            frameLayout.addView(layout);
            // references to views inside the inflated layout
            startBtn = findViewById(R.id.startBtn); // if it exists in the selected layout
            minTextView = findViewById(R.id.timerBtn);
            startBtn.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    if (MinTimeRunning) {
                        stopTimer();
                    } else {
                        startTimer();
                    }
                }
            });
        }
    }
}
```

```
1 usage
private void stopTimer() {
    countdownTimer.cancel();
    MinTimeRunning = false;
    startBtn.setText("START");
}

1 usage
private void startTimer() {
    final CharSequence value1 = minTextView.getText();
    String num1 = value1.toString();
    String num2 = num1.substring(0, 2);
    String num3 = num1.substring(3, 5);

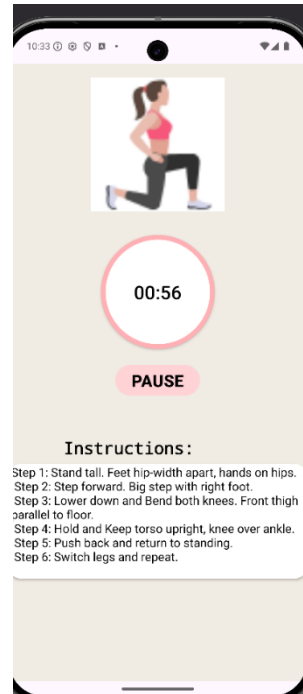
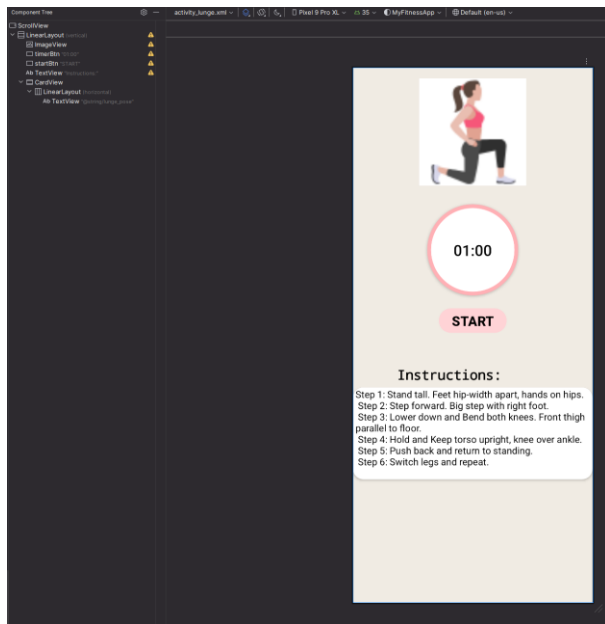
    final int number = Integer.parseInt(num2) * 60 + Integer.parseInt(num3);
    MTimeLeftinmills = number * 1000L;

    countdownTimer = new CountdownTimer(MTimeLeftinmills, countdownInterval: 1000) {
        5 usages
        @Override
        public void onTick(long millisUntilFinished) {
            MTimeLeftinmills = millisUntilFinished;
            updateTimer();
        }
        @Override
        public void onFinish() {
            int newValue = Integer.parseInt(buttonvalue) + 1;
            if (newValue <= 7) {
                Intent intent = new Intent(packageContext: Stretches.this, Stretches.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
                intent.putExtra(name: "value", String.valueOf(newValue));
                startActivity(intent);
            } else {
                newValue = 1;
                Intent intent = new Intent(packageContext: Stretches.this, Stretches.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
                intent.putExtra(name: "value", String.valueOf(newValue));
                startActivity(intent);
            }
        }
    }.start();
    startBtn.setText("PAUSE");
    MinTimeRunning = true;
}
```

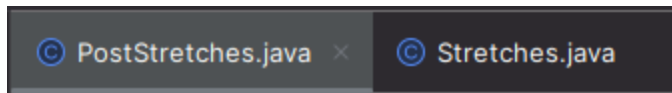
```
1 usage
private void updateTimer() {
    int minutes = (int) MTimeLeftinmills / 60000;
    int seconds = (int) MTimeLeftinmills % 60000 / 1000;

    String timeLeftText = "";
    if (minutes < 10) {
        timeLeftText = "0";
    }
    timeLeftText = timeLeftText + minutes + ":";
    if (seconds < 10) {
        timeLeftText += "0";
    }
    timeLeftText += seconds;
    minTextView.setText(timeLeftText);
}

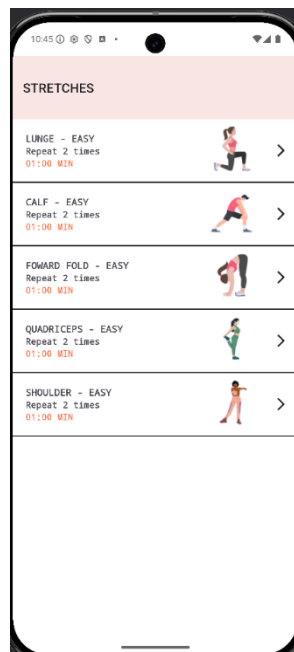
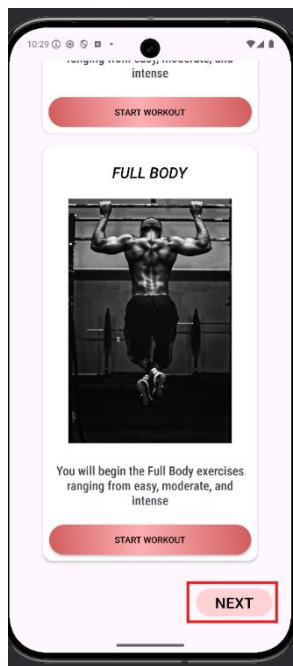
//end of stretch class
```



Post-workout



When the user chooses the post workout option, it will redirect them to the exercise workout for certain muscle groups. At the very bottom of the page, I have implemented a next button that will redirect the users to the Stretching page. This Activity is very similar to what I have implemented in the pre-workout activity except the users will be greeted with the four workout options then stretches to end off. Each stretch displays the image, timer, start/pause button, and alongside instructions on how to do the stretch. In landscape view, the users are able to scroll through the stretches



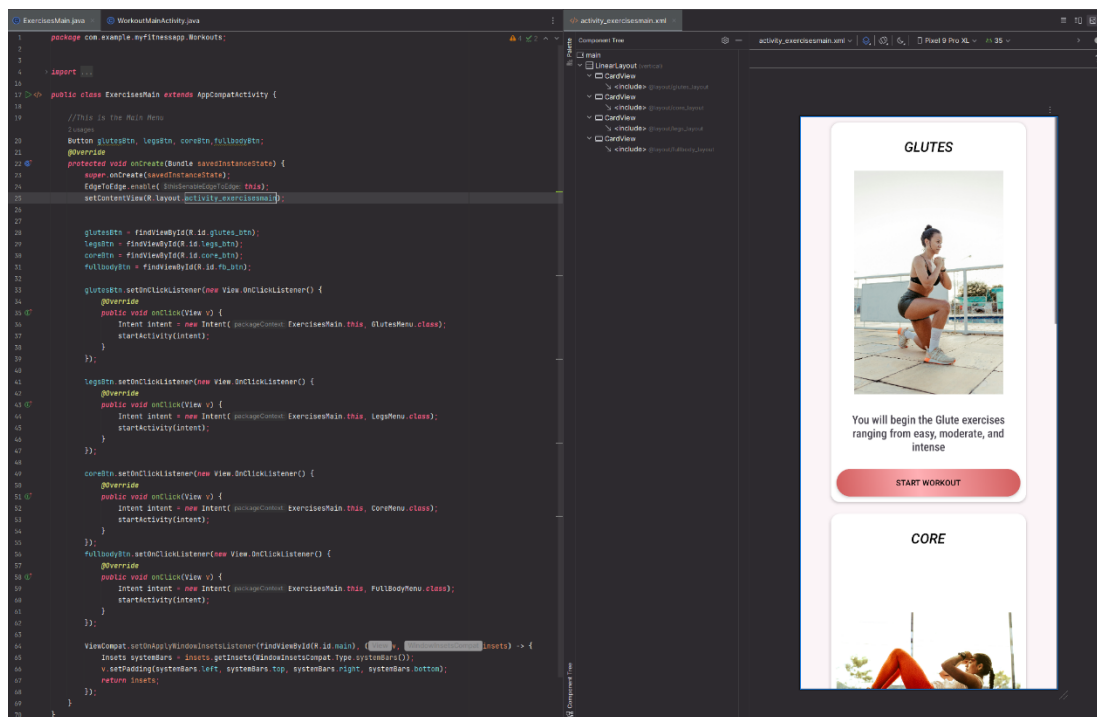
```

public void iconBtnClicked(View view) {
    for(int i=0; i<newArray.length;i++){
        if(view.getId() == newArray[i]){
            int value = i+1;
            Log.i( tag: "First", String.valueOf(value));
            Intent intent = new Intent( packageContext: PostStretches.this, Stretches.class);
            intent.putExtra( name: "value",String.valueOf(value));
            startActivity(intent);
        }
    }
}

```

Exercises

Users can scroll down to choose one of the four exercise workouts for a specific muscle group. I have added a start workout button that will redirect users to the specific muscle group activity page.



Glutes, Core, Legs, Full Body

I have implemented the same layout as the stretching page where the user is able to click on any of the exercise pose, it will redirect the user to that specific page, for example Glutes workout. However, each pose is different from all the four workouts and is it only specific for that type of muscle group. The user will have the option if they would like to focus on certain muscle groups and once, they are done, I have implemented a next button at the very bottom of the screen after scrolling down that will redirect the users to the post-workout stretches if the user chooses the post-workout option.

