Mobile Application for Supporting and Motivating Yoga Practice

Adriana Buchmei*
Supervised by: prof. Ing. Adam Herout Ph.D.†

Faculty of Information Technology
Brno University of Technology
Brno / Czech Republic

Abstract

Sport and physical activity, especially yoga, are becoming more popular, and the demand for effective apps to support exercise comes with increasing technological advances. This paper focuses on designing and testing a user interface for a mobile app for yoga practitioners that allows users to track their progress, motivate themselves, access past training sessions, and take pictures during training without having to think about the phone. The design was iteratively tested with potential users and continuously improved based on their feedback. A prototype of the application was implemented for the Android platform in Kotlin using the Jetpack Compose framework.

Keywords: Mobile App, User Interface, Usability, User Experience

1 Introduction

This work aims to create an application that supports and motivates yoga practice by allowing users to easily track their progress, capture photos during workouts, and monitor improvements over time. While many existing solutions provide workout plans and related features, this app focuses on offering users a seamless way to document and review their practice without interfering with their workout plans. The development process follows a User-Centered Design (UCD) methodology to ensure the app meets user needs and expectations. In Section 2, we analyze existing applications and define a task statement based on user research. In Section 3, we explain the app's design and visual identity. Section 4 delves into the details of individual app components, such as UI design and image processing. Finally, Section 5 covers the gallery's application components like data handling, streak score computation, and gesture handling. The application is currently being tested on Google Play.

*xbuchm03@stud.fit.vut.cz
†herout@fit.vut.cz

2 Analysis and Task Statement

The growing interest in yoga has led to numerous mobile apps for practicing yoga, such as Daily Yoga [2], Yoga for Beginners [5], and Zenia [11]. These apps provide workouts, lesson plans, and audio and video instructions. However, none of them provide personalized progress tracking, where the user can see their photos in a pose over time, easier real-time photography during a yoga workout without having to think about the phone, or integration of multiple motivational elements in the application. This gap opens the space for an app that fills these needs, providing a more tailored and engaging yoga experience.

2.1 Analysis of Existing Applications

A review of existing apps highlights their strengths and limitations:

Key Features: Existing yoga applications provide structured training plans, instructional content, and statistical progress tracking, such as the number of sessions, time spent, and calories burned. They also offer a calendar for tracking workout history and setting reminders.

Identified Gaps: Despite these features, current solutions lack visual progress tracking. They do not allow users to document their training process with photos or track qualitative improvements in pose techniques. Motivation is primarily driven by numerical goals rather than visual progress, which can be a more effective motivator for many users.

Thus, there is a need for an application that enables users to capture photos during training, store them, and track their progress over time by visually comparing their pose improvements.

2.2 Requirements Analysis

Understanding user needs is critical in developing any user interface, especially when targeting a diverse audience. A combination of methods was employed to uncover the expectations and desires of potential users, including user interviews[1], questionnaires, and observational studies of

[1] User Interviews

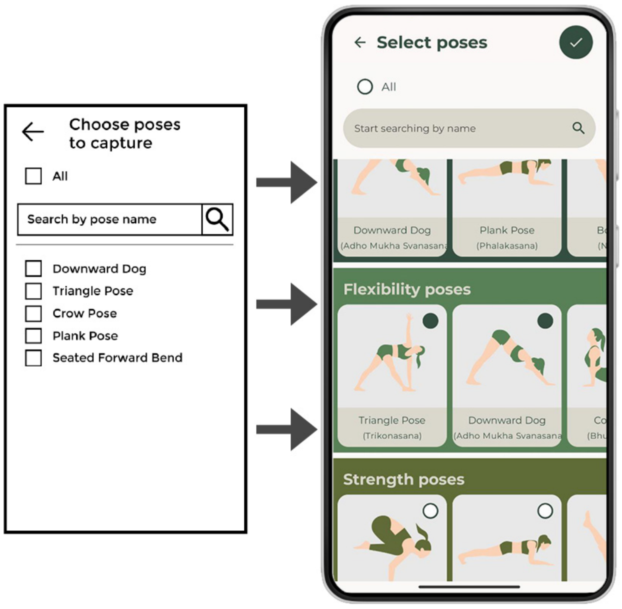

Figure 1: Wireframe and final design of the camera settings screen. The screen allows users to select individual poses or choose all poses they want to be captured.

users interacting with similar applications. These insights helped define the core features necessary for the app to fulfill user expectations effectively: hands-free real-time pose capture with user-defined preferences (**Pose recognition [10]**), saved workout history and calendar integration (**Photo and data logs**), visual comparison of poses over time (**Progress tracking**), and prompts, reminders, and streak tracking to encourage consistency (**Motivational tools**).

2.3 Development and Testing Approach

The development of the application followed the User-Centered Design (UCD) methodology [8], focusing on the needs and goals of the users throughout the design and development process, which included: **User Research:** identifying user needs through interviews and surveys; **Prototyping and Development:** designing wireframes and interactive models in Figma[2], followed by developing Android prototype; **User Testing:** conducting usability tests with diverse participants; **Iteration:** refining features based on continuous feedback.

Testing was performed by a group of 9 participants who represented a variety of experience levels with yoga and mobile applications. Details on the testing of specific parts are described in the following sections.

[2]https://www.figma.com

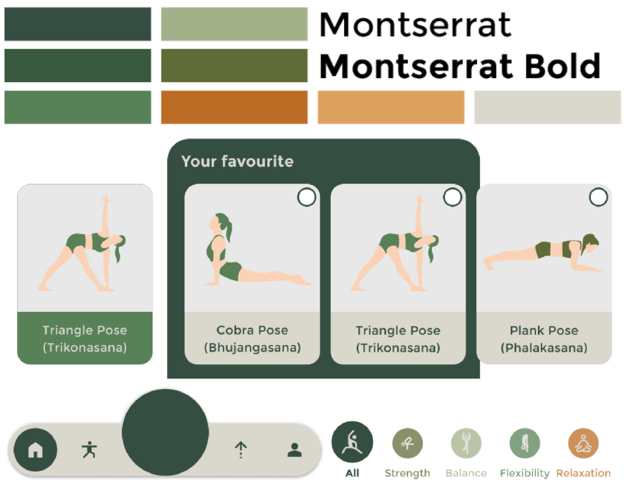

Figure 2: Some design elements, including the color palette, fonts, and UI components, create a cohesive user experience.

3 App Design and Visual Identity

The app's design follows modern UI principles, emphasizing clarity, aesthetics, and functionality to create an intuitive and engaging experience inspired by Steve Krug's *Don't Make Me Think!* [6].

Incorporating Material 3 design principles[3], the app maintains a cohesive, responsive, and scalable layout for optimal usability across different devices. Wireframes outlined the app's structure and flow before transitioning to high-fidelity prototypes in Figma, ensuring an intuitive and efficient user experience. This iterative approach allowed for continuous improvements based on feedback, resulting in a refined and user-friendly UI (Figure 1).

The color palette balances energy and calmness: **Dark Green** symbolizes stability and growth, **Orange** adds energy and motivation, and **Light Beige** enhances readability and contrast. Montserrat [9], a modern sans-serif font, ensures readability and scalability across various screen sizes (Figure 2).

A cohesive visual identity was achieved through uniform illustrations of yoga poses, reinforcing the app's aesthetic. Figma, Adobe Illustrator, and Photoshop[4] were utilized for wireframing, vector illustrations, and visual refinements.

4 User Interface Details and Image Processing

The app uses the modern Jetpack Compose framework [3]. This declarative UI toolkit provides a flexible and intuitive way to create the UI, thus simplifying the development and

[3]https://m3.material.io
[4]https://www.adobe.com/uk/creativecloud.html

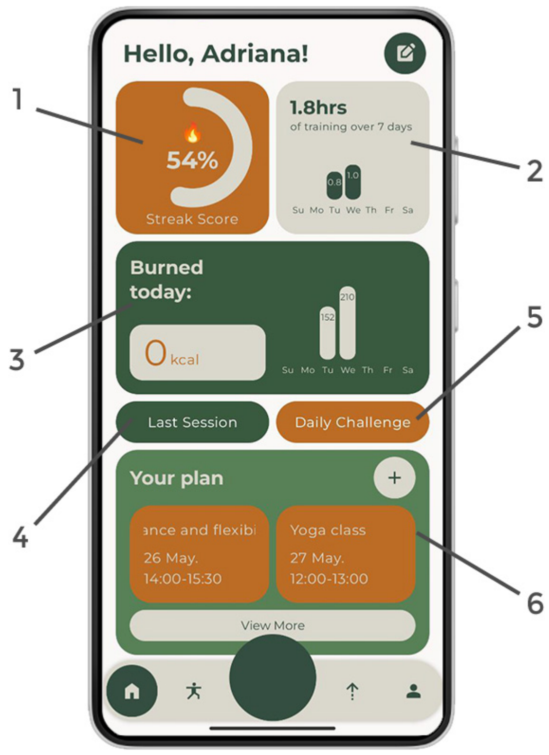

Figure 3: The homepage of the application displays 1) streak score, 2) total training time, 3) calories burned, 4) last session details, 5) daily challenges, 6) planning of future activities.

maintenance of the application. With Jetpack Compose, it is possible to create dynamic and responsive screens with less code compared to traditional XML solutions. The different parts of the user interface are described in the following sections.

4.1 Dashboard

Upon opening the application, the user first sees the home page, providing quick access to key features through a dashboard and navigation bar (see Figure 3). The dashboard includes several widgets, explained below:

1. **Streak Score:** After consulting with users, it was found that counting days often leads to demotivation if a user misses a day. Based on these findings, the task was to find a more effective approach. As a result, a system was introduced where the score of a series is calculated as a percentage based on the completion of a set target. This approach prevents frustration and promotes better motivation to achieve goals; see the Section 5.2 for a more detailed description.

2. **Total training time:** The section showing the total time spent training over the last seven days provides an overview of the duration and consistency of the exercise.

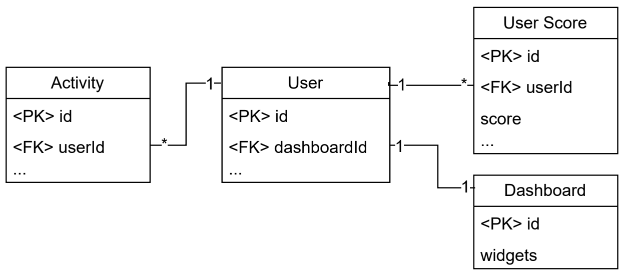

Figure 4: Entity-Relationship diagram of the data model. Each user has a personal dashboard containing editable widgets. The User Score table stores the streak score, while scheduled activities are recorded in the Activity table.

3. **Calories burned:** The panel informs the user about the approximate energy expenditure during the day, while the displayed graph shows the number of calories burned over the last seven days.

4. **Last Session details:** The user can access the previous session quickly.

5. **Daily Challenges:** Here are the ideas for user challenges that provide additional motivation to meet training goals (although this section has not yet been implemented).

6. **Planning of future activities:** The application allows user to add planned sessions while their visual display provides an overview of upcoming activities. Alerts for upcoming sessions can be set individually, allowing users to customize when they want to be notified before the exercise begins.

After talking to participants, it was confirmed that each user preferred a different dashboard layout. Based on these findings, we added an edit mode to the app that allows users to add, remove, or rearrange widgets according to their needs and preferences.

Figure 4 illustrates the data model structure. It represents how each user interacts with their dashboard, which includes customizable widgets and stored activity data.

4.2 Session History

Another app functionality is session history, allowing users to view previous workouts by date and edit their details. This screen displays the workout history, where the user can select a specific date via the horizontal calendar or the detailed calendar that opens by clicking on the icon at the top of the screen. Once a date is selected, the details of that day's workout are displayed, including the poses practiced and a timelapse bar that visually displays the different categories of poses. This screen also includes a notes section where the user can add comments and an emoji button to change the emoji associated with that workout.

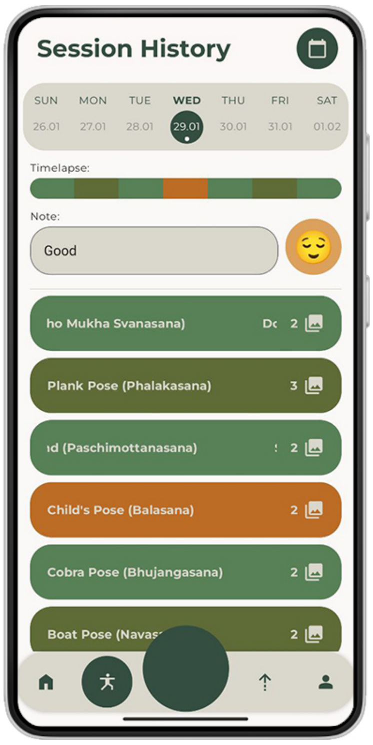

Figure 5: User session history view, allowing users to browse past workouts by date, view details, add notes, and assign an emoji to each session.

Each pose is linked to photos of the user during the workout. Viewing these images is explained in detail in Section 5.3. This functionality allows users to go back to their past workouts and view them in detail. Figure 5 shows a screenshot of the user interface.

Users appreciated the addition of a timelapse bar, which displays the progress of the exercise using colors, with each color representing a different yoga category, such as strength, flexibility, relaxation, and balance.

The data model for the Session entity represents a user's training unit in the database (Figure 6). Each Session is uniquely identified by a primary key (id) and assigned to a user via a foreign key (`userId`). A session can include multiple exercise poses, each referencing a specific pose. Captured images of these poses are stored and linked accordingly.

4.3 Capturing

The application is designed to provide automatic photo capture during yoga sessions, allowing users to track their progress visually. The main screen of this functionality, `CaptureScreen` (Figure 7), displays the recording interface using CameraX[5]. Users can easily switch between the front and rear cameras and start or stop recording according to their preferences. The camera is automatically initialized and shows a preview using `PreviewView`. The

[5]https://developer.android.com/media/camera/camerax

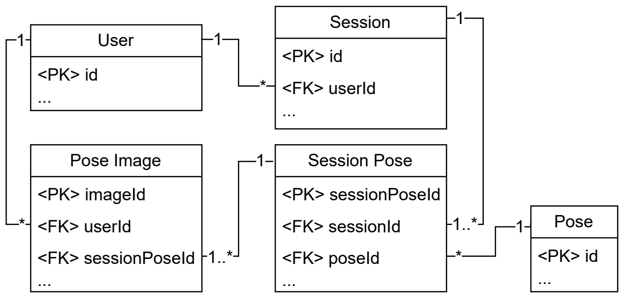

Figure 6: ER diagram for user sessions showing key entity relationships. A Session links to a User via `userId` and includes multiple Session Poses. Each Session Pose references a Pose and connects to one or more Pose Images captured during training.

user has to allow access to the camera. If the user does not grant permission, a notification will appear.

The capture settings select the yoga poses the user wants to capture while practicing yoga, as shown in Figure 1. The screen contains categories of poses such as strength, balance, flexibility, and relaxation. The application recommends poses that the user has already practiced in past sessions in the section of favorite poses, which speeds up the process of finding poses. The user can mark individual poses as selected using checkboxes or select all at once. The search field allows users to filter poses by name. The top bar contains a back button and a button to confirm the selection.

4.4 Progress Tracking

The application allows users to view their photos sorted by individual poses they performed in past sessions. The first screen displays a list of yoga poses the user has practiced (Figure 9). Upon selecting a pose, the gallery of captured photos for that pose opens (Figure 10). Tapping on a photo opens a detailed image view. The gesture handling in these screens is more explained in Section 5.3.

A key gallery feature is progress tracking, which selects multiple photos from an extended period, enabling the user to compare their improvement visually. A "See Progress" button appears if enough progress data is available, allowing the user to access a detailed comparison.

When tracking progress in a yoga pose, consistency in photo alignment is crucial to ensure clear visibility of changes in posture. To achieve this, we captured images at different stages of Downward Dog practice and systematically processed them to align key body points. The alignment was performed manually using Adobe Photoshop, with a fixed pelvic point as the primary reference. This ensured that the pelvis remained at the same level across all images, preventing distortions in posture analysis.

We aligned the feet and hands along a single baseline to maintain body proportions accurately. This approach allowed us to track subtle changes in spine angle, weight

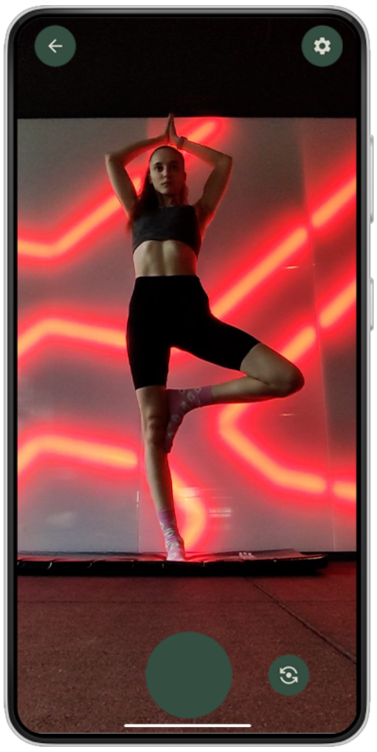

Figure 7: Camera screen with camera preview. There are two buttons on the top bar – one to go back and the other to open the capturing settings. At the bottom is a large button to start/stop capturing and an icon to switch cameras.

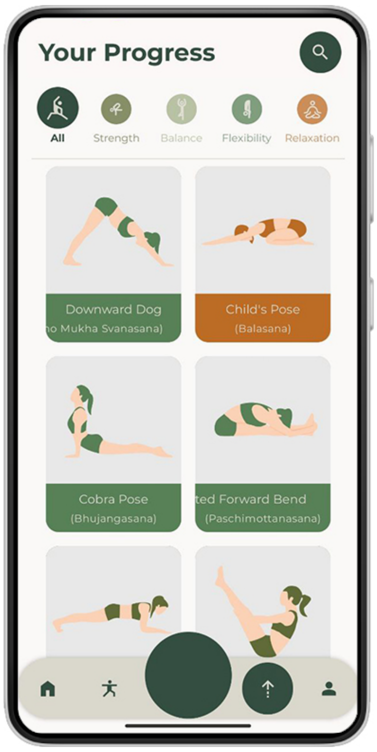

Figure 9: View of practiced poses. Users can browse a list of previously performed yoga poses, search for specific ones, and filter by categories like strength, balance, flexibility, and relaxation. Each pose is displayed as a card that opens a gallery when clicked.

distribution, and overall body alignment over time. By keeping the reference points consistent, we minimized visual inconsistencies that could arise from slight shifts in camera positioning or body posture during different practice sessions. This method proved helpful in analyzing improvements in stability and flexibility without introducing distortions that could misrepresent progress (see Figure 8).

One of the main challenges in manual alignment is the effort required to adjust each image, especially when the practitioner is captured from a different angle. In such cases, perspective correction tools were applied to align the images more accurately.

As part of the future functionality of this application, we plan to implement automatic image alignment using neural networks specialized in pose estimation. Models such as MediaPipe Pose[6], MoveNet[7], or ML Kit Pose Detection[8], can detect key joints—including the pelvis, knees, shoulders, and wrists. By leveraging these technologies, the app will dynamically align and crop photos, eliminating the need for manual adjustments while improving progress tracking accuracy.

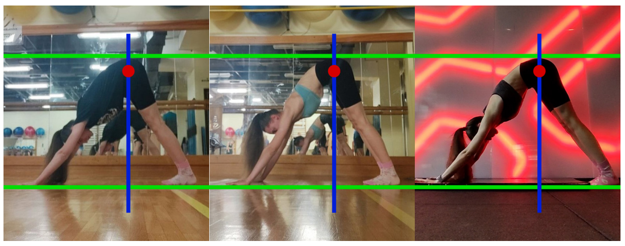

Figure 8: Aligned images of Downward Dog pose, ensuring consistent pelvic positioning and limb alignment for accurate progress tracking.

After conducting user interviews, we gathered insights into how users prefer to visualize their progress. The general preference was to track progress over at least six months, with a minimum of seven images. Users preferred having fewer older photos and more frequent recent ones, as this distribution better highlights their improvements. Regarding labeling, some users found a simple date caption (e.g., October 26) sufficient, while others preferred a relative time format (e.g., taken six months ago). Further user testing is required to determine the optimal approach, as both labeling methods received similar support.

This functionality helps users analyze their progress in detail, monitor technical improvements, and stay motivated to continue their growth.

[6] MediaPipe Pose
[7] MoveNet
[8] ML Kit Pose Detection

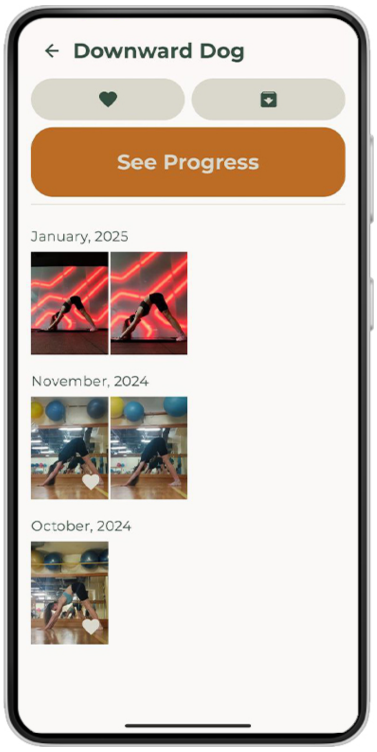

Figure 10: Image gallery screen. Displays photos of a selected pose, sorted by date. Users can toggle favorites, archive images, or progress view and tap for a detailed image view.

5 Key Application Components

This section describes selected parts of the application implementation related to working with data, gestures, and other interesting aspects of the code.

5.1 Data handling and UI Integration

The application is developed in the MVVM (Model-View-ViewModel) architecture [1], which ensures a clear separation of application logic from the user interface, thereby improving the maintainability and testability of the code.

The application utilizes a local Room database [4], ensuring all user data is stored securely on the device. This approach enhances user privacy by preventing data from being transmitted over public networks or stored on external servers. As a result, users have complete control over their information, minimizing the risk of unauthorized access or data breaches.

The application uses asynchronous data processing and reactive UI updates, ensuring smooth and efficient work with data. An essential element is the combination of `ViewModels`[9] and `LiveData`[10] (or `StateFlow`[11]), which store, retrieve, and reactively display data in the UI. When the search query or category filter changes or the user chooses another image from the gallery, the UI adapts immediately without a manual refresh. The navigation between screens was implemented using `NavController`[12], with passing parameters such as the identifiers.

[9] Android ViewModel docs
[10] LiveData overview
[11] StateFlow overview

5.2 Streak Score

The application features a Streak Score system to consistently motivate users to maintain practice. Unlike traditional streak mechanisms that track the number of consecutive days of exercise, this system uses a percentage score calculation. This approach is more flexible and adapts to the user's preferences, better reflecting their training regularity.

The Streak Score is based on the number of workouts per week that the user sets as their target. Based on this target, the score changes dynamically: if the user exercises regularly according to their plan, their streak increases in percentage, or if they exercise less frequently than their set goal, the score gradually decreases.

This system draws inspiration from the Loop Habit Tracker app [7], which incorporates an advanced streak score calculation system based on the frequency of performing an activity. The concept was adapted for this app to track exercise activities more effectively. The formula considers how often a user exercises over the past seven days, ensuring that consistency is rewarded while preventing drastic score drops.

To further refine this approach, the formula includes a boost factor for users who exceed their target training frequency. If the user exercises significantly more than their target, they receive a higher score increase. This mechanism encourages users to maintain consistency while rewarding occasional extra effort. The final streak score is calculated as follows:

$$x = \frac{\sqrt{t}}{13}, \tag{1}$$

$$M = 0.5^x, \tag{2}$$

$$B = \begin{cases} 1.3, & \text{if } f \geq 1.5t \\ 1.15, & \text{if } f \geq t \\ 1.0, & \text{otherwise} \end{cases}, \tag{3}$$

$$S = S_p \times M + c \times (1 - M) \times B \tag{4}$$

where: f is the frequency (how many times the user exercised in the past seven days), t is the target training frequency per week, x is the exponent used in the multiplier calculation, M is a multiplier based on frequency, S_p is the previous streak score, c is the checkmark value (1 if the user exercised, 0 otherwise), B is the boost factor, which increases the score if the user trains more than their target, S is the final computed streak score, ensuring that it never exceeds 100 %.

[12] NavController docs

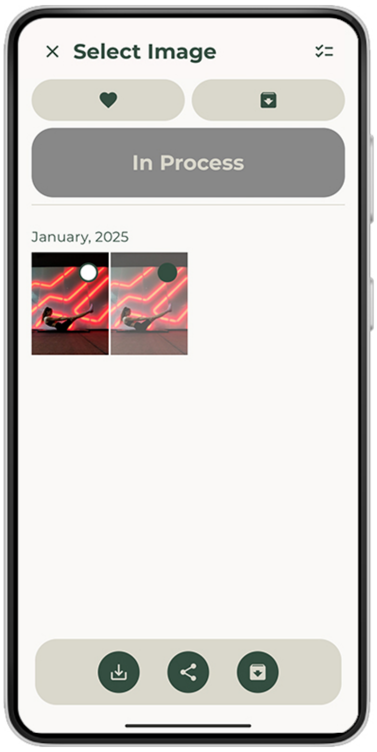

Figure 11: Users can select images by long-pressing and toggle selection by tapping. Zooming adjusts the number of images per row. Users can download, share, or archive the selected images in selection mode.

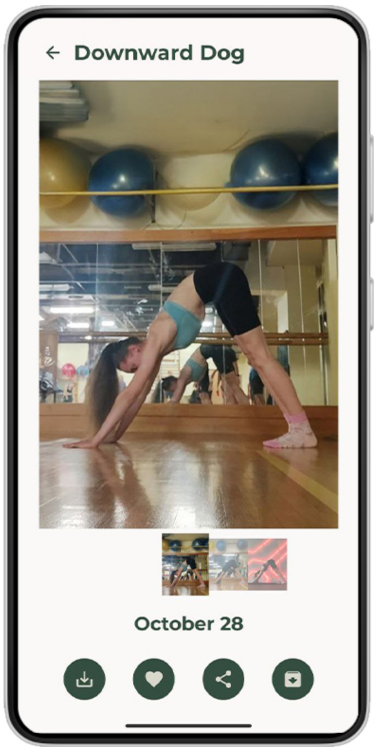

Figure 12: Thumbnails below the image enable navigation and display image data, and a bottom bar provides options to download, share, archive, or mark as a favorite.

The streak score is recalculated once per day. If the user exercises on that day, the score is updated accordingly. Additionally, when the user opens the app, it checks if the streak scores for previous days have been calculated. The app will compute the missing scores if any days are missing, ensuring the user's progress is always up-to-date. This feature ensures an accurate and timely reflection of the user's consistency without delays.

5.3 Gesture Handling in Gallery

Gestures enhance usability by allowing users to interact with the gallery intuitively through tapping, swiping, zooming, and long-pressing. These interactions eliminate the need for extra buttons or controls, providing a seamless browsing and selection experience.

Users can interact with images in the gallery view through selection and zoom functionality. The `detectTapGestures` within the `pointerInput` modifier enables image selection: `onLongPress` activates selection mode and adds the image to the selection. In contrast, `onTap` either opens the image in detail or toggles its selection, depending on the mode. Additionally, the zoom functionality allowing users to adjust the number of images per row using `detectPinchGestures`. Zooming in reduces the number of columns (fewer images per row), while zooming out increases them, improving browsing efficiency. This selection and zooming interaction is illustrated in Figure 11.

In the ImageCarousel component, which focuses on providing a detailed image viewing experience (Figure 12), users can toggle fullscreen mode by zooming in or double tap on image (Figure 13). Users can swipe left or right to navigate through images, and select a specific image from the thumbnails below. The component dynamically updates the displayed image, with the `onImageChange` callback notifying the parent component of any changes.

User testing indicated that the zoom and swipe features in the image carousel allowed users to interact more intuitively with the images. When users swiped between images, the zoom level and position stayed consistent, which was appreciated as it allowed users to review the technique or posture without interruption. However, a limitation was found in the transition animations when switching to and from fullscreen. These animations were not as smooth as expected, so this is an area for future improvement to enhance the fluidity of the user experience.

6 Conclusions and Future Works

The development of this application provided valuable experience in Android development, particularly in MVVM architecture, Jetpack Compose, Kotlin, and mo-

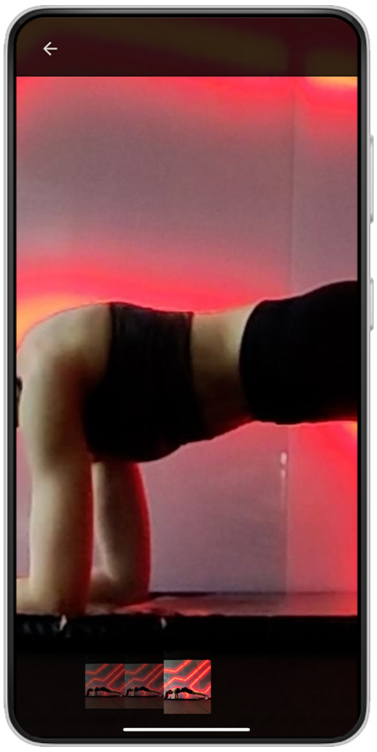

Figure 13: The figure shows fullscreen mode with maximum zoom. The fullscreen mode adds a navigation bar at the top to exit the fullscreen and a row of thumbnails at the bottom to navigate between images.

bile databases. A key realization was the importance of continuous user testing, which refined the user experience and emphasized early feedback.

The app is designed to help users effortlessly track their yoga progress. It allows logging past workouts, adding notes, and visualizing improvements through photos. CameraX integration was implemented, contributing to future automatic photo capture during practice. A streak score system encourages consistency, while gesture-based navigation ensures a seamless user experience. The application is currently being tested on Google Play. A link[13] is available to request access to the testing program; please contact to be added to the testing group.

Future works include **Gamification and Challenges:** introducing challenges, goals, and rewards to boost engagement; **Neural Network Integration:** implementing AI-powered features like real-time pose detection and automatic image alignment and cropping for progress visualization; **Smoother Gallery Animations:** enhancing gallery animations for a smoother browsing experience; **Beginner's Guide:** adding a simple, step-by-step guide for easier app navigation for a new user; and **Improved Dashboard Gestures:** implementing drag-and-drop gestures to allow users to rearrange the widgets more easily. The goal is to help users monitor their progress, stay motivated, and achieve their yoga goals through visual feedback.

[13]Testing program

References

[1] Nayab Akhtar and Sana Ghafoor. Analysis of architectural patterns for android development. *no. June*, 2021.

[2] Daily Fitness. Daily yoga ®: Yoga for fitness, 2025. https://play.google.com/store/apps/details?id=com.dailyyoga.inc.

[3] Google. Jetpack compose ui app development toolkit. https://developer.android.com/compose.

[4] Google Inc. Room persistence library, 2018. https://developer.android.com/topic/libraries/architecture/room.

[5] Leap Fitness Group. Yoga for beginners weight loss, 2025. https://play.google.com/store/apps/details?id=yogaworkout.dailyyoga.go.weightloss.loseweight.

[6] Steve Krug. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability (3rd Edition)*. New Riders, 2013. 3rd edition.

[7] Loop Habit Tracker Developers. Loop habit tracker, 2025. https://github.com/iSoron/uhabits.

[8] Travis Lowdermilk. *User-centered design: a developer's guide to building user-friendly applications*. "O'Reilly Media, Inc.", 2013.

[9] Julieta Urtubey. Montserrat font, 2011. https://github.com/JulietaUla/Montserrat?tab=readme-ov-file.

[10] Santosh Kumar Yadav, Amitojdeep Singh, Abhishek Gupta, and Jagdish Lal Raheja. Real-time yoga recognition using deep learning. *Neural computing and applications*, 31:9349–9361, 2019.

[11] Zenia. Zenia app, 2025. https://www.producthunt.com/products/zenia.