دانشکده مهندسی
گروه مهندسی کامپیوتر

پایان نامه کارشناسی

نرم‌افزار Stamina

مولف
نوید آل قرائی

استاد راهنما
امیرحسین طاهری نیا

تاریخ ارائه
تیرماه ۱۴۰۱

فهرست مطالب

چکیده

در این پروژه، یک نرم‌افزار مبتنی بر تلفن همراه طراحی شد که در آن الگوریتم **corrective exercise** پیاده سازی شد. در این نوشته، پس از طراحی و تحلیل، چالش‌های به وجود آمده در مسیر مطرح شد و راه حلی که برای آن‌ها در نظر گرفته شد ارائه گردید.

کلمات کلیدی: گوشی، حالت، رویداد، بلاک، معماری، مخزن، سرویس

مقدمه

با ظهور ویروس کوید ۱۹، به علت بسته شدن اماکن ورزشی مثل باشگاه ها و کلاس های نرمش، سلامتی و تحرک مردم تحت شعاع قرار گرفته است. در نتیجه، نرم افزارها و سرویس‌هایی از قبیل Apple Fitness+ و Peloton که در زمینه ورزش و سلامت فعالیت دارند با استقبال بی نظیری از سوی کاربران رو به رو شده است. این سرویس ها از ویدئوهای آموزشی استفاده می‌کنند که در آن، مربی های مطرح نرمش و حرکات را انجام می‌دهند و کاربران میتوانند حرکات مربیان را دنبال کنند. با وجود تمام این ویژگی‌ها، این سرویس‌ها هیچ معیاری برای مانیتور کردن کاربر نظر نگرفته‌اند و به تبع آن، نمی‌توان اطمینان حاصل کرد که کاربران این سرویس ها، به خصوص کاربرانی که تخصص کافی در حوزه ورزش ندارند، تا چه حدی حرکات را به درستی انجام میدهند. بنابراین، کاربران نه تنها ممکن است انگیزه خود را برای ادامه ورزش از دست بدهند، بلکه انجام اشتباه حرکات ممکن است آن ها را با آسیب های جدی در دراز مدت رو به رو کند.

فصل ۱. امکان سنجی و تعریف پروژه

۱.۱. نقاط مثبت و منفی سرویس ها و نرم افزارهای موجود در بازار

سرویس های متعددی توسط شرکت های مختلف وارد بازار شده‌اند و از پردازش تصویر برای مانیتور کردن کاربر استفاده میکنند. ما تعدادی از این سرویس ها را بررسی کردیم و اکنون به شرح اپلیکیشن‌هایی میپردازیم که نیازی به تجهیزات جانبی ندارند.

۱.۲. اپلیکیشن Plank، توسعه دهنده Exer

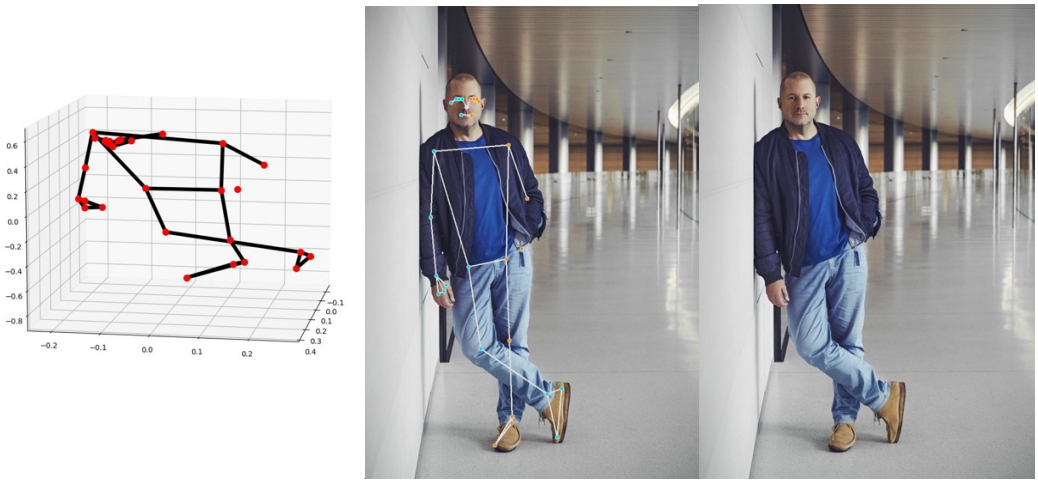
این اپلیکیشن برای نرمش و ورزش در داخل خانه طراحی شده است که قادر است، حرکات کاربر را به صورت زنده تحلیل کند و همزمان، بازخوردی برای کاربر صادر کند. این بازخورد میتواند حاوی یک پیشنهاد برای اصلاح حرکت یا حالت بدن باشد، یا اینکه جمله ای باشد تا کاربر را به ادامه روند فعلی‌اش تشویق کند. ثانیا، این برنامه شامل فیلم های آموزشی است که توسط افراد حرفه ای در هر زمینه های مختلف ورزشی ارائه می‌شود. همچنین، این برنامه دارای یک جدول رتبه بندی است که کاربران را براساس میزان عملکردشان رتبه بندی میکند و آنان را به بهبود رتبه خود ترغیب میکند. با وجود تمام این ویژگی‌ها، موانعی برای کاربران وجود دارد. این برنامه تنها برای پلتفرم هایی که توسط اپل تولید شده است، توسعه داده شده است. به عبارت دیگر، تنها کاربران آیفون و آیپد می‌توانند از این سرویس استفاده کنند. از طرفی، درصد زیادی از کاربران از دستگاه های مبتنی بر سیستم عامل اندروید استفاده می‌کنند و این افراد نمی‌توانند از تکنولوژی های این سرویس استفاده کنند.

۱.۳. اپلیکیشن Onyx Personal Training، توسعه دهنده Onyx Inc

این اپلیکیشن نیز می‌تواند حرکات کاربر را تحلیل کند و بازخوردهایی درباره عملکرد کاربر صادر کند. این نرم‌افزار حاوی مجموعه ای از نشان است که در صورتی که کاربر حرکات خاصی را انجام دهد یا رکورد جدیدی ثبت کند، این نشان ها برای تشویق به کاربر داده می‌شود. همچنین، این برنامه فیلم های آموزشی ارائه می‌دهند که در آن، ورزشکاران حرفه‌ای حرکات ورزشی را انجام میدهند. اما همانند سرویس قبل، این نرم‌افزار تنها برای کاربران iOS قابل استفاده است و کاربرانی که از گوشی ها و تبلت های مبتنی بر اندروید استفاده می‌کنند، امکان نصب و استفاه از این نرم‌افزار را ندارند.

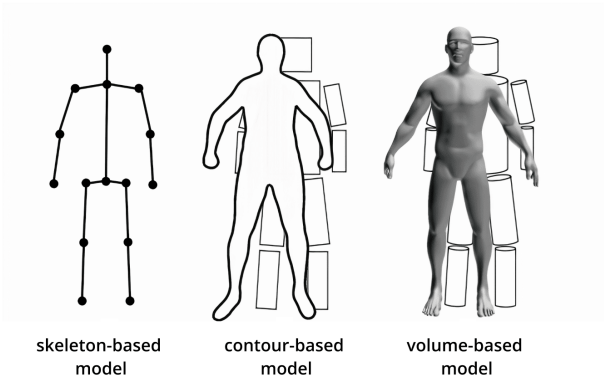
۱.۴. تخمین وضعیت بدن

تخمین وضعیت بدن یک تکنولوژی مبتنی بر بینایی ماشین می‌باشد که وظیفه شناسایی و دسته بندی نقاط خاصی از بدن انسان را بر عهده دارد. این نقاط نمایانگر مفاصل بدن می‌باشند تا زوایا و خمیدگی و انعطاف بدن و در نتیجه طرز قرار گرفتن بدن را محاسبه کند. نمونه‌ای از ورودی و خروجی الگوریتم در تصویر ۱.۱ به نمایش گذاشته شده‌است.



تصویر ۱.۱

به طور کلی سه مدل برای نشان دادن انسان وجود دارد: ۱. مبتنی بر اسکلت ۲. میتنی بر محیط ۳. مبتنی بر حجم. مدل مبتنی بر اسکلت به دلیل انعطاف پذیری بالا بیشترین کاربرد را در میان سه مدل یاد شده دارد. اسکلت حاصل شامل مهم ترین نقاط بدن می‌باشد که عبارتند از: مچ پا، زانو، سرشانه، آرنج، مچ دست و جهت گیری‌های اندام. این سه مدل در تصویر ۱.۲ نشان داده شده‌است.



تصویر ۱.۲

مدل اسکلت از حالت سه بعدی و دوبعدی استفاده می‌کند. تخمین سه بعدی وضعیت بدن دقت و صحت بیشتری نسبت به دو بعدی ارایه می‌دهد. زیرا عمق را نیز مدنظر قرار می‌دهد و برای بسیاری از حرکات عمق مهم است زیرا خاصیت بدن انسان طوری است که صرفا در فضای دو بعدی حرکت نمی‌کند.

به طور کلی معماری مناسب برای یادگیری عمیق مبتنی بر شبکه‌های عصبی کانولوشن می‌باشند. دو روش وجود دارد که عبارتند از: بالا به پایین و پایین به بالا.

در روش پایین به بالا، مدل، کلیه موارد مربوط به نقطه ای خاص را پیدا می‌کند. (مانند همه دست‌های چپ موجود در عکس) سپس تلاش می کند تا گروهی از نقاط کلیدی را در اسکلت هایی از اشیاء متمایز با یکدیگر ادغام کند. روش بالا به پایین که

خلاف روش قبل عمل می‌کند، ابتدا از اشیای داخل عکس را شناسایی کرده و با کشیدن مربع دور آن از سایر اشیا متمایل می‌کند. سپس نقاط کلیدی آن شی برش خورده را محاسبه و پیدا می‌کند.

۱.۵. بازخوردهای سه‌گانه سیستم

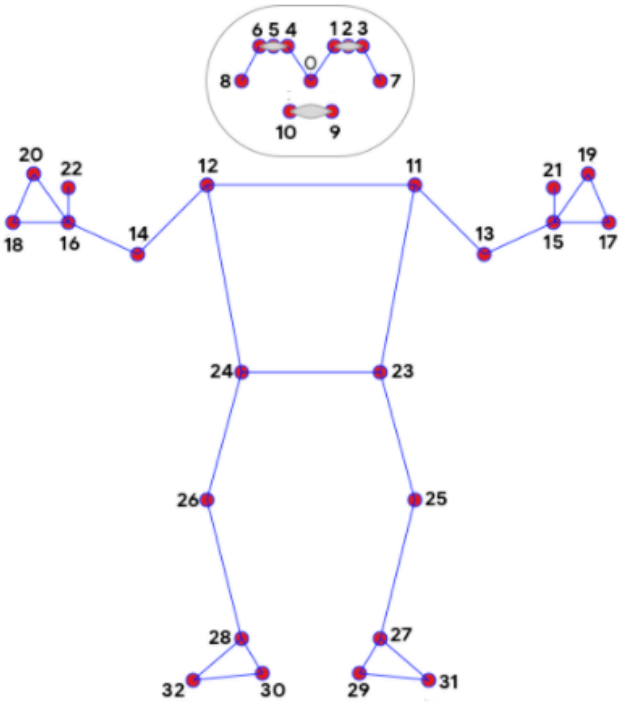
در فرایند نمره دهی در این روش از اسکلت شناسایی شده توسط OpenPose استفاده میشود، به طوری که ۴ نقطه از مهمترین قسمت‌های بدن مورد ارزیابی قرار می‌گیرند. اسکات ایده‌آل در شرایطی رخ میدهد که ران‌ها هم سطح با زمین و زانوها در امتداد انگشتان پا قرار گیرند در صورت اجرای حرکت به صورت ذکر شده ماکزیمم نمره دریافت میشود. برای اندازه‌گیری چنین معیاری ۴ نقطه اساسی بدن مورد بررسی قرار می‌گیرد. کاربر می‌تواند نتیجه عملکرد خود را به صورت اعداد درج شده بر روی قسمت‌های مختلف اسکلت مشاهده کند. روش بعدی مبتنی بر بازخورد سریع میباشند. در حاشیه نویسی ویدیویی، دو نوع بازخورد پیشنهاد می شود. اولاً کاربر می تواند تصاویر آموزشی خود را با مربی روی صفحه مقایسه کند، ثانیاً دستیار صوتی می تواند کاربر را راهنمایی کند و به او اطلاع دهد. زاویه عکاسی نیز می تواند بر نتیجه تاثیر بگذارد. بر اساس یک آزمایش، ثابت شد که برخی از نقاط ممکن است بسیار شبیه به سیستم به نظر برسند، در حالی که از زاویه ای دیگر، تفاوت ها نمایان می شود. لازم به ذکر است که ما قصد نداریم آنالیز را به تمرین‌هایی مانند یوگا یا اسکات محدود کنیم، بلکه هدف ما این است که آن را به هر فعالیتهایی تعمیم دهیم که وضعیت بدن در آن دخیل باشد. زیرا معتقدیم این قابلیت میتواند در کاربردهای نظامی یا آموزش اسب سواری ، رقص و ..... استفاده شود.

اما سرویسی که در این برنامه مورد استفاده قرار گرفته است، بر اساس MediaPipe می‌باشد که در بخش بعدی توضیح داده خواهد شد.

۱.۶. تکنولوژی MediaPipe Pose

Pose فریمورکی مبتنی بر یادگیری ماشین است که برای تشخیص دقیق حالت بدن است که با استفاده از ۳۳ نقطه عطف بر روی بدن در فضای سه بعدی و جداسازی بدن از عکس پس‌زمینه اسکلتی از حالت بدن را استخراج می‌کند. رویکردهای پیشرفته فعلی از قبیل openpose، صرفا به محیطهای قدرتمند دسکتاپ متکی هستند؛ درحالی که این تکنولوژی در تمامی محیط‌ها مثل دستگاه‌های قابل حمل از قبیل گوشی و لپ‌تاپ و یا حتی بر روی محبط وب قابل اجرا است.

ML Kit Pose Detection یک تطابق اسکلتی ۳۳ نقطه‌ای برای تمام بدن ایجاد می‌کند که شامل نقاط برجسته صورت (گوش، چشم، دهان و بینی) و نقاط روی دست‌ها و پاها می‌شود.



تصویر ۱.۳. نقاط عطف که توسط این کتابخانه شناسایی می‌شود

طرز کار این راه حل به این شکل است که ابتدا عکس ورودی به کتابخانه MediaPipe داده می‌شود تا نقاط کلیدی بدن کاربر شناسایی شوند. خروجی لیستی از مختصات ۳۳ نقطه عطف از بدن در فضای سه بعدی می‌باشد. لیست مختصات مکان هر یک از نقاط مهم را تعریف می‌کند. با استفاده از این نقاط، می‌توان یک مدل دقیق از اسکلت بدن کاربر ساخت که از صفر تا ۳۲ شماره‌گذاری می‌شوند. ۱۱ نقطه اول که نقاط صفر تا ۱۰ هستند، برای تشخیص عناصر صورت به کار گرفته می‌شوند.۱۱ نقطه بعدی که نقاط ۱۱ تا ۲۲ هستند برای شناسایی قسمت‌های بالایی بدن هستند که شامل شانه‌ها، آرنج‌ها، مچ‌های دست، دست‌ها و تخمینی از سه انگشت اعم از انگشت اشاره، انگشت شست و انگشت کوچک می‌باشند. ۱۱ نقطه انتهایی که نقاط ۲۳ تا ۳۲ هستند، قسمت پایینی بدن را تعریف می‌کنند که شامل مفاصل ران، زانوها و پاها هستند. با استفاده از این نقاط، نه تنها شمای کلی اسکلت بدن انسان بدست می‌آید، بلکه جهت بدن در فضای سه‌بعدی نیز حاصل می‌گردد. تشخیص حالت بدن از طریق یک مدل از پیش آموزش دیده انجام می‌گردد.

۱.۷. نگاه کلی به سیستم

ما سیستمی را طراحی کردیم که می‌تواند به کمک تلفن‌های هوشمند ایرادات و اشتباهات را در انجام حرکات ورزشی شناسایی می‌کند. با استفاده از این نرم افزار، کاربر میتواند فیلمی از حرکات ورزشی را به عنوان الگو آپلود کند. برای اینکه کاربر بهترین نتیجه را بگیرد، بایستی فیلم حرکات ورزشکار حرفه ای را آپلود کند که حرکاتش، تمام معیارهای یک حرکت صحیح را داشته باشد. سپس، کاربر خود را در حالی که حرکات را انجام می‌دهد، با استفاده از دوربین گوشی ضبط می‌کند. پس از ضبط فیلم، فیلم بر روی سرور بارگذاری می‌شود و سرور نیز فیلم کاربر را متناسب با فیلم الگو تحلیل می‌کند و یک فیلم آنالیز شده از حرکات کاربر را می‌فرستد، به طوری که کاربر میتواند ببیند حرکاتش تا چه حد منطبق بر حرکات ورزشکار حرفه‌ای است. فرایند به این شکل طراحی شده است که کاربر می‌تواند یک جلسه جدید ایجاد کند و پس از آپلود فیلم نمونه، به تعداد دلخواه تصاویر ضبط شده از حرکات خود را بفرستد. به این ترتیب، برنامه میتواند روند پیشرفت کاربر را در غالب نمودار به کاربر نشان بدهد.

همچنین، این سیستم به گونه‌ای طراحی شده‌است که کاربر امکان پشتیبان‌گیری از فرایند خود را دارد. کاربر میتواند یک نسخه پشتیبان از عملکرد خود در فضای ابری آپلود کند و فرایند خود را در دستگاه دیگر از سرگیری کند. با استفاده از این ویژگی، چند کاربر میتوانند از یک دستگاه استفاده کنند بدون آن که عملکرد آنان ریست شود.

۱.۸. ابزارهای توسعه

در این پروژه از زبان دارت و فریمورک فلاتر به عنوان ابزار اصلی توسعه استفاده می‌شود تا کلیه سیستم عامل های گوشی را پشتیبانی کند. همچنین از محیط VSCode ورژن ۱.۶۷ برای توسعه کد و اجرای برنامه استفاده می‌شود و در مواردی که نیاز به خروجی iOS می‌باشد، ابزار Xcode ورژن ۱۳.۴ مورد استفاده قرار می‌گیرد.

برای اجرای برنامه بر دستگاه iOS از نرم افزار Simulator و برای Android از Android Studio استفاده می‌شود.

برای رسم نمودار های usecase, class, Activity از نرم افزار آنلاین lucid chart استفاده شده‌است.

برای اجرای برنامه در دستگاه‌های خارجی از iPhone SE ۲ و Samsung Galaxy A۰۱ استفاده می‌شود تا امکاناتی که از طریق شبیه سازهای مجازی مانند دوربین قابل تست نمی‌باشد را پوشش دهد.

جهت پیاده سازی فرایند ورود و ثبت نام کاربران از سرویس Firebase استفاده شده است. Firebase سرویسی است که توسط گوگل توسعه داده شده است و امکانات متعددی را برای پیاده سازی بسترهای سمت سرور را فراهم می‌کند. همچنین، سرویس Firebase بستری برای فضای ابری سریع اما محدود فراهم می‌کند و برای پشتیبان‌گیری از سرویس Firebase Storage استفاده شده است.

این برنامه از سرویسی تحت عنوان سایوا بهره می‌برد که وظیفه تحلیل حرکات کاربر را به عهده دارد. به طوری که مجموعه‌ای از فریم‌ها بارگزاری شده و کاربر در دو حالت می‌تواند حرکات خود را به عنوان ورودی حرکت تست بارگزاری کند:

- حرکت تمرینی به صورت فیلم با ظرفیت حداکثر ۱۰ مگابایت بارگزاری شود.
- حرکت تمرینی به صورت فریم های گسسته و متناظر با عکس مرجع بارگزاری شود.

همچنین از Github به عنوان سیستم کنترل ورژن استفاده می‌شود و سیستم نام‌گذاری برنچ بدین شرح است که به ازای هر قابلیت که قرار است به برنامه افزوده شود برنچی از روی آخرین نسخه برنچ اصلی ساخته شود و پس از آزمون و تضمین نبود باگ نهایتا با همان برنچ ادغام می‌شود.

در فرایند گرفتن خروجی iOS و Android مشکلات متعددی به وجود آمد که با بررسی فراوان مشخص شد که استفاده از VPN به دلایل زیر ضروری است:

۱. سیستم مدیریت پکیج زبان دارت که Pub نام دارد فیلتر شده است.
۲. دسترسی به کلیه مشتقات Firebase از طریق IP ایران غیر ممکن است.

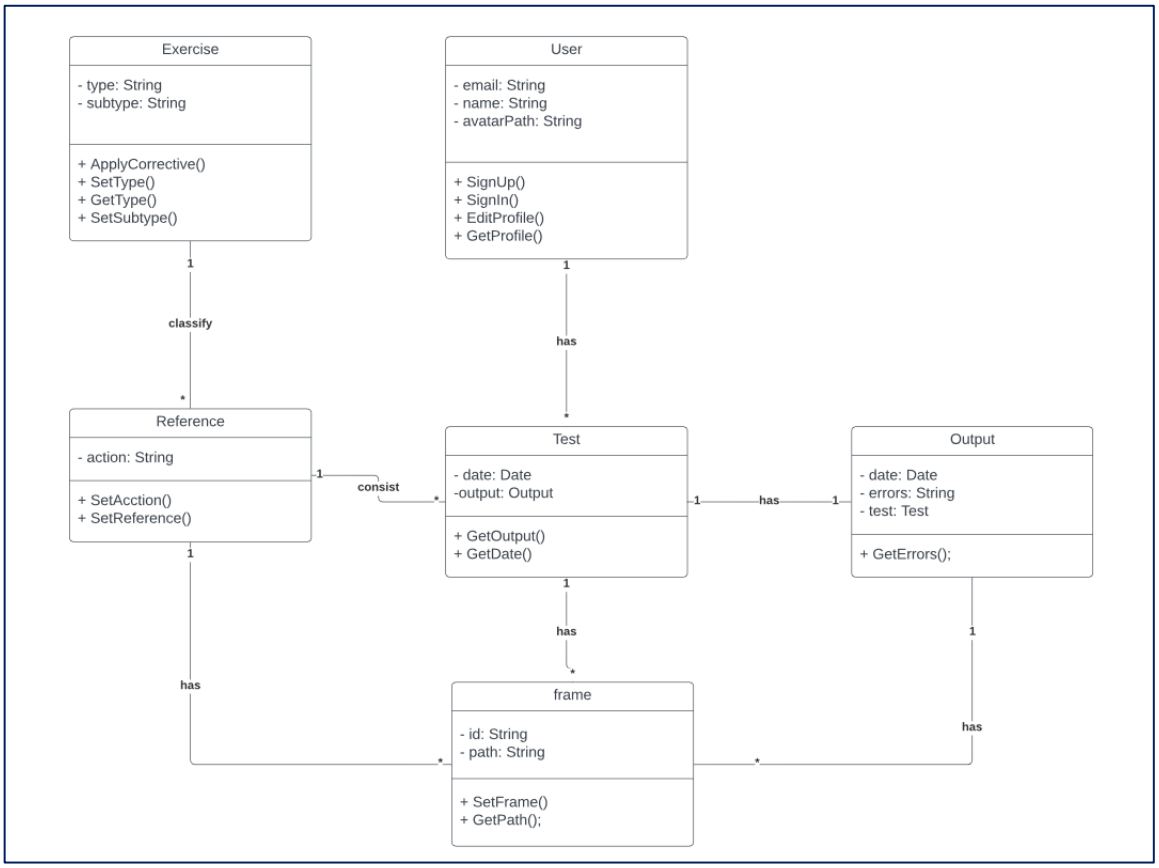
بنابراین فرایند خروجی گیری می‌بایست مستقل از موقعیت مکانی انجام می‌شد. خوشبختانه Github قابلیتی را اخیرا توسعه داده که میتوان اجرای دستورات خط فرمان را به سرورهای آن واگذار کرد. این قابلیت که Github Actions نام دارد مزیت‌های بسیاری به همراه داشت از جمله:

- کلیه پردازش‌های فرایند خروجی گیری به خارج از کامپیوتر درگیر توسعه منتقل گردید که بار سنگینی از دوش پردازنده برداشته شد.
- فرایند کاملا مستقل از سرعت اینترنت بود و حتی نیاز به VPN را نیز به طور کامل برطرف کرده بود.

فصل ۲. طراحی و تحلیل

۲.۱. نمودار کلاس

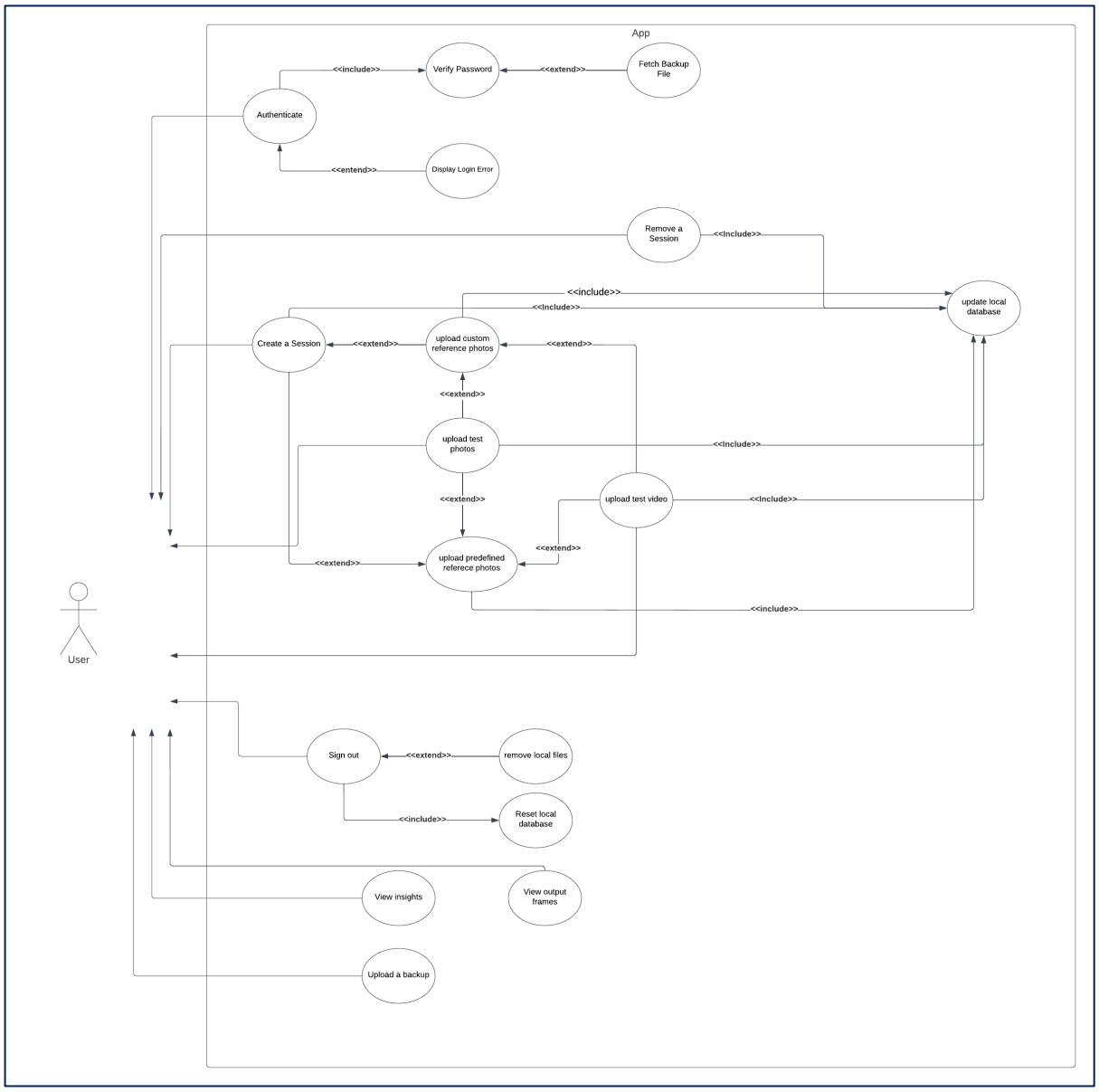
این نمودار شامل ۵ کلاس می‌باشد. کلاس User شامل نام، ایمیل و عکس کاربر می‌باشد. کلاس Exercise بخش اصلی برنامه را تشکیل میدهد و نماینده جلسات ورزشی است که کاربر ایجاد می‌کند. هر Exercise شامل یک موجودیت به نام reference است که اطلاعات مربوط به فایل های مرجع را نگهداری می‌کند. هر Reference از مجموعه ای از Frame ها تشکیل شده است که این Frame ها شامل اطلاعاتی درباره فریم هستند. پس از ایجاد یک Exercise، کاربر میتواند در برای هر Exercise یک Test ایجاد کند که در آن فریم های مربوط به حرکات خود را آپلود میکند و برای هر Reference میتواند به تعداد دلخواه Test ایجاد کند. کلاس Test نیز همانند کلاس Reference، شامل Frame‌هایی از حرکات کاربر می‌باشد. از آنجایی که یکی از اهداف این برنامه تحلیل تصاویر است، یک کلاس به عنوان Output در نظر گرفته شده است و به ازای هر Test، یک کلاس Output ساخته میشود که شامل تصاویر تحلیل شده و اطلاعات مربوط به Frame‌ها می‌باشد. بنابراین هر Test دارای یک Output و یک Reference است که هر کدام از این کلاس‌ها، شامل مجموعه ای از Frame هستند. هر Reference میتواند توسط یک یا چند Test مورد استفاده قرار گیرد و هر کاربر میتواند به تعداد دلخواه Test ایجاد کند.



تصویر ۲.۱. نمودار کلاس

۲.۲. نمودار usecase

نمودار usecase که برای این سیستم طراحی شده‌است دارای یک actor می‌باشد. کاربر از ویژگی‌های مربوط به حساب کاربری اعم از ثبت نام، ورود، خروج از حساب کاربری و عملیات پشتیبان گیری برخورد دار است. همچنین، کاربر امکان ایجاد جلسات جدید برای آپلود تصاویر مرجع و تصاویر تست را دارد و می‌تواند عملکرد خود را در غالب تصاویر پردازش شده، اسکلت حالت بدن خود، تعداد نقاط غیر منطبق و همچنین در غالب نمودارهای قابل تعامل مشاهده کند.

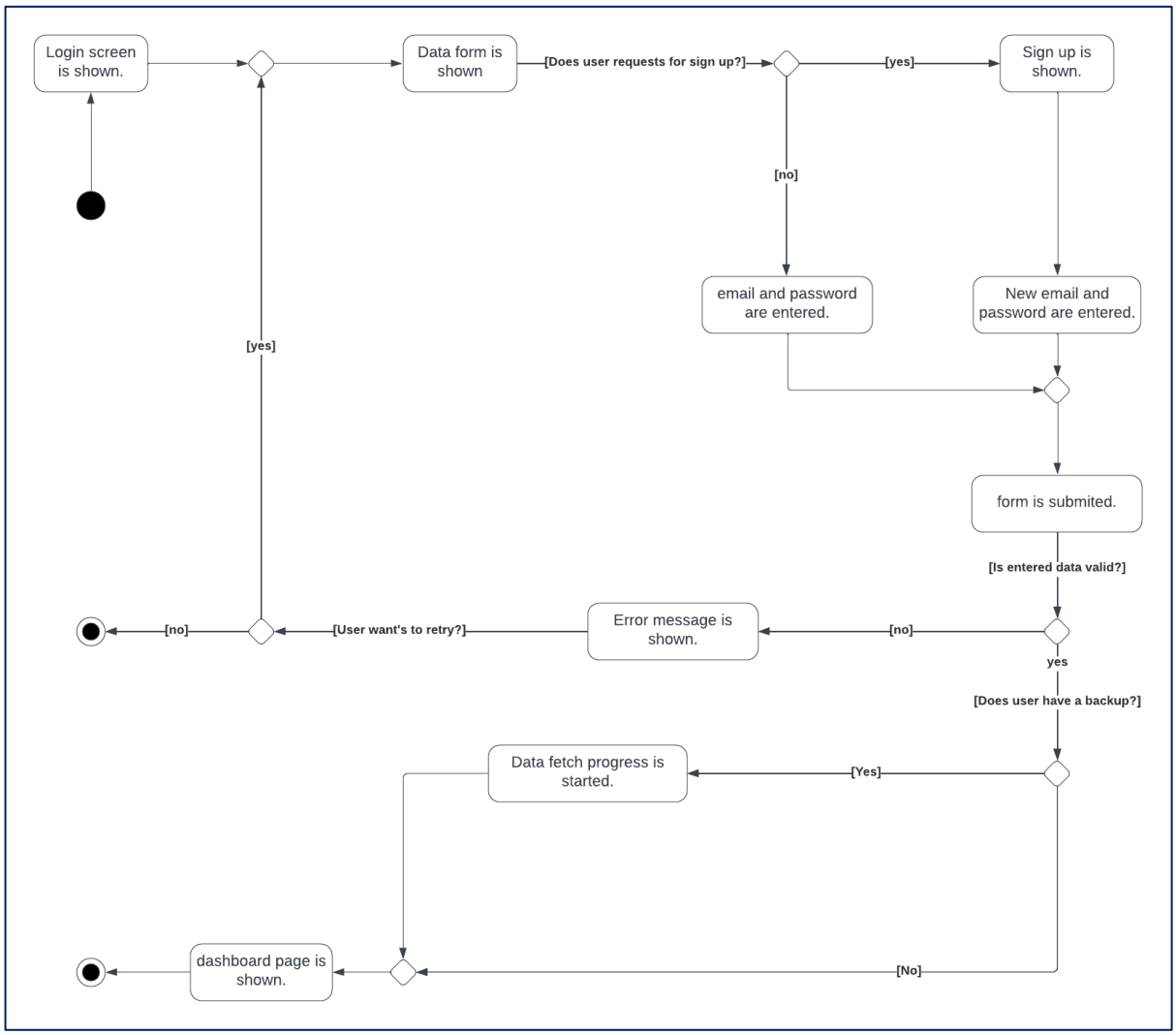


تصویر ۲.۲. نمودار usecase برنامه

۲.۳. نمودارهای فعالیت

۲.۳.۱. ثبت نام و ورود کاربر

در این بخش ابتدا کاربر با صفحه ورود کاربران مواجه می‌شود. درصورتی که کاربر بخواهد برای اولین بار از این سرویس استفاده کند، می‌تواند درخواست ثبت نام کند و در این صورت با صفحه ثبت نام مواجه می‌شود. در هر دو سناریو کاربر اطلاعات را در یک فرم وارد می‌کند و آن را ثبت می‌کند. در صورتی که اطلاعات وارد شده توسط کاربر معتبر نباشد، با پیغام خطا مواجه می‌شود و در صورت تمایل، می‌تواند دوباره تلاش کند. در غیر اینصورت، کاربر به صفحه داشبورد هدایت می‌شود و در مابین این فرایند، اگر کاربر اطلاعات خود را در فضای قبلی ذخیره‌شده داشته باشد، اطلاعات از فضای ابری بارگیری می‌شوند و کاربر در صفحه داشبورد، اطلاعات خود را می‌یابد.

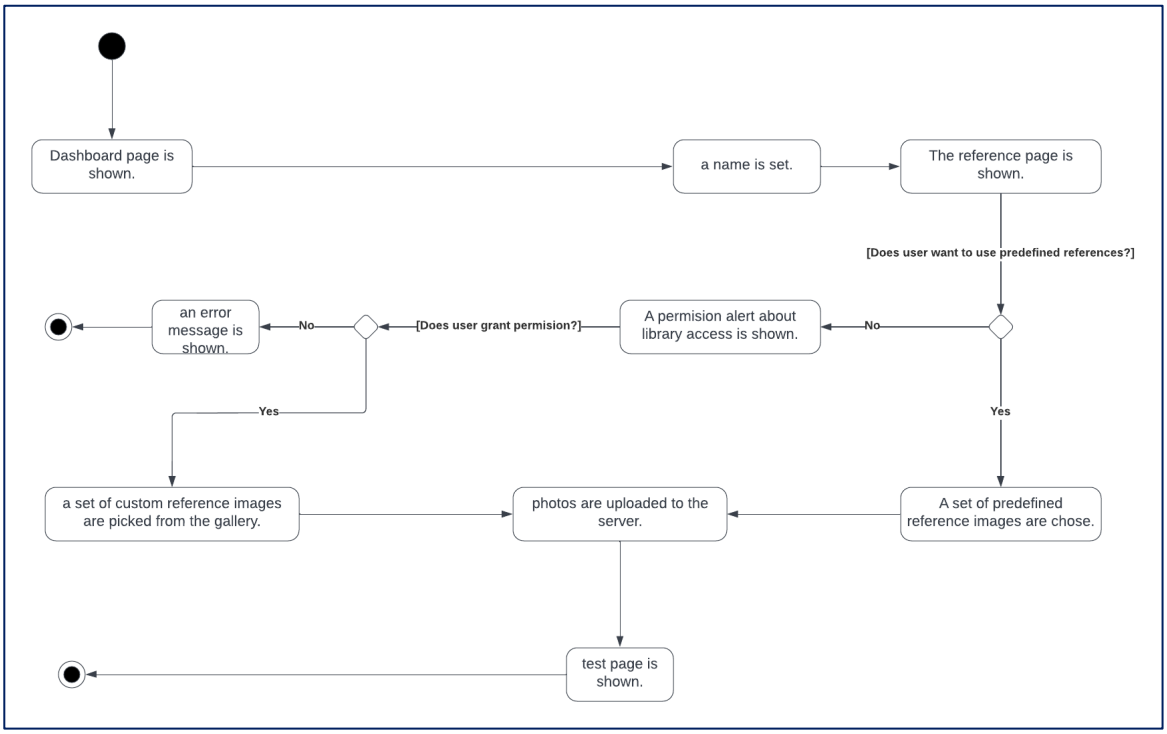


تصویر ۲.۳.۱. نمودار فعالیت فرایند ثبت نام و ورود کاربر

۲.۳.۲. ایجاد یک جلسه جدید

در این بخش یک جلسه جدید توسط کاربر ایجاد می‌گردد. ابتدا یک نام برای آن در نظر گرفته می‌شود. سپس کاربر باید تصمیم بگیرد که از تصاویر پیش فرض به عنوان تصاویر مرجع استفاده کند یا تصاویر دلخواه خود را آپلود کند. در صورتی که بخواهد تصاویر خود را از گالری انتخاب کند با پیغامی مواجه می‌شود که از او خواسته می‌شود مجوز دسترسی به گالری را برای برنامه

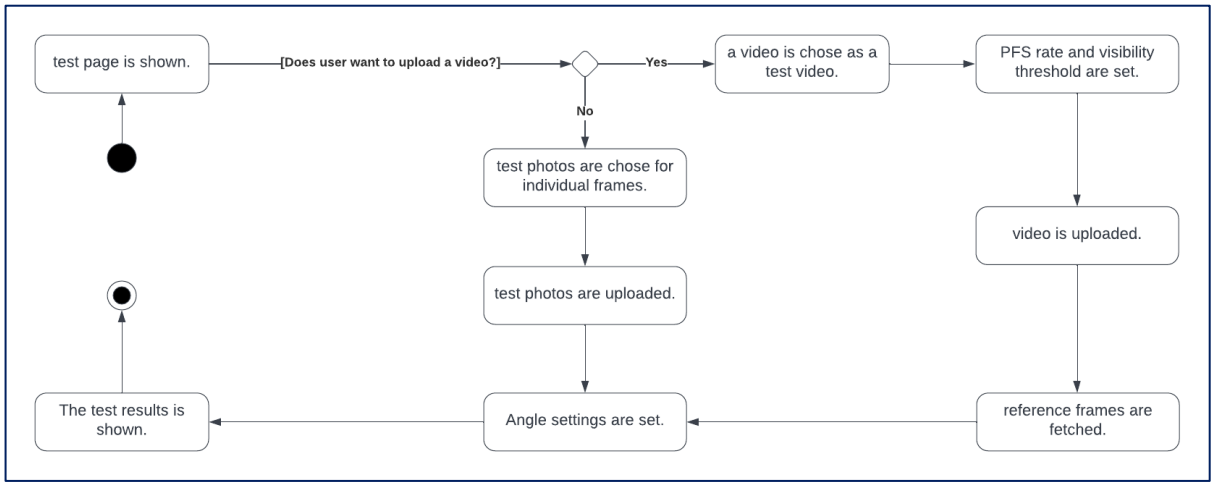
صادر کند. در صورت صدور این مجوز، کاربر قادر خواهد بود تصاویر خود را انتخاب کند. در نهایت کاربر تصاویر را اپلود می‌کند و فرایند ساخت یک جلسه اینجا به پایان می‌رسد.



تصویر ۲.۳.۲. نمودار فعالیت فرایند ایجاد یک جلسه

۲.۳.۳. ایجاد یک تست جدید

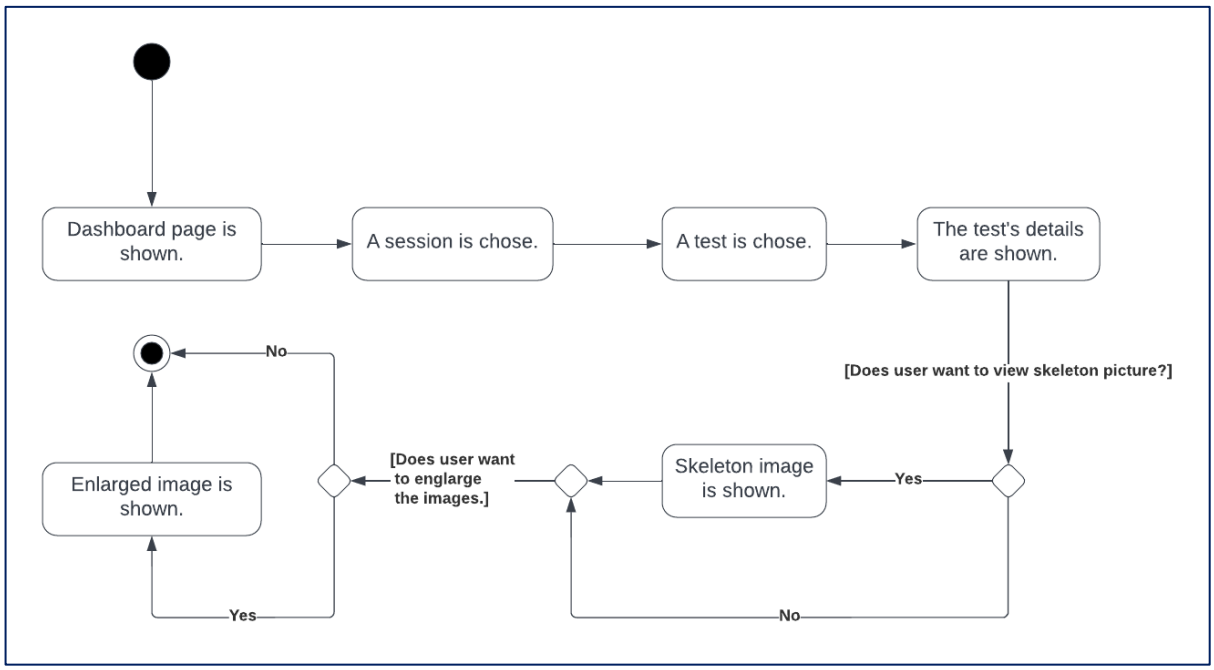
در این بخش کاربر می‌تواند از جلسه‌ای که در سناریوی پیش استفاده کرده است استفاده کند و تصاویر یا فیلم از خود که در حال انجام حرکات است، آپلود کند تا با تصاویر مرجع مقایسه شده و تحلیل گردد. فرایند آن به این صورت است که کاربر ابتدا فیلم از حرکات خود را آپلود می‌کند تا الگوریتم فریم‌های کلیدی آن را استخراج کند یا می‌تواند تصاویر را خود انتخاب کند. در مرحله بعد تنظیمات مربوط به تصحیح زاویه را انجام می‌دهد و در نهایت تصاویر تحلیل شده را دریافت می‌کند و به صفحه جزئیات آن تست هدایت می‌شود.



تصویر ۲.۳.۳. نمودار فعالیت فرایند ایجاد یک تست

۲.۳.۴. مشاهده جزئیات یک تست

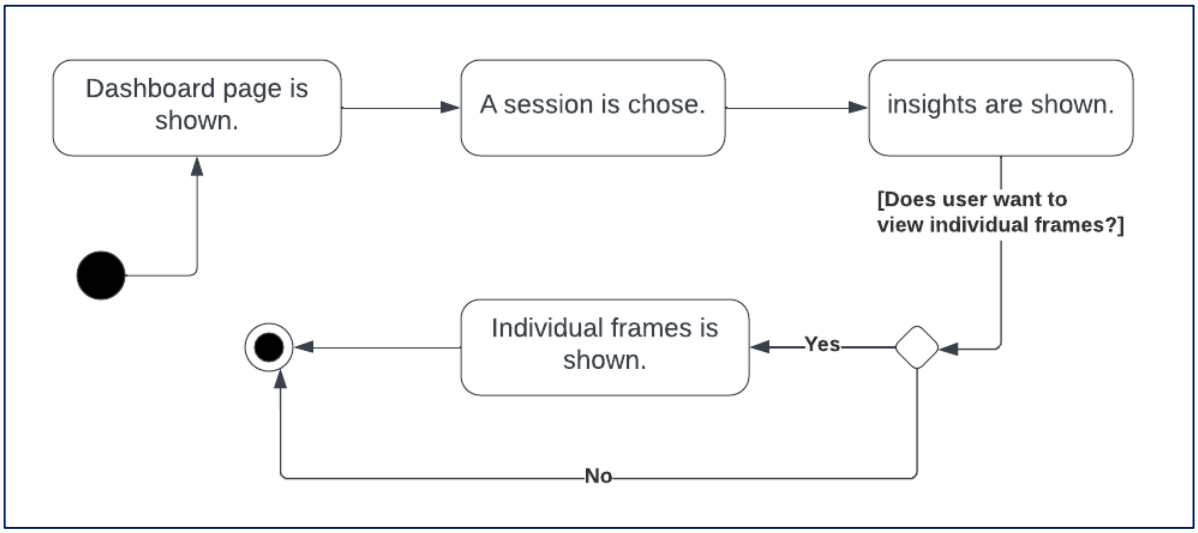
در این بخش، کاربر می‌تواند جزئیات یکی از تست‌هایی که قبلا انجام شده است را مشاهده کند. در این فرایند ابتدا کاربر یکی از جلساتی که ایجاد کرده را انتخاب می‌کند و از آنجایی که هر جلسه می‌تواند حاوی حداقل یک تست باشد، یکی از تست‌ها را انتخاب می‌کند. در قدم بعد کاربر با تصاویر تست و تصاویر مرجع مواجه می‌شود که برای مقایسه در کنار هم قرار داده شده‌اند و کاربر در صورت تمایل می‌تواند از بین ترکیب تصویر اسکلت با تصویر تست یا تصویر اسکلت خالی یکی را انتخاب کند. همچنین تعداد ایرادات هر عکس به کاربر نمایش داده می‌شود. همچنین، کاربر امکان تعامل با هر یک از عکس ها را در صورت تمایل دارد و می‌تواند آنان را بزرگنمایی کند.



تصویر ۲.۳.۴. نمودار فعالیت فرایند مشاهده جزئیات یک تست

۲.۳.۵. مشاهده نمودارهای مربوط به عملکرد کاربر در هر جلسه

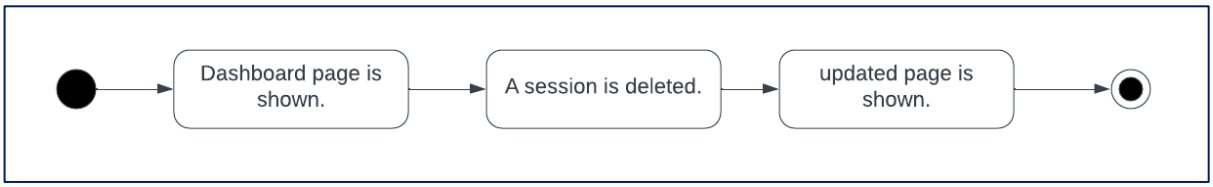
در این بخش کاربر قادر است عملکرد خود را به صورت خواناتر در قالب نمودارهای تعامل پذیر مشاهده کند. در ابتدا کاربر باید جلسه مورد نظر خود را انتخاب کند. پس از این مرحله نمودارها به کاربر نمایش داده می‌شوند و کاربر می‌تواند در صورت تمایل با آن‌ها تعامل داشته باشد؛ برای مثال می‌تواند عملکرد خود را در یکی از قسمتای حرکت خود مشاهده کند یا فقط دو بخش از حرکات خود را با یکدیگر مقایسه کند.



تصویر ۲.۳.۵. نمودار فعالیت فرایند مشاهده نمودارها

۲.۳.۶. حذف یک جلسه

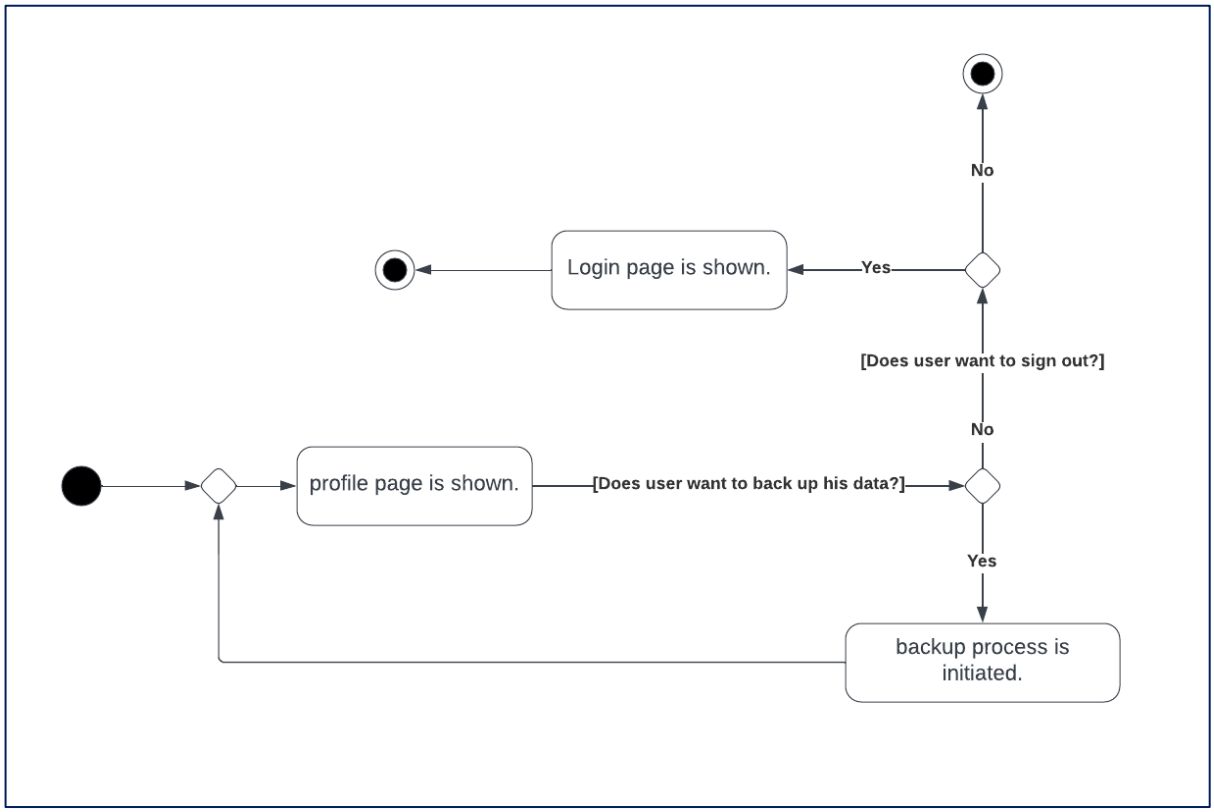
فرایند حذف یک جلسه در این بخش به این صورت است که کاربر می‌تواند لیستی از جلسات را مشاهده کند و هرکدام را که خواست حذف کند.



تصویر ۲.۳.۶. فرایند حذف یک جلسه

۲.۳.۷. تهیه نسخه کاربری، مشاهده حساب کاربری و خروج از حساب کاربری

در این بخش، کاربر اطلاعات مربوط به حساب کاربری خود را مشاهده می‌کند. سپس می‌تواند در صورت تمایل از فرایند خود یک نسخه پشتیبان تهیه کند تا درصورت تعویض دستگاه خود یا ورود به حساب کاربری در یک دستگاه دیگر بتواند بتواند فرایند پیشین خود را از سرگیری کند.



تصویر ۲.۳.۷. فرایند پشتیبان‌گیری و خروج از حساب کاربری

۲.۴. آشنایی با الگوریتم saiwa corrective exercise

جهت آشنایی با الگوریتم انواع و حالت های مختلفی از حرکات ورزشی آزمایش گردید که در ادامه به شرح آن ها پرداخته شده است. گفتنی است که نکات زیر در تعامل با این سرویس باید در نظر گرفته شود:

- زاویه فیلم مرجع با فیلم تست باید یکسان باشد به طوری که اگر در ضربه تنیس فیلمی از پشت بازیکن آپلود میکردیم امکان استفاده از الگوریتم وجود نداشت.
- تعداد خطاهای فیلم تست با تغییر تنظیمات از دو بعدی به ترکیبی از دوبعدی و سه بعدی کاهش یافت.
- حجم فایل بایستی کمتر از ۱۰ مگابایت باشد.
- در فیلم مرجع و تست انجام دهندگان حرکت باید یا راست دست یا چپ دست باشند.

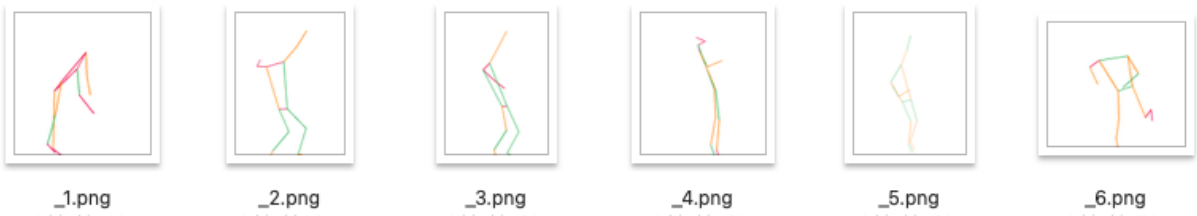
۲.۴.۱. ضربه سرویس تنیس:

در این آزمایش از مرجع پیش فرض موجود در API استفاده گردید که ۶ فریم از حرکت سرویس رافائل نادال میباشد. در ادامه حرکتی مشابه از راجرز فدرر را اپلود شد.

تصویر ۲.۴.۱. ورودی مرجع

تصویر ۲.۴.۲. ورودی تست

تصویر ۲.۴.۳. تصاویر خروجی الگوریتم

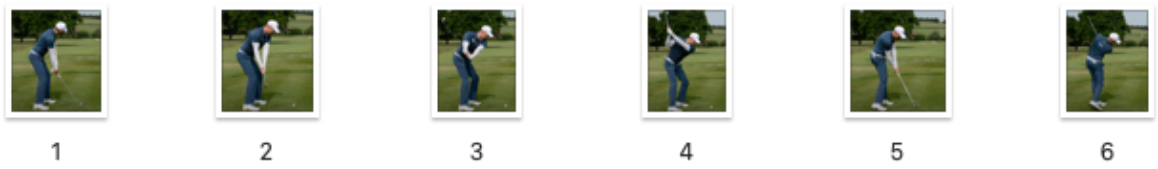


تصویر ۲.۴.۴. تصویر خروجی اسکلت

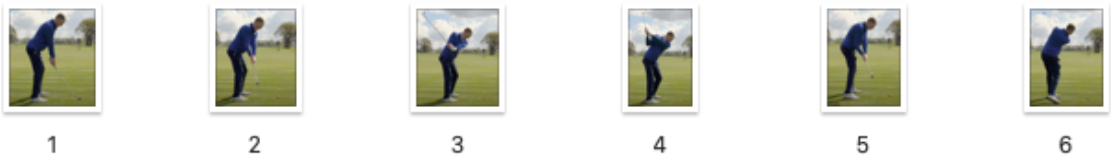
۲.۴.۲. ضربه گلف

در این حرکت هر دو نوع ورودی حرکات دو بازیکن حرفه ای گلف یکی به عنوان مرجع و دیگری به عنوان تست به API داده شد که هر یک شامل ۶ عکس میباشد .

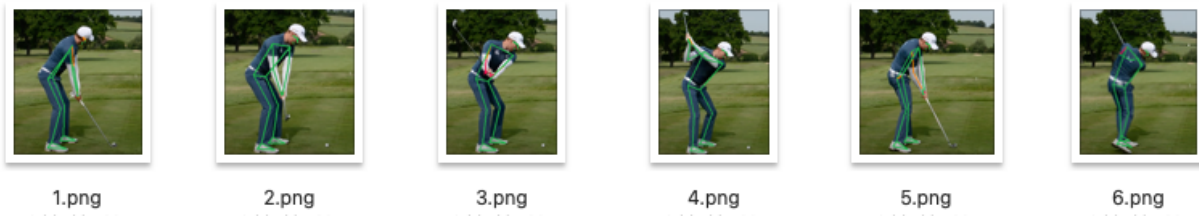
از آن جایی که هر دو نفر حرکات را با خطای کمی انجام می‌دادند اسکلت خروجی تقریبا مشابه مرجع می‌باشد.



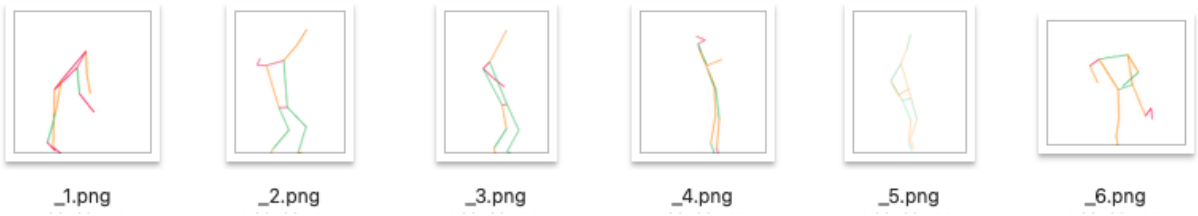
تصویر ۲.۴.۵. ورودی مرجع



تصویر ۲.۴.۶. تصاویر ورودی تست



تصویر ۲.۴.۷. تصاویر خروجی الگوریتم

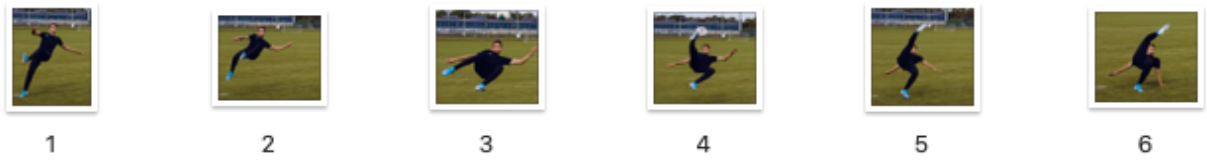


تصویر ۲.۴.۸. تصاویر خروجی اسکلت

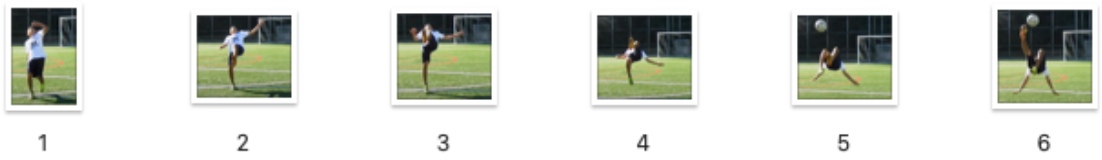
۲.۴.۳. ضربه قیچی برگردون

در این حرکت دو نوع ورودی شامل یه بازیکن حرفه ای و یک بازیکن مبتدی به ترتیب به عنوان ورودی مرجع و تست داده شد که هر یک شامل ۶ عکس از مراحل یک ضربه قیچی برگردون می‌باشد.

با توجه به این که ورودی تست یک بازیکن مبتدی میباشد تعداد خطاهای هر فریم بسیار بالا می‌باشد.



تصویر ۲.۴.۹. ورودی مرجع



تصویر ۲.۴.۱۰. تصاویر ورودی تست

تصویر ۲.۴.۱۱. تصاویر خروجی الگوریتم

تصویر ۲.۴.۱۲. تصاویر خروجی اسکلت

۳. چالش‌های پیاده‌سازی

بسته به نوع طراحی و پیاده‌سازی، نرم‌افزارها از دو بخش اساسی تشکیل شده‌اند: ۱. بخش مربوط به تعاملات کاربر و رابط کاربری ۲. بخش مربوط به فرایندهایی که توسط کاربر قابل مشاهده نیست که تحت عنوان Back-end یا منطق برنامه نیز بیان می‌گردد. پیش از آن که به بررسی چالش‌های پیاده‌سازی بپردازیم، ابتدا به بررسی بخش‌های جزئی‌تر برنامه میپردازیم.

۳.۱. مخزن[۱] پایگاه داده

این مخزن وظیفه مدیریت پایگاه داده برنامه را دارد. در این مخزن توابعی برای ایجاد یک پایگاه داده، ایجاد جداول موجودیت‌ها، درج داده‌های جدید و ویرایش آنها و همچنین توابعی برای حذف و بازگردانی کلی پایگاه داده تعریف شده است.

۳.۲. مخزن مدیریت فایل

طبیعت این برنامه به این شکل است که با فایل‌های زیادی سروکار دارد. تمام تصاویر اعم از تصاویر مرجع و تصاویر کاربر، به همراه تصاویر تحلیل شده که توسط سرور گرفته می‌شود، به صورت سازمان دهی شده در حافظه گوشی ذخیره می‌شود. در این مخزن توابعی که وظیفه ایجاد، ویرایش و حذف فایل ها را دارند تعریف شده است.

۳.۳. مخزن Firebase

همانطور که پیش‌تر مطرح شد، این برنامه برای اعتبار سنجی، ورود و ایجاد حساب کاربری از سرویس Firebase استفاده می‌کند. بنابراین یک مخزن طراحی شده است تا توابع مربوط به تعامل با این تکنولوژی را در بر بگیرد. توابع این مخزن عبارت‌اند از ایجاد حساب کاربری، ورود کاربر و خروج از حساب کاربری.

۳.۴. مخزن فراخوانی سرویس‌های تحلیل حرکات

این برنامه یکی از قسمت‌های کلیدی این برنامه، مربوط به تعامل برنامه با سرورهای سایوا می‌باشد. همه توابع مورد نیاز کاربر برای تعامل با این سرویس در این مخزن طراحی و پیاده‌سازی شده است.

۳.۵. مخزن ترجیحات کاربر

نرم‌افزارهایی که برای تلفن همراه طراحی می‌شوند، دارای مجموعه فرایندهایی هستند که ترجیحات کاربر را در ذخیره می‌کند. یکی از بخش‌هایی که این روال بسیار کاربرد دارد، بخش مدیریت حساب کاربری کاربر است. منطقا، کاربر بایستی فقط یک بار فرایند مربوط به ورود حساب کاربری را سپری کند و در صورتی که کاربر پس از راه‌اندازی مجدد برنامه مجبور باشد این فرایند را طی کند، تجربه کاربری دلپذیری نخواهد بود. بنابراین باید فرایندی طراحی شود تا اطلاعات کاربر را ذخیره کند. مخزن ترجیحات کاربر مسئول ذخیره سازی چنین اطلاعاتی است.

۳.۶. مخزن پشتیبان‌گیری

در این مخزن توابع مربوط به فراخوانی سرویس‌های Firebase Storage فراخوانی می‌شود. این مخزن شامل تمام عملیات مربوط به تهیه پشتیبان، آپلود داده‌ها به سرویس و همچنین، بارگیری داده‌ها از این سرویس است.

---

۱. Repository

۳.۷. توابع و کلاس‌های رابط کاربری

این برنامه از صفحات متعددی تشکیل شده که هر کدام از این صفحات، حاوی عناصر رابط کاربری اعم از لیست‌ها، دکمه‌ها، و سایر تعاملات هستند. هر کدام از صفحات به صورت یک کلاس تعریف میشوند و در صورتی که نیازمند طراحی یک رابط کاربری باشیم که پیچیدگی زیادی دارد و توسط صفحات دیگر نیز استفاده میشود، کلاسی جداگانه برای آن‌ها طراحی می‌شود.

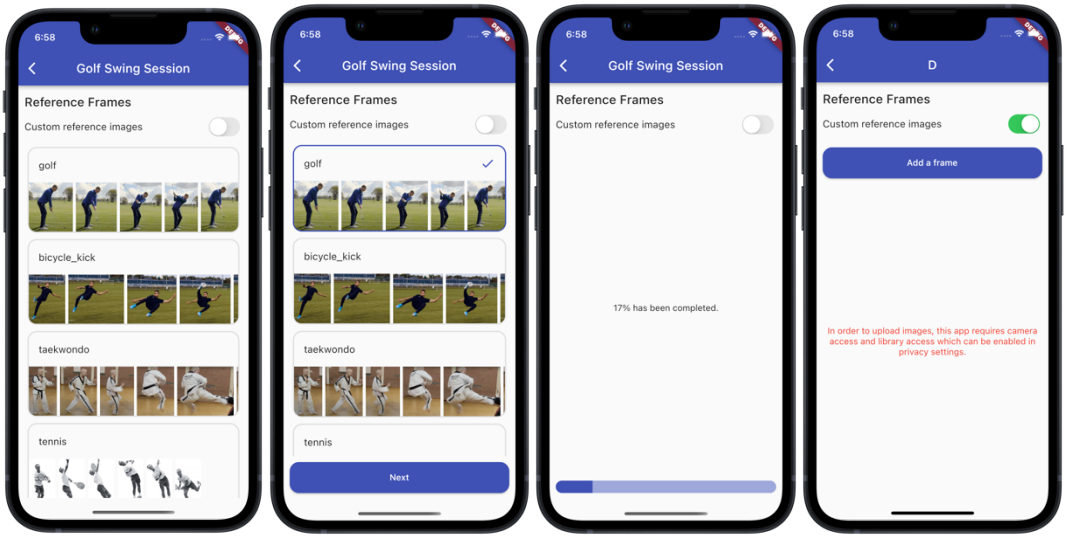
۳.۸. تزریق اصول مهندسی نرم‌افزار در طراحی کد

یکی از بحث‌هایی که در مهندسی نرم‌افزار مطرح می‌شود بیان می‌کند که وابستگی بین اجزای مختلف برنامه باید حداقل باشد و در عین حال هبستگی بین اجزای برنامه بالا باشد. دلیلی که برای این موضوع مطرح می‌شود، این است که وابستگی اجزای مختلف برنامه به یکدیگر قابلیت گسترش آن را دشوار می‌کند. برای رعایت این اصل، تمام توابعی که در یک مخزن در قالب یک کلاس قرار داده شده‌اند فقط از توابع موجود در همان کلاس استفاده می‌کنند. برای مثال در کلاس مربوط به عملیات متعدد پایگاه داده هیچ تابعی از مخزن مدیریت فایل فراخوانی نشده‌است.

۳.۹. طراحی رابطه بین توابع Back-End و رابط کاربری

یکی دیگر از چالش‌های دیگر در طراحی برنامه‌هایی که از یک رابط کاربری بهره می‌برند، طراحی یک معماری است که به بهترین شکل، اجزا و ماجول‌های مربوط به رابط کاربری و منطق برنامه را از یکدیگر جدا کند. برای درک ساده‌تر این موضوع یکی از سناریوهای برنامه را توضیح می‌دهیم.

یکی از صفحاتی که در برنامه وجود دارد، صفحه مربوط به انتخاب و آپلود فریم‌هایی از تصاویر مرجع است. کاربر می‌تواند هم از فریم‌های آماده که از پیش در برنامه تعریف شده‌است استفاده کند و هم می‌تواند فریم‌های دلخواه خود که از پیش در گالری گوشی خود ذخیره کرده‌است استفاده کند. پس از این که کاربر تصمیم گرفت از کدام تصاویر استفاده کند، تصاویر در فولدر مربوط به آن تست کپی می‌شوند، پس از تایید نهایی کاربر، عملیات مربوط به آپلود در سرویس سایوا شروع می‌شود و پس از آن، فریم‌ها در پایگاه داده ذخیره می‌شوند و در نهایت کاربر به صفحه بعد هدایت می‌شود. همچنین، در صورتی که کاربر اجازه دسترسی به گالری را به برنامه نداشته باشد، با یک پیغام خطا مواجه می‌شود که کاربر به فعال سازی دسترسی های لازم دعوت می‌کند. تصویر ۳.۱ رابط کاربری مربوط به این سناریو را نشان می‌دهد.



تصویر ۳.۱. سناریوی مربوط به انتخاب فریم‌های مرجع

همانطور که در تصویر ۳.۱ دیده می‌شود، رابط کاربری از حالات و وضعیت‌های متعددی تشکیل می‌شود که هر کدام از این حالات دارای ماجول‌های بعضا منحصر به فرد هستند و با توابع متعددی در تعامل هستند.

برای طراحی کلاسی که در آن، عناصر رابط کاربری پیاده‌سازی می‌شود، دو نوع کلاس در نظر گرفته شده‌است که باید از یک کدوم از آن‌ها ارث بری کند؛ ۱. کلاس StatefullWidget و ۲. کلاس StatelessWidget. کلاس StatefullWidget در صورتی ارث بری می‌شود که کلاس رابط کاربری دارای حالات مختلف باشد و یا لازم باشد اجزای رابط کاربری در زمان اجرا مورد تغییر قرار بگیرد. به عبارت دیگر، هنگامی که یکی از صفات یکی از عناصر رابط کاربری از قبیل رنگ، مقدار، سایز و غیره قرار باشد تغییر کند، کلاسی که این عناصر در آن قرار گرفته‌اند باید از نوع StatefullWidget باشد. در غیر اینصورت بقیه عناصر رابط کاربری که بدون حالت هستند بایستی از کلاس StatelessWidget ارث بری کنند.

در ماژول‌هایی که از کلاس StatefullWidget ارث بری می‌کنند، از تابعی به نام ()SetState استفاده می‌شود که به کمک آن می‌توان حالت متغیری که در صفحه قرار دارد را تغییر داد. برای مثال هنگامی که یک ماژول سوییچ در صفحه قرار دارد، برای این که مقدار آن را از True به False تغییر بدهیم، بایستی مقدار جدید آن را در تابع ()SetState تنظیم کنیم. با وجود این ویژگی‌ها که به صورت ذاتی در فریمورک فلاتر وجود دارند، راه حلی که در ابتدا بدیهی به نظر می‌رسد، این است که منطق برنامه را در کلاس رابط کاربری مربوط به همان صفحه فراخوانی کنیم. اما مشکلات زیادی پیرامون این نوع پیاده‌سازی وجود دارد و دلایل زیادی برای این موضوع وجود دارد که به آن ها میپردازیم.

چنین معماری پکپارچگی، قابلیت نگهداری و قابلیت گسترش ندارد. هنگامی که پروژه وارد ابعاد بزرگتر می‌شود معماری‌ها باید متناسب با آن انتخاب شود. برای این برنامه معماری BLoC انتخاب گردید که به دنبال آن نیاز برنامه به ویجت های StatefullWidget تا حد زیادی برطرف می‌شود که در قسمت بعدی به طور مفصل به آن پرداخته خواهد شد.

۳.۱۰. بلاک

بلاک[۲] کوتاه شده عبارت **B**usiness **Lo**gic **C**omponents می‌باشد که در مدیریت حالات مختلف برنامه و معماری آن استفاده گردید. در واقع بلاک واسطه ای میان رابط کاربری و منطق برنامه می‌باشد. با استفاده از این معماری، نه تنها سرعت طراحی و توسعه برنامه افزایش می‌یابد، بلکه کدها قابلیت گسترش بیشتری پیدا می‌کنند و خوانایی بیشتری خواهند داشت. همچنین تمام حالات برنامه به صورت واضح برای توسعه دهندگان معین میگردد.

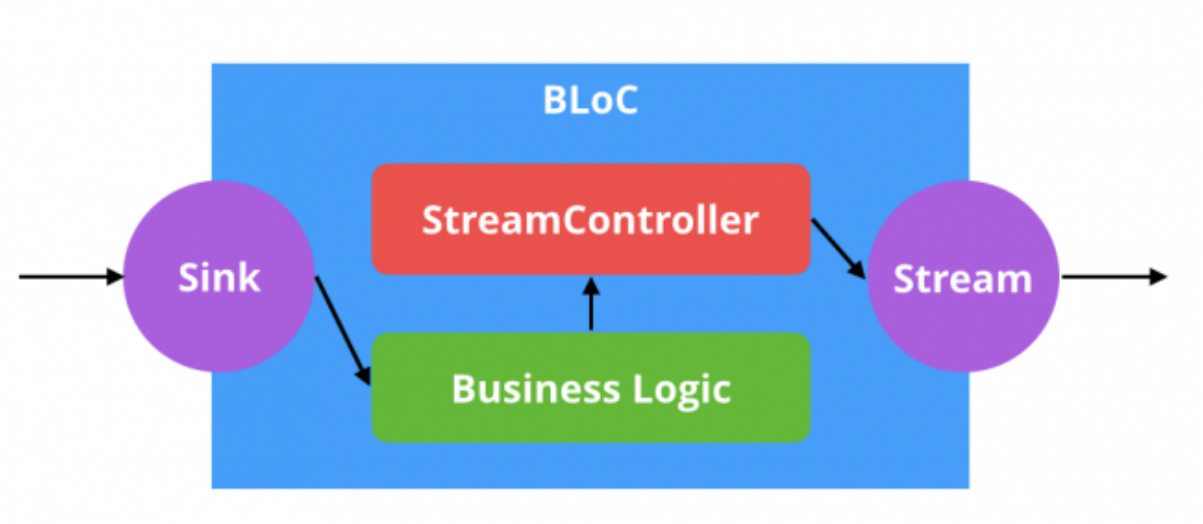
عناصر اصلی بلاک عبارت اند از:

- رویدادها[۳]: رویدادها ورودی‌های بلاک هستند و در رابط کاربری فراخوانی می‌شوند. هنگامی که رویدادی به بلاک اضافه می‌شود، انتظار می‌رود که در ازای آن یک حالت از بلاک خارج شود.

۲. BLoC
۳. Events

- حالت‌ها[۴]: بیانگر حالات مختلف برنامه می‌باشند. کلیه عناصر رابط کاربری به حالت کنونی اشراف دارند و با تغییر آن خود را به روز رسانی می‌کنند.
- بلاک: بخش اصلی این معماری است که جریانی[۵] از رویدادها رو به جریانی از حالات تبدیل می‌کند. بلاک مانند ذهنی است که اطلاعات را دریافت، پردازش و در نهایت پاسخ می‌دهد.

جریان به دنباله ای از داده های همروند اطلاق می‌شود. رابط کاربری و بلاک به استریم گوش فراداده و در مقابل هر تغییری واکنش نشان می‌دهند.



تصویر ۳.۲. ترسیمی از معماری بلاک

حال به مثالی که پیشتر مطرح کردیم باز می‌گردیم. هنگامی که وارد این صفحه می‌شویم بلاک رویداد PresetInitial را فراخوانی می‌کند تا داده ها بارگذاری شوند. پس از آن که داده ها و عکس ها آماده شد به کاربر نشان داده می‌شوند. سپس کاربر می‌تواند هر یک از آن ها را انتخاب کند که با انتخاب هر یک مجدد رویدادی فراخوانی می‌شود که منجر به انتخاب آن گروه از عکس‌ها می‌گردد که حاصل تغییر حالتی است که در بلاک طراحی شده است. با زدن دکمه Next رویداد Upload فراخوانی می‌شود که از آن جایی که صبر کاربر را می‌طلبد دکمه به ویجت لود کامل شونده تبدیل می‌گردد تا کاربر متوجه شود نیازی نیست دکمه مجددا فشرده شود. لازم به ذکر است که بالای صفحه سوییچی قرار گرفته که با زدن آن بارگزاری عکس های مرجع به کاربر واگذار می‌گردد.

یک بلاک تحت عنوان ReferenceSetupEvent برای این صفحه در نظر گرفته می‌شود و حالات و رویدادها متناسبا تعریف می‌شود. رویدادها و حالات را در تصویر ۳.۳ می‌توانید مشاهده کنید.

۴. States
۵. Stream

تصویر ۳.۳. رویدادها و حالات تعریف شده در فرایند ایجاد جلسه

۴. نرم‌افزار Stamina

۴.۱. صفحه جلد[۶]

جلد برنامه که splash نام دارد در واقع اولین صفحه ای است که در اجرای برنامه نمایش داده می‌شود که بهترین مکان برای پردازش اطلاعات اولیه‌ای است که مسیر برنامه را مشخص می‌کند. در این صفحه ابتدا با چک کردن حافظه محلی برنامه در گوشی مشخص می‌شود که آیا حساب کاربری وجود دارد یا خیر. چنان چه کاربری وجود داشته باشد مستقیما وارد صفحه اصلی برنامه می‌شود در غیر این صورت وارد صفحه ورود و ثبت نام می‌شود.

۴.۲. صفحه ورود به حساب کاربری و ثبت نام کاربران جدید

اگر برنامه برای بار اول اجرا شود کاربر مستقیم وارد صفحه ورود می‌شود. چنان چه کاربر جدید باشد بایستی ابتدا ثبت نام کند، در غیر این صورت با وارد کردن ایمیل و رمز می‌تواند وارد صفحه اصلی شود. دکمه ثبت نام یا ورود تنها هنگامی فعال می‌شود که اطمینان حاصل کند رمز وارد شده امن و ایمیل از یک الگوی معتبر پیروی می‌کند. بررسی این مساله در سمت کلاینت باعث می‌شود تا حد امکان از فرستادن پرس و جوی بیهوده به سرور Firebase جلوگیری شود. رمز امن، رشته ای است که حداقل شامل شش کاراکتر باشد. کلیه عملیات مربوط به ثبت نام و ورود کاربر از طریق سرویس Firebase Authentication مدیریت می‌شود. Firebase در سایت مربوط به این امکانات که از آن به عنوان کنسول Firebase یاد می‌شود، داشبوردی در اختیار ما قرار می‌دهد که امکان مشاهده کاربران و اطلاعات آنان وجود دارد. پس از آن اطلاعات کاربر به صورت محلی در حافظه موبایل ذخیره می‌شود.

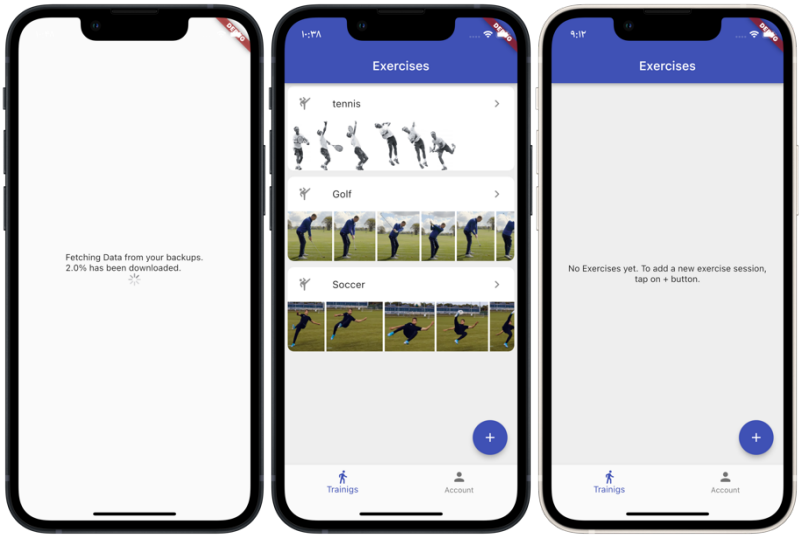
تصویر ۴.۱. صفحه ورود و ثبت‌نام

۴.۳. صفحه اصلی

پس از این که کاربر با موفقیت عملیات ورود به برنامه را سپری کرد، وارد مرحله‌ای می‌شود که برنامه با سرویس Firebase Storage ارتباط برقرار می‌کند و چک می‌کند که آیا کاربر نسخه پشتیبانی در فضای ابری دارد یا خیر. در صورتی که کاربر از قبل، عملکرد خود را در فضای ابری پشتیبان‌گیری کرده باشد، عملیات بارگیری اطلاعات کاربر که شامل پایگاه داده و عکس‌های

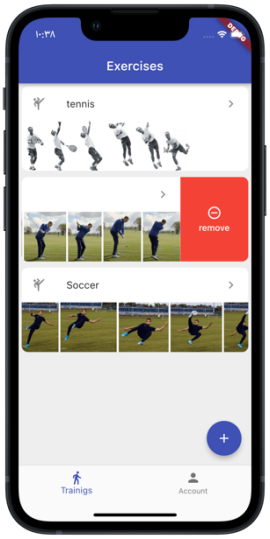
۶. Splash Screen

حرکات کاربر است، آغاز می‌گردد و در صفحه اصلی برنامه کاربر اطلاعات خود را می‌یابد. در غیر این صورت کاربر با یک صفحه خالی از جلسات مواجه می‌شود که او را به ایجاد یک جلسه جدید دعوت می‌کند. صفحه اصلی دارای یک دکمه است به کاربر این امکان را می‌دهد تا جلسات جدید را اضافه نماید.



تصویر ۴.۲. صفحه بارگیری نسخه پشتیبان(چپ)، صفحه حاوی اطلاعات کاربر(وسط)، صفحه اصلی عاری از اطلاعات(راست)

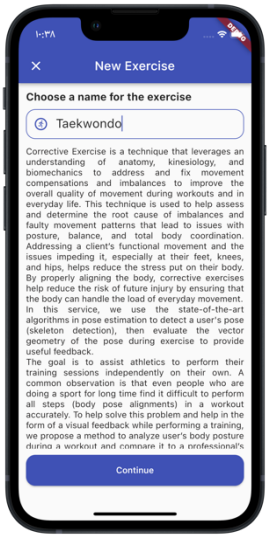
در این صفحه، لیست جلساتی که کاربر ایجاد کرده‌است از پایگاه داده گرفته می‌شود و نام آن جلسه نمایش داده می‌شود. همچنین، هر جلسه دارای یک شناسه منحصر به فرد است که به واسطه این شناسه، عکس‌های مربوط به هر جلسه مشخص می‌شوند. همزمان، شناسه‌های جلسات از پایگاه داده گرفته می‌شود و عکس‌های مرجع مربوط به آن جلسات از حافظه گوشی خوانده شده و نمایش داده می‌شود. لازم به ذکر است که کاربر امکان حذف جلسات را دارد. برای حذف جلسات کاربر باید جلسه مورد نظر را به سمت چپ سر بدهد و در صورت تایید نهایی، گزینه حذف را لمس کند. در این صورت، جلسه از پایگاه داده حذف می‌شود و تمام عکس‌های مربوط به آن جلسه از حافظه گوشی حذف می‌گردند.



تصویر ۴.۳. قابلیت حذف جلسه توسط کاربر

۴.۴. صفحه ایجاد جلسه

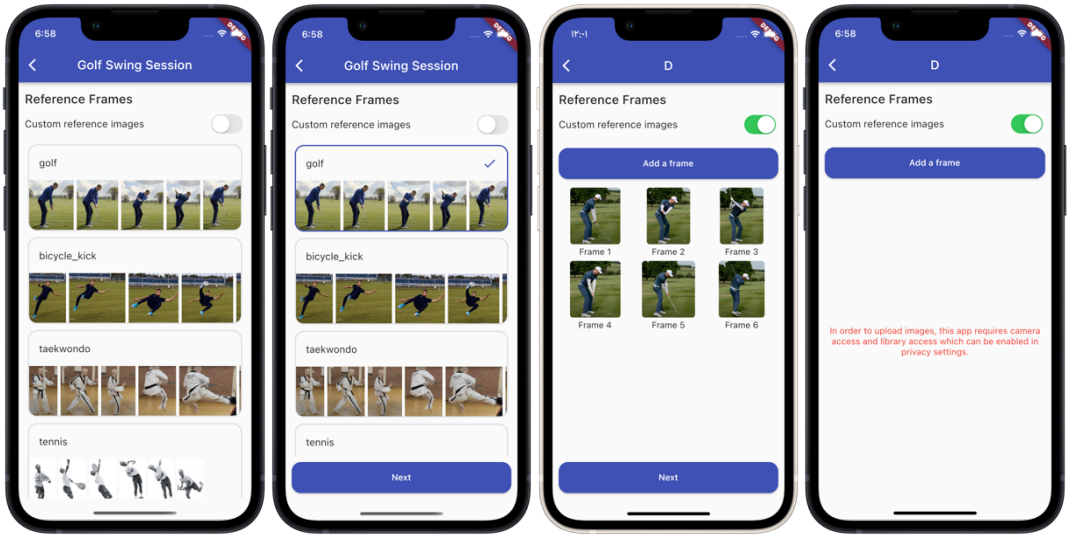
هنگامی که کاربر دکمه + را می‌فشارد، به صفحه‌ای منتقل می‌شود که از او خواسته می‌شود تا نامی برای جلسه‌ای که در حال ایجاد آن است، تعریف کند. در میان صفحه، مجموعه‌ای از توضیحات درباره عملکرد این الگوریتم داده می‌شود و هنگامی که کاربر نامی برای این جلسه تعریف کرد، می‌تواند دکمه Next را لمس کند. در این صورت یک جلسه در پایگاه داده ایجاد می‌گردد و به آن شناسه تعلق می‌گیرد. همچنین، یک پوشه در حافظه گوشی برای ذخیره‌سازی عکس‌های مربوط به این جلسه ایجاد می‌شود.



تصویر ۴.۴. صفحه ایجاد جلسه

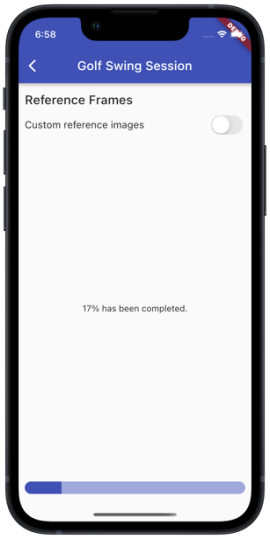
۴.۵. صفحه انتخاب تصاویر مرجع

در این صفحه از کاربر خواسته می‌شود تصاویر مرجع را برای این جلسه انتخاب کند. هر جلسه از مجموعه‌ای از تصاویر مرجع تشکیل شده است که برای تمام تست‌های آن جلسه مورد استفاده قرار می‌گیرد. کاربر می‌تواند به انتخاب خود، یا از تصاویر از پیش آماده استفاده کند، یا تصاویر دلخواه خود را از گالری گوشی خود انتخاب کند. اگر کاربر بخواهد از تصاویر از پیش آماده استفاده کند، بایستی یکی از مجموعه تصاویر را که در زمینه‌های مختلفی هستند، انتخاب کند تا بتواند به مرحله بعد برود. در غیر اینصورت باید مجوز دسترسی به گالری را به این برنامه صادر کند تا بتواند به گالری گوشی دسترسی داشته باشد. اگر کاربر مجوز را صادر کرد، تصاویر انتخاب شده با شماره فریم نمایش داده می‌شوند و اگر از صدور مجوز امتناع کرد، با پیغام خطایی مواجه می‌شود که راهنمایی لازم را برای صدور مجوز به کاربر نشان می‌دهد.



تصویر ۴.۵. حالات مختلف صفحه انتخاب تصاویر مرجع از چپ به راست:
هنگامی که کاربر هنوز هیچ عکس پیش‌فرضی را انتخاب نکرده‌است، هنگامی که کاربر یک مجموعه از تصاویر را انتخاب کرده‌است، هنگامی که کاربر تصاویر مرجع را از گالری انتخاب می‌کند، هنگامی که کاربر از صدور دسترسی به گالری گوشی امتناع می‌کند.

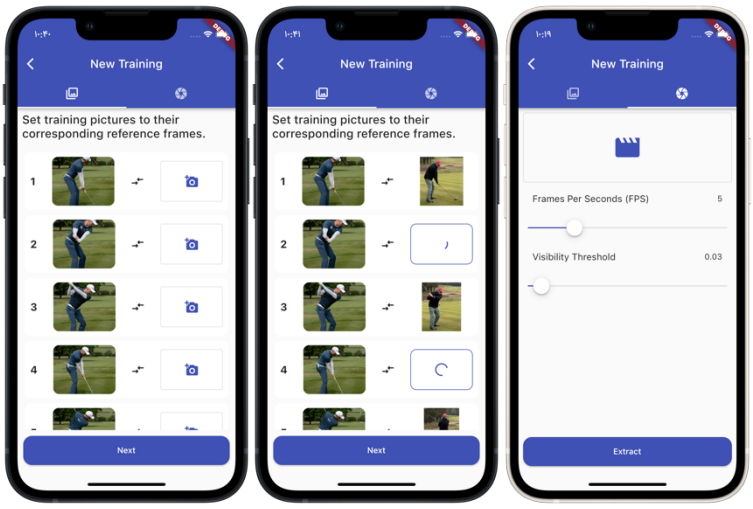
هنگامی که کاربر با موفقیت تصاویر مورد نظر خود را انتخاب کرد، بر دکمه Next را لمس می‌کند. در این حالت فرایند بارگذاری تصاویر به سرویس سایوا آغاز می‌شود و میزان پیشرفت آن به کاربر نمایش داده می‌شود. هنگام اتمام این فرایند، تصاویر در حافظه گوشی در پوشه مربوط به این جلسه ذخیره می‌شوند و پایگاه داده مربوط به این جلسه به روز نشانی می‌شود. در نهایت کاربر به صفحه بعدی هدایت می‌شود.



تصویر ۴.۶. فرایند بارگذاری در سرویس سایوا

۴.۶. صفحه بارگذاری تست

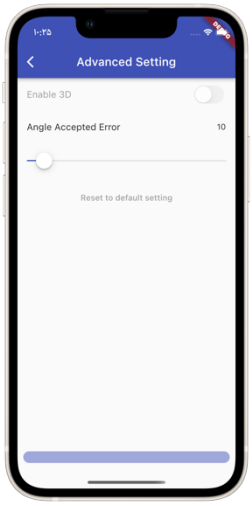
در این صفحه کاربر بایستی تست‌های خود را آپلود کند. کاربر می‌تواند تست را به صورت مجموعه‌ای از فریم‌های متناظر با تصاویر مرجع یا به صورت فیلم آپلود کند تا الگوریتم خود فریم‌ها را استخراج کند. هنگامی که کاربر بخواهد فریم‌ها را آپلود کند، باید فریم‌ها را جدا جدا انتخاب کند و در این صورت ابتدا تصاویر به سرویس فرستاده می‌شوند؛ سپس پایگاه داده به‌روزرسانی می‌شود. هنگامی که کاربر فیلم را انتخاب می‌کند، فیلم به سرویس فرستاده می‌شود، سپس فریم‌ها به حافظه گوشی بارگیری می‌شوند و پایگاه داده به‌روزرسانی می‌گردد. لازم به ذکر است که کاربر می‌تواند تنظیمات مربوط به استخراج فریم‌ها از فیلم اعم از تعداد فریم های استخراج شده در ثانیه و آستانه دید برای مفاصل اسکلت خروجی را شخصی سازی کند.



تصویر ۴.۷. صفحه انتخاب و بارگذاری تست

۴.۷. صفحه تنظیمات فریم

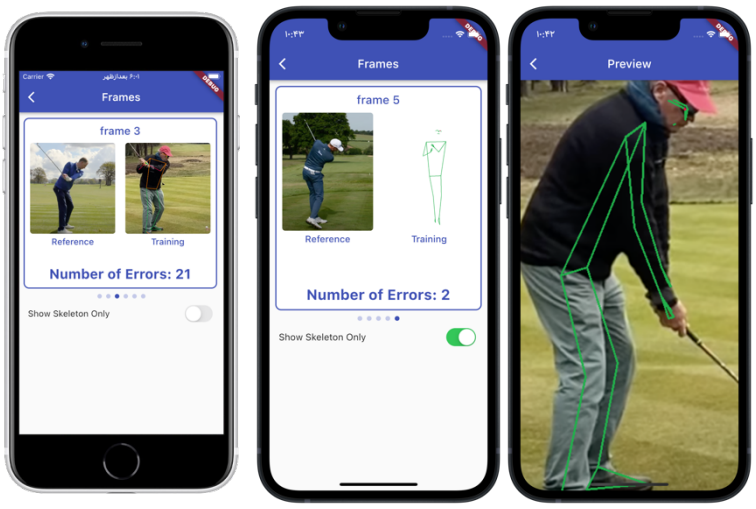
در این صفحه، کاربر می‌تواند میزان اختلاف درجه بین تصویر مرجع و تصویر تست را مشخص کند. پس از اعمال تنظیمات عملیات بارگذاری و بارگیری تصاویر آغاز می‌گردد و میزان پیشرفت آن به کاربر نشان داده می‌شود. در این فرایند پایگاه داده به‌روزنشانی می‌شود و تصاویر در حافظه گوشی ذخیره می‌شوند.



تصویر ۴.۸. صفحه تنظیمات مربوط به فریم(چپ) و عملیات بارگذاری و بارگیری(راست)

۴.۸. صفحه مشاهده جزئیات نتیجه تست

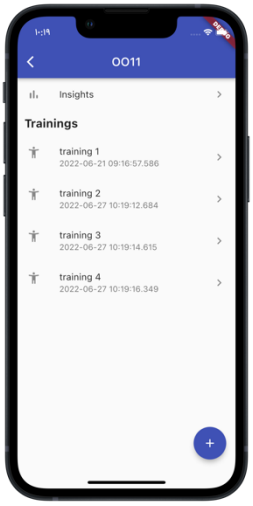
در این صفحه، کاربر فریم‌های مربوط به حرکت خود را در کنار تصویر مرجع متناظر با آن می‌بیند. کاربر می‌تواند فریم‌ها را به ترتیب ببیند و بین فریم‌های متعدد جابه‌جا شود. تعداد نقاطی که حرکات کاربر با حرکات تصویر مرجع منطبق نبوده‌است نیز نمایش داده می‌شود. این اطلاعات از حافظه گوشی خوانده می‌شوند. کاربر می‌تواند در صورت تمایل، اسکلت بدن خود را در هنگام اجرای حرکت مشاهده کند. لازم به ذکر است که تصاویر قابلیت بزرگنمایی دارند و هنگامی که کاربر یکی از تصاویر را لمس کند، می‌تواند آن را بزرگنمایی کند.



تصویر ۴.۸. صفحه جزئیات عملکرد کاربر

۴.۹. صفحه تست‌های کاربر

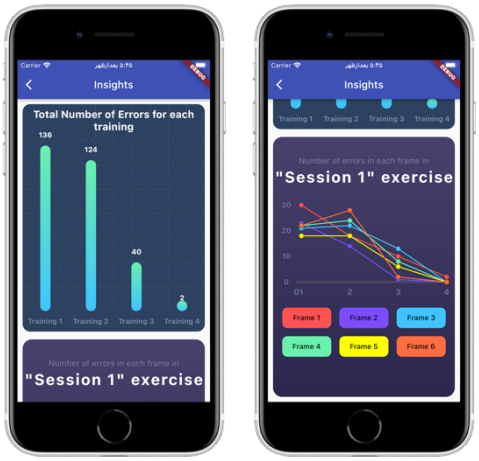
در این صفحه، کاربر لیستی از تست‌های خود را که برای یک جلسه ایجاد کرده مشاهده می‌کند. این تست‌ها از پایگاه داده خوانده می‌شوند و دارای تاریخ و شناسه‌ای هستند که به آن جلسه مرتبط شده‌اند. کاربر می‌تواند با لمس دکمه + یک تست دیگر ایجاد کند و با لمس گزینه Insights نمودار عملکرد خود را مشاهده کند.



تصویر ۴.۹. صفحه تست‌ها

۴.۱۰. صفحه نمودارها

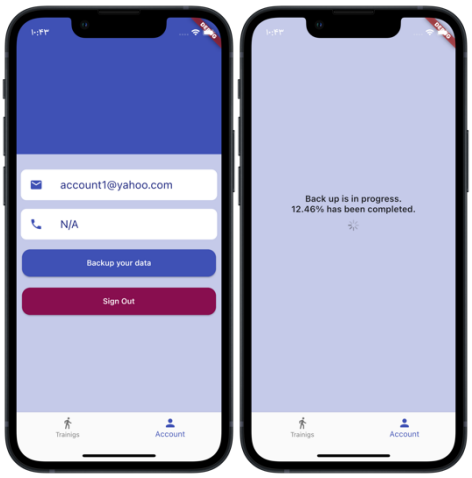
در این صفحه کاربر با دو نوع نمودار قابل تعامل سروکار دارد. نمودار اول، یک نمودار میله‌ای است که تعداد خطاهای کاربر را در تست‌های مختلف نشان می‌دهد. نمودار دوم، یک نمودار خطی است که پیشرفت کاربر را در هر یک از فریم‌ها نشان می‌دهد. کاربر می‌تواند فریم‌هایی را که می‌خواهد پیشرفت آن را بررسی کند انتخاب کند و آنان را با یکدیگر مقایسه می‌شود. هر نقطه از این نمودار را که کاربر لمس کند، می‌تواند مقدار دقیق خطاها را مشاهده کند.



تصویر ۴.۱۰. نمودار میله‌ای (چپ) و نمودار خطی (راست)

۴.۱۱. صفحه حساب کاربری

در این صفحه اطلاعات کاربر نمایش داده می‌شود. در این مرحله امکان درخواست پشتیبان‌گیری از اطلاعات خود را دارد. در صورتی که کاربر دکمه backup را لمس نماید، عملیات پشتیبان‌گیری آغاز می‌گردد. پایگاه داده و تصاویر به صورت فایل zip در می‌آیند به به سرویس Firebase Storage بارگذاری می‌شوند. همچنین، کاربر می‌تواند از حساب کاربری خود خارج شود و در این صورت، تمام اطلاعات کاربر از حافظه گوشی حذف می‌شود و پایگاه‌داده به حالت اولیه بازنشانی می‌گردد.



تصویر ۴.۱۱. صفحه حساب کاربری

۵. ضمائم
۵.۱. کدهای مربوط به کلاس‌های مدل
۵.۱.۱. فایل exercise

```
class Exercise {
  int? id;
  final String name;

  Exercise({
    this.id,
    required this.name,
  });

  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'name': name,
    };
  }

  @override
  String toString() {
    return 'Exercise{id: $id, name: $name}';
  }
}
```

۵.۱.۲. فایل extraction_configue

```
import 'dart:io';

import 'package:saiwa_app/models/enums/reference_type.dart';

class ExtractConfigue {
  final int fps;
  final double visibilityThreshold;
  final ReferenceType exerciseTypeId;
  final String videoGuid;
  final List referenceList;
  final String fileType;

  ExtractConfigue({
    required this.fileType,
    required this.fps,
    required this.visibilityThreshold,
    required this.exerciseTypeId,
    required this.videoGuid,
    required this.referenceList,
  });
}
```

۵.۱.۳. فایل frame

```
class Frame {
  final String guid;
  String fileName;
  final String mimeType;
  final int fileSize;
  final String extension;
  final bool isUsed;
  final String? dlFiles;
  final String? serviceRunFiles;
  final int id;
  Training? training;
```

```
  Exercise? exercise;

  Frame({
    required this.guid,
    required this.fileName,
    required this.mimeType,
    required this.fileSize,
    required this.extension,
    required this.isUsed,
    this.dlFiles,
    this.serviceRunFiles,
    required this.id,
    this.training,
    this.exercise,
  });

  Map<String, dynamic> toMap() {
    return {
      'guid': guid,
      'fileName': fileName,
      'mimeType': mimeType,
      'fileSize': fileSize,
      'extension': extension,
      'isUsed': isUsed ? 1 : 0,
      'dlFiles': dlFiles,
      'serviceRunFiles': serviceRunFiles,
      'id': id,
      'training_id': training?.id,
      'exercise_id': exercise?.id,
    };
  }

   Map<String, dynamic> toJson() {
    return {
      "guid": guid,
      "fileName": fileName,
      "mimeType": mimeType,
      "fileSize": fileSize,
      "extension": extension,
      "isUsed": isUsed ? 1 : 0,
      "dlFiles": dlFiles,
      "serviceRunFiles": serviceRunFiles,
      "id": id,
      "training_id": training?.id,
      "exercise_id": exercise?.id,
    };
  }

  Frame.fromJson({
    required Map<String, dynamic> json,
    this.exercise,
    this.training,
  })  : guid = json['guid'],
        fileName = json['fileName'],
        mimeType = json['mimeType'],
        fileSize = json['fileSize'],
        extension = json['extension'],
        isUsed = json['isUsed'],
        dlFiles = json['dlFiles'],
        serviceRunFiles = json['serviceRunFiles'],
        id = json['id'];
}
```

۵.۱.۴. فایل training

```
import 'package:saiwa_app/models/exercise.dart';
class Training {
  Exercise exercise;
  DateTime? dateTime;
  int? id;

  Training({this.id, required this.exercise}) {
    dateTime = DateTime.now();
  }
  Training.fromTraining({
    required this.id,
    required this.exercise,
    required String dataTimeTimeStamp,
  }) {
    this.dateTime =
        DateTime.fromMillisecondsSinceEpoch(int.parse(dataTimeTimeStamp));
  }
  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'date': dateTime!.millisecondsSinceEpoch.toString(),
      'reference_id': exercise.id
    };
  }
}
```

۵.۲. کدهای مربوط به کلاس‌های مخزن

۵.۲.۱. فایل auth_repo

```
import 'package:firebase_auth/firebase_auth.dart';
class AuthRepository {
  static User? get user => FirebaseAuth.instance.currentUser;
  static void updateUserName(String name) {
    User? user = FirebaseAuth.instance.currentUser;
    if (user != null) {
      user.updateDisplayName(name);
    }
  }

}
```

۵.۲.۲. فایل authentication

```
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:saiwa_app/repository/prefs.dart';
import 'package:shared_preferences/shared_preferences.dart';

class Authentication {
  static Future<http.Response> signIn(String email, String password) async {
    http.Response response = await http.post(
      Uri.parse('https://services.saiwa.ai/api/user/Signin'),
      headers: {"Content-Type": "application/json"},
      body: json.encode({'username': email, 'password': password, 'type': 1}),
    );
    return response;
  }

  static Future<http.Response> autoSignIn() async {
    try {
      String user = EMAIL;
```

```
      String password = PASSWORD;
      http.Response response = await signIn(user, password);
      return response;
    } catch (e) {
      print(e);
    }
    return http.Response("", 1000);
  }

  static Future<http.Response> refreshToken() async {
    try {
      String token = await Prefs.refreshToken;
      http.Response response = await http.post(
        Uri.parse('https://services.saiwa.ai/api/User/refreshToken'),
        headers: {"Content-Type": "application/json"},
        body: json.encode({'refreshToken': token}),
      );
      return response;
    } catch (e) {
      print(e);
    }
    return http.Response('{}', 500);  }

  static Future<http.Response> signOut() async {
    String token = await Prefs.refreshToken;
    http.Response response = await http.post(
      Uri.parse('https://services.saiwa.ai/api/User/Signout'),
      headers: {"Content-Type": "application/json"},
      body: json.encode({'refreshToken': token}),
    );
    return response;
  }
}
```

۵.۲.۳. فایل corrective_repo

```
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:saiwa_app/models/corrective.dart';
import 'package:saiwa_app/repository/authentication.dart';
import 'package:saiwa_app/repository/prefs.dart';

class CorrectiveRepository {
  static Future<http.Response> applyCorrective(Corrective corrective) async {
    try {
      http.Response authResponse = await Authentication.autoSignIn();
      await Prefs.saveTokens(authResponse.body);
      http.Response response = await http.post(
Uri.parse('https://services.saiwa.ai/api/ServiceRun/ApplyCorrective'),
        headers: {
          'Content-Type': 'application/json',
          'Accept': 'application/json',
          'Authorization': Prefs.accessToken,
        },
        body: json.encode({
          "fileDirGuid": null,
          "extraFileIds": corrective.extraFileIds,
          "extraFileDirGuid": null,
          "unchangedExtraFileIds": [],
          "serviceRunParams": [
            {"parameterId": 65, "value": "2"},
            {"parameterId": 66, "value": 10}
          ],
```

```
          "referenceType": 2,
          "exerciseTypeId": null,
          "subtypeId": null,
          "actionId": null,
          "testType": 1,
          "fileIds": corrective.fileIds,
          "unchangedFileIds": []
        }),
      );

      return response;
    } catch (e) {
      print(e);
    }
    return http.Response("", 404);
  }
}
```

۵.۲.۴. فایل database_repository

```
import 'package:path_provider/path_provider.dart';
import 'package:saiwa_app/models/exercise.dart';
import 'package:saiwa_app/models/frame.dart';
import 'package:saiwa_app/models/training.dart';
import 'package:sqflite/sqflite.dart';
import 'package:sqflite/sql.dart';

class DatabaseRepository {
  static Database? _database;
  Future<Database> get database async {
    if (_database != null) return _database!;
    _database = await initializeDB();
    return _database!;
  }

  static initializeDB() async {
    String path = await getDatabasesPath();
    _database = await openDatabase(
      join(path, 'database.db'),
      onCreate: (database, version) async {
        print(path);
        await database.execute(
          "CREATE TABLE exercise (id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, name
Text );",
        );
        await database.execute(
          "CREATE TABLE training ( id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
date TEXT, reference_id INTEGER REFERENCES exercise (id));",
        );
        await database.execute(
          "CREATE TABLE frame ( guid TEXT,     fileName TEXT, mimeType TEXT,
fileSize INTEGER, isUsed INTEGER, dlFiles TEXT, serviceRunFiles TEXT, id INTEGER
PRIMARY KEY, training_id INTEGER REFERENCES training (id), exercise_id INTEGER
REFERENCES exercise (id), extension TEXT );",
        );
      },
      version: 1,
    );
  }

  static Future<Exercise> insertExercise(Exercise exercise) async {
    await _database!.insert('exercise', exercise.toMap());
    List<Map> result = await _database!.query("exercise");
    print(result);
```

```
    exercise.id = result[result.length - 1]['id'];
    return exercise;
  }

  static Future<Training> insertTraining(Training training) async {
    await _database!.insert('training', training.toMap());
    List<Map> result = await _database!.query("training");
    print(result);
    training.id = result[result.length - 1]['id'];
    return training;
  }

  static insertFrame(Frame frame) async {
    await _database!.insert('frame', frame.toMap());
  }

  static getTrainings(Exercise exercise) async {
    List<Map> result = await _database!
        .query("training", where: "reference_id=${exercise.id}");
    List<Training> trainings = result
        .map((e) => Training.fromTraining(
            exercise: exercise, id: e['id'], dataTimeTimeStamp: e['date']))
        .toList();
    return trainings;
  }

  static getExercises() async {
    List<Map> result = await _database!.query("exercise");
    List<Exercise> exercises =
        result.map((e) => Exercise(name: e['name'], id: e['id'])).toList();
    return exercises;
  }

  static getReferenceImagesIDs(Exercise exercise) async {
    List<Map> result = await _database!
        .query("frame", where: "exercise_id=${exercise.id}", columns: ['id']);
    List<int> ids = result.map((e) => e['id'] as int).toList();
    print(ids);
    return ids;
  }

  static getTrainingImagesIDs(Training training) async {
    List<Map> result = await _database!
        .query("frame", where: "training_id=${training.id}", columns: ['id']);
    List<int> ids = result.map((e) => e['id'] as int).toList();
    print(ids);
    return ids;
  }

  static removeExercise(Exercise exercise) async {
    List<int> trainingIds = await getTrainingIdsByExercise(exercise);

    for (int trainingId in trainingIds) {
      print(trainingId);
      await _database!.delete('frame', where: "training_id=${trainingId}");
    }
    int result =
        await _database!.delete('exercise', where: "id=${exercise.id}");
    int result1 = await _database!
        .delete('training', where: "reference_id=${exercise.id}");
    await _database!.delete('frame', where: "exercise_id=${exercise.id}");
  }

  static removeTraining(Training training) async {
```

```
    await _database!.delete('training', where: "id=${training.id}");
  }

  static getTrainingIdsByExercise(Exercise exercise) async {
    List<Map> result = await _database!
        .query("training", where: "reference_id=${exercise.id}");
    List<int> ids = result.map((e) => e['id'] as int).toList();
    print(ids);
    return ids;
  }

  static checkExerciseExistance(Exercise exercise) async {
    List<Map> exerciseResult =
        await _database!.query("frame", where: "exercise_id=${exercise.id}");
    print(exerciseResult.isNotEmpty);
    return exerciseResult.isNotEmpty;
  }

  static checkTrainingExistance(Training training) async {
    List<Map> trainingResult =
        await _database!.query("frame", where: "training_id=${training.id}");
    print(trainingResult.isNotEmpty);
    return trainingResult.isNotEmpty;
  }

  static resetTables() {
    _database?.delete('training');
    _database?.delete('exercise');
    _database?.delete('frame');
  }

  static Future<List<Map<String, dynamic>>> getFramesJson(
      Exercise exercise) async {
    List<Map<String, dynamic>> trainingResult =
        await _database!.query("frame", where: "exercise_id=${exercise.id}");
    return trainingResult;
  }
}
```

۵.۲.۵. فایل file_repo

```
import 'package:dio/dio.dart';
import 'package:path/path.dart';
import 'package:async/async.dart';
import 'dart:io';
import 'package:http/http.dart' as http;
import 'package:saiwa_app/models/extraction_configue.dart';
import 'dart:convert';

import 'package:saiwa_app/models/training.dart';
import 'package:saiwa_app/repository/authentication.dart';
import 'package:saiwa_app/repository/prefs.dart';
import 'package:saiwa_app/repository/storage_repository.dart';

class FileRepository {
  static Future<http.StreamedResponse> uploadFile(File imageFile) async {
    var stream = new http.ByteStream(imageFile.openRead());
    stream.cast();
    var length = await imageFile.length();
    var uri = Uri.parse("https://services.saiwa.ai/api/file/uploadFile");
    var request = new http.MultipartRequest("POST", uri);
    var multipartFile = new http.MultipartFile('file', stream, length,
        filename: basename(imageFile.path));
```

```
    request.files.add(multipartFile);
    var streamedResponse = await request.send
    return streamedResponse;
  }

  static Future<Response> upload(File imageFile) async {
    Dio dio = Dio();
    var formData = FormData.fromMap({
      'file': await MultipartFile.fromFile(
        imageFile.path,
        filename: imageFile.path.split('/').last,
      ),
    });
    var response = await dio.post(
      'https://services.saiwa.ai/api/file/uploadFile',
      data: formData,
      onSendProgress: (int sent, int total) {
      },
    );
    return response;
  }

  static downloadFile(
      String guid, String fileName, int fileType, Training training) async {
    String savePath =
        await StorageRepository.getCorrectiveDirectoryPath(fileType, training);
    savePath = savePath + fileName;
    print(savePath);
    Dio dio = Dio();
    Response response = await dio.download(
        "https://services.saiwa.ai/static-files/file/temp/$guid/$fileName",
        savePath);
    print(response);
  }

  static downloadFileAndRename(String guid, String fileName, int newFileName,
      int fileType, Training training) async {
    String savePath =
        await StorageRepository.getCorrectiveDirectoryPath(fileType, training);
    savePath =
        savePath + newFileName.toString() + '.' + fileName.split(".").last;
    print(savePath);
    Dio dio = Dio();
    Response response = await dio.download(
        "https://services.saiwa.ai/static-files/file/temp/$guid/$fileName",
        savePath);
  }

  static Future<http.Response> extractFrames(
      ExtractConfigue extractConfigue) async {
    try {
      http.Response authResponse = await Authentication.autoSignIn();
      await Prefs.saveTokens(authResponse.body);
      http.Response response = await http.post(
        Uri.parse('https://services.saiwa.ai/api/Video/ExtractFrames'),
        headers: {
          'Content-Type': 'application/json',
          'Authorization': Prefs.accessToken,
        },
        body: json.encode({
          "outputType": "png",
          "fps": extractConfigue.fps,
          "visibilityThreshold": extractConfigue.visibilityThreshold,
          "exerciseTypeId": null,
```

```
          "videoGuid": extractConfigue.videoGuid,
          "isPermanent": false,
          "references": extractConfigue.referenceList,
        }),
      );
      return response;
    } catch (e) {
      print(e);
      return http.Response("", 500);
    }
  }

  static Future<String> uploadAndGetVideoGuid(File videoFile) async {
    Response response = await upload(videoFile);
    String guid = response.data['guid'];
    return guid;
  }

  static List<dynamic> getOutputMetaFiles(http.Response response) {
    List<dynamic> testFiles = [];
    try {
      testFiles = jsonDecode(response.body)['testFiles'];
    } catch (e) {
      print(e);
    }

    return testFiles;
  }

  static Future downloadOutputFrame(
      dynamic outputMetaData, Training training, int fileIndex) async {
    String guid = outputMetaData['guid'];
    String fileName = outputMetaData['fileName'];
    await downloadFileAndRename(guid, fileName, fileIndex, 0, training);
  }
}
```

۵.۲.۶. فایل firebase_storage_repository

```
import 'dart:io';

import 'package:firebase_storage/firebase_storage.dart';

class FirebaseStorageRepository {
  static UploadTask? uploadUserData(File file, String email) {
    try {
      Reference reference = FirebaseStorage.instance.ref("$email.zip");
      return reference.putFile(file);
    } on Exception catch (e) {
      print(e);
      return null;
    }
  }

  static Future<bool> checkFileExistance(String email) async {
    Reference reference = FirebaseStorage.instance.ref();
    try {
      var file = await reference.child("$email.zip").getDownloadURL();
      return true;
    } on Exception catch (e) {
      return false;
    }
  }
```

```
  static DownloadTask? downloadUserData(File file, String email) {
    try {
      Reference reference = FirebaseStorage.instance.ref("$email.zip");
      return reference.writeToFile(file);
    } on Exception catch (e) {
      print(e);
      return null;
    }
  }
}
```

۵.۲.۷. فایل storage_repository

```
import 'dart:io';

import 'package:archive/archive_io.dart';
import 'package:flutter/services.dart';
import 'package:path_provider/path_provider.dart';
import 'package:saiwa_app/core/injectable/injectable.dart';
import 'package:saiwa_app/models/exercise.dart';
import 'package:saiwa_app/models/frame_data.dart';
import 'package:saiwa_app/models/training.dart';

class StorageRepository {
  static initiateDirectories(Directory directory) async {
    await Directory(directory.path + "original/").create();
    await Directory(directory.path + "annotation/").create();
    await Directory(directory.path + "skeleton/").create();
    await Directory(directory.path + "errors/").create();
  }

  static clearReferences(Exercise exercise) async {
    final Directory documentsDirectory =
        await getApplicationDocumentsDirectory();

    Directory directory =
        Directory("${documentsDirectory.path}/${exercise.id}/reference/");
    if (await directory.exists()) {
      await directory.delete(recursive: true);
      await directory.create(recursive: true);
    }
  }

  static copyPresetReferences(Exercise exercise, String preset) async {
    final Directory documentsDirectory =
        await getApplicationDocumentsDirectory();
    Directory directory =
        Directory("${documentsDirectory.path}/${exercise.id}/");
    if (!await directory.exists()) {
      await directory.create();
    }
    await Directory(directory.path + "trainings/").create();
    directory =
        Directory("${documentsDirectory.path}/${exercise.id}/reference/");
    if (!await directory.exists()) {
      await directory.create();
    }

    for (int i = 1; i <= 6; i++) {
      ByteData bytes =
          await rootBundle.load('assets/references/$preset/$i.png');

      File file = File('${directory.path}/$i.png');
```

```
      await file.writeAsBytes(
          bytes.buffer.asUint8List(bytes.offsetInBytes, bytes.lengthInBytes));
    }
  }

  static writeReference(String imagePath, Exercise exercise) async {
    final Directory documentsDirectory =
        await getApplicationDocumentsDirectory();
    Directory directory =
        Directory("${documentsDirectory.path}/${exercise.id}/");

    if (!await directory.exists()) {
      await directory.create();
    }
    await Directory(directory.path + "trainings/").create();
    directory =
        Directory("${documentsDirectory.path}/${exercise.id}/reference/");
    if (!await directory.exists()) {
      await directory.create();
    }

    var directoryList = await directory.list(recursive: false).toList();
    int sizeName = directoryList.length + 1;
    final File newImage =
        await File(imagePath).copy("${directory.path}/$sizeName.png");
    await File(imagePath).delete();
  }

  static Future<List<File>> getAllImagesOfReference(Exercise exercise) async {
    final Directory documentsDirectory = getIt<Directory>();
    final Directory directory =
        Directory("${documentsDirectory.path}/${exercise.id}/reference/");

    if (!await directory.exists()) {
      return [];
    }
    var directoryList = await directory.list(recursive: false).toList();
    List<String> filesPath = directoryList.map((e) => e.path).toList();
    filesPath.sort((a, b) => int.parse(a.split('/').last.split('.').first)
        .compareTo(int.parse(b.split('/').last.split('.').first)));

    return filesPath.map((e) => File(e)).toList();
  }

  static Future<File> writeTraining(
      String imagePath, Training training, imageIndex) async {
    final Directory documentsDirectory = getIt<Directory>();
    Directory directory = Directory(

"${documentsDirectory.path}/${training.exercise.id}/trainings/${training.id}/");
    if (!await directory.exists()) {
      await directory.create();
      await initiateDirectories(directory);
    }
    directory = Directory(

"${documentsDirectory.path}/${training.exercise.id}/trainings/${training.id}/origin
al/");
    final File newImage =
        await File(imagePath).copy("${directory.path}$imageIndex.png");
    File(imagePath).delete();
    return newImage;
  }
```

```
  static initiateTrainingDirectoryFromVideo(Training training) async {
    final Directory documentsDirectory = getIt<Directory>();
    Directory directory = Directory(

"${documentsDirectory.path}/${training.exercise.id}/trainings/${training.id}/");
    if (!await directory.exists()) {
      await directory.create();
      await initiateDirectories(directory);
    }
  }

  static String getCorrectiveDirectoryPath(int fileType, Training training) {
    final Directory documentsDirectory = getIt<Directory>();

    String path =

"${documentsDirectory.path}/${training.exercise.id}/trainings/${training.id}/";

    if (fileType == 100) {
      path = path + "annotation/";
    } else if (fileType == 117) {
      path = path + "skeleton/";
    } else if (fileType == 200) {
      path = path + "errors/";
    } else if (fileType == 0) {
      path = path + "original/";
    }
    return path;
  }

  static removeExercise(Exercise exercise) async {
    final Directory documentsDirectory = getIt<Directory>();
    Directory directory =
        Directory("${documentsDirectory.path}/${exercise.id}/");
    await directory.delete(recursive: true);
  }

  static void removeTraining(Training training) {
    final Directory documentsDirectory = getIt<Directory>();
    Directory directory = Directory(

"${documentsDirectory.path}/${training.exercise.id}/reference/${training.id}/");
    directory.delete();
  }

  static FrameData getFrameDataOf(Training training, int index) {
    final Directory documentsDirectory = getIt<Directory>();
    String path =

"${documentsDirectory.path}/${training.exercise.id}/trainings/${training.id}/";
    String originalImagePath = '$path/original/$index.png';
    String annotatedImagePath = '$path/annotation/$index.png';
    String errorFilePath = '$path/errors/$index.txt';
    String skeletonImagePath = '$path/skeleton/$index.png';
    String referenceImagePath =
        '${documentsDirectory.path}/${training.exercise.id}/reference/$index.png';
    FrameData frameData = FrameData(
      referenceImage: File(referenceImagePath),
      originalImage: File(originalImagePath),
      skeltonImage: File(skeletonImagePath),
      annotatedImage: File(annotatedImagePath),
      errorTextFile: File(errorFilePath),
    );
    return frameData;
```

```
  }

  static Future<int> countFramesOf(Training training) async {
    final Directory documentsDirectory = getIt<Directory>();
    Directory directory = Directory(

"${documentsDirectory.path}/${training.exercise.id}/trainings/${training.id}/origin
al");
    var directoryList = await directory.list(recursive: false).toList();
    int result = directoryList.length;
    return result;
  }

  static Future<List<List<int>>> getTrainingsErrors(
      Exercise exercise, List<Training> trainings) async {
    final Directory documentsDirectory = getIt<Directory>();
    int numberOfFrames = await countFramesOf(trainings[0]);
    String directoryPath;
    List<List<int>> errors = [];
    for (int i = 1; i <= numberOfFrames; i++) {
      List<int> frameErrors = [];
      for (var training in trainings) {
        directoryPath =

"${documentsDirectory.path}/${exercise.id}/trainings/${training.id}/errors/$i.txt";
        File errorFile = File(directoryPath);
        try {
          final String contents = await errorFile.readAsString();
          final String extractedInt = contents.split(' ').last;
          frameErrors.add(int.parse(extractedInt));
        } catch (e) {
          print(e);
        }
      }
      errors.add(frameErrors);
    }
    print(errors);
    return errors;
  }

  static Future<List<int>> getTrainingsTotalErrors(
      Exercise exercise, List<Training> trainings) async {
    final Directory documentsDirectory = getIt<Directory>();
    int numberOfFrames = await countFramesOf(trainings[0]);
    String directoryPath;
    List<int> errors = [];
    for (var training in trainings) {
      int trainingErrors = 0;
      for (int i = 1; i <= numberOfFrames; i++) {
        directoryPath =
"${documentsDirectory.path}/${exercise.id}/trainings/${training.id}/errors/$i.txt";
        File errorFile = File(directoryPath);
        try {
          final String contents = await errorFile.readAsString();
          final String extractedInt = contents.split(' ').last;
          trainingErrors += int.parse(extractedInt);
        } catch (e) {
          print(e);
        }
      }
      errors.add(trainingErrors);
    }
    print(errors);
    return errors;
```

```
  }

  static Future<File> getLocalUserData(String email) async {
    final Directory documentsDirectory =
        await getApplicationDocumentsDirectory();
    var directoryList =
        await documentsDirectory.list(recursive: false).toList();
    List<File> files = directoryList.map((e) => File(e.path)).toList();
    var encoder = ZipFileEncoder();
    encoder.create('${documentsDirectory.path}/$email.zip');
    for (var file in files) {
      try {
        await encoder.addFile(file);
      } catch (_) {
        await encoder.addDirectory(Directory(file.path));
      }
    }
    encoder.close();
    return File("${documentsDirectory.path}/$email.zip");
  }

  static clearUserData() async {
    final Directory documentsDirectory =
        await getApplicationDocumentsDirectory();
    var directoryList =
        await documentsDirectory.list(recursive: false).toList();
    List<File> files = directoryList.map((e) => File(e.path)).toList();
    for (var file in files) {
      if (!file.path.contains("database.db")) {
        file.deleteSync(recursive: true);
      }
    }
  }

  static initiateZipDirectory() async {
    final Directory documentsDirectory =
        await getApplicationDocumentsDirectory();
    return File("${documentsDirectory.path}/zipedBackup.zip");
  }

  static extractBackup() async {
    final Directory documentsDirectory =
        await getApplicationDocumentsDirectory();
    final bytes =
        await File('${documentsDirectory.path}/zipedBackup.zip').readAsBytes();
    final archive = ZipDecoder().decodeBytes(bytes);
    for (final file in archive) {
      final filename = file.name;
      if (file.isFile) {
        final data = file.content as List<int>;
        File newFile = File('${documentsDirectory.path}/' + filename);
        await newFile.create(recursive: true);
        await newFile.writeAsBytes(data);
      } else {
        Directory('${documentsDirectory.path}/' + filename)
            .create(recursive: true);
      }
    }
    await File('${documentsDirectory.path}/zipedBackup.zip').delete();
  }
}
```

۶. منابع و مراجع

[1] - flutter.dev

[2] - pub.dev

[3] - bloclibrary.dev

[4] - stackoverflow.com

[5] - github.com

[6] - developers.google.com

[7] - dart.dev

[8] - firebase.google.com

[9] - firebase.flutter.dev

[10] - mobindustry.net

[11] - gradle.org

[12] - developer.apple.com

[13] - dev.saiwa.ai

[14] - doc.saiwa.ai

[15] - https://www.theverge.com/2020/2/26/21154185/tempo-smart-home-gym-kinect-computer-vision-ai-form-correction

[16] - https://venturebeat.com/2020/05/08/exer-launches-ai-based-fitness-coach-on-your-mobile-phone/

[17] - https://ieeexplore.ieee.org/document/8856547/authors#authors

[18] - https://insider.fitt.co/onyx-ai-fitness-startup/

[19] - https://towardsdatascience.com/just-used-machine-learning-in-my-workout-ff079b8e1939