
بسم الله الرحمن الرحيم

دانشگاه صنعتی اصفهان
دانشکده برق و کامپیوتر

گزارش کارآموزی

نام و نام خانوادگی کارآموز: پرنیان ملک‌زاده
شماره دانشجویی: ۴۰۰۱۷۰۵۳
استاد کارآموزی: دکتر مجتبی خلیلی
سرپرست کارآموزی: محمود باقری

محل کارآموزی: شرکت مهندسی حضور و غیاب کسرا (کیمیاگران سرزمین رایانه)
آدرس: خیابان سهروردی، خیابان فرهنگ، کوچه یکم، ساختمان کسرا

تلفن: ۳۴۱۰۴-۰۳۱

تاریخ شروع: ۱۴۰۳/۰۵/۰۱ **تاریخ پایان:** ۱۴۰۳/۰۶/۲۵

فرم شماره ۱
گزارش خلاصه ای از فعالیتهای هفتگی

نام و نام خانوادگی کارآموز: پرنیان ملک‌زاده **شماره دانشجویی: ۴۰۰۱۷۰۵۳**

رشته تحصیلی: مهندسی کامپیوتر **دانشکده: برق و کامپیوتر**

هفته اول **آشنایی با محیط کار، تیم و ابزارها** **از تاریخ: ۰۵/۰۱** **لغایت تاریخ: ۰۵/۰۷**	**در هفته اول، تمرکز من بر آشنایی با محیط کاری و ابزارهای مورد استفاده در تیم توسعه بود. در ابتدای کار، Android Studio را نصب و تنظیمات لازم برای شروع به کار را انجام دادم و ابزارهایی مانند Git برای مدیریت نسخه‌ها و Postman برای تست APIها را نصب کردم. همچنین با همکاران تیم آشنا شدم و ساختار کلی پروژه، معماری آن، و نحوه تعامل اجزای مختلف اپلیکیشن را فرا گرفتم. یادگیری این ابزارها به من کمک کرد تا به‌سرعت با فرآیند کاری تیم هماهنگ شوم و برای مراحل بعدی آماده باشم.**
هفته دوم **مطالعه و بررسی کدهای پروژه موجود** **از تاریخ: ۰۵/۰۸** **لغایت تاریخ: ۰۵/۱۴**	**در هفته دوم، زمان خود را صرف مطالعه کدهای موجود و درک معماری پروژه کردم. با معماری MVVM که به‌طور گسترده در پروژه استفاده می‌شد، آشنا شدم و با فریم‌ورک‌ها و کتابخانه‌های مهم مانند Retrofit برای ارتباط با سرور، Room برای مدیریت دیتابیس محلی، و Glide برای بارگذاری تصاویر کار کردم. همچنین با بررسی کدهای گذشته و مطالعه مستندات داخلی پروژه، درک عمیق‌تری از الگوهای کدنویسی و ساختار پروژه به دست آوردم.**
هفته سوم **یادگیری کاتلین و انجام تسک‌های مقدماتی** **از تاریخ: ۰۵/۱۵** **لغایت تاریخ: ۰۵/۲۱**	**در هفته سوم، من بر روی یادگیری زبان کاتلین به‌عنوان زبان اصلی برنامه‌نویسی اپلیکیشن‌های اندروید تمرکز کردم. با شرکت در دوره‌های آموزشی داخلی و مطالعه مستندات رسمی، با مبانی کاتلین مانند تعریف متغیرها، ساختارهای کنترلی، و ویژگی‌هایی مانند Null Safety و Type Inference آشنا شدم. سپس به مباحث پیشرفته‌تر مانند توابع Extension، کار با Lambdaها، و Higher-Order Functions پرداختم که به بهبود و تمیزتر شدن کدنویسی کمک می‌کنند. همچنین Coroutines را برای مدیریت عملیات ناهمگام یاد گرفتم که به بهبود عملکرد برنامه در هنگام انجام کارهای سنگین کمک می‌کند. در کنار یادگیری مباحث تئوری، به انجام تسک‌های ساده و مقدماتی مانند اصلاح باگ‌های کوچک در رابط کاربری و پیاده‌سازی تغییرات جزئی در کد پرداختم.**
هفته چهارم **توسعه ویژگی‌های جدید (Feature Development)** **از تاریخ: ۰۵/۲۲** **لغایت تاریخ: ۰۵/۲۸**	**در هفته چهارم، مسئولیت توسعه یک ویژگی جدید به من محول شد. این تسک شامل ایجاد یک صفحه جدید در اپلیکیشن، اتصال آن به APIهای موجود و نمایش داده‌ها در رابط کاربری بود. از Retrofit برای ارتباط با APIها و از ViewModel و LiveData برای مدیریت داده‌ها استفاده کردم. این تجربه به من کمک کرد تا در کار با معماری MVVM مهارت بیشتری کسب کنم و اولین کد خود را برای بررسی تیمی (Code Review) ارسال کردم که بازخوردهای مفیدی از سایر اعضای تیم دریافت کردم.**

نام و نام خانوادگی سرپرست کارآموزی
امضاء سرپرست

فرم شماره ۱ (ادامه)
گزارش خلاصه ای از فعالیتهای هفتگی

نام و نام خانوادگی کارآموز: پرنیان ملک‌زاده **شماره دانشجویی: ۴۰۰۱۷۰۵۳**

رشته تحصیلی: مهندسی کامپیوتر **دانشکده: برق و کامپیوتر**

هفته پنجم **طراحی UI/UX با Material Design** **از تاریخ: ۰۵/۲۹** **لغایت تاریخ: ۰۶/۰۴**	**در هفته پنجم، تمرکز من بر طراحی و پیاده‌سازی رابط کاربری با استفاده از اصول Material Design بود. با ابزار Figma نمونه‌های اولیه رابط کاربری را طراحی کردم و سپس آن‌ها را با استفاده از ConstraintLayout و سایر عناصر UI پیاده‌سازی نمودم. همچنین با استفاده از Navigation Component، مدیریت ناوبری بین صفحات را بهبود بخشیدم و از SafeArgs برای ارسال داده‌ها بین Fragmentها استفاده کردم. این فعالیت‌ها به من کمک کرد تا اصول طراحی اندروید را به‌صورت عملی یاد بگیرم و تجربه خوبی در ایجاد UIهای جذاب و کاربرپسند کسب کنم.**
هفته ششم **تست و رفع اشکال با استفاده از ابزارهای تست اندروید** **از تاریخ: ۰۶/۰۵** **لغایت تاریخ: ۰۶/۱۱**	**در هفته ششم، به مرحله تست و رفع اشکالات کد رسیدم. ابتدا با استفاده از JUnit و Mockito، تست‌های واحد (Unit Tests) را برای بخش‌های مختلف برنامه پیاده‌سازی کردم. سپس با استفاده از Espresso، تست‌های رابط کاربری (UI Tests) را انجام دادم تا مطمئن شوم تمامی عناصر UI به درستی کار می‌کنند. در این مرحله، با استفاده از ابزارهای Android Profiler و LeakCanary، به بهینه‌سازی عملکرد اپلیکیشن و شناسایی و رفع مشکلات مصرف حافظه پرداختم. این تجربه به من کمک کرد تا مهارت‌های تست و بهینه‌سازی خود را بهبود بخشم و کیفیت نرم‌افزار را افزایش دهم.**
هفته هفتم **بهینه‌سازی کد و بهبود عملکرد اپلیکیشن** **از تاریخ: ۰۶/۱۲** **لغایت تاریخ: ۰۶/۱۸**	**در هفته هفتم، هدف اصلی من بهینه‌سازی کد و بهبود عملکرد اپلیکیشن بود. بهینه‌سازی‌هایی در بخش مدیریت حافظه و کاهش زمان بارگذاری صفحات انجام دادم که شامل بهبود در Queryهای دیتابیس با استفاده از Room و کاهش مصرف حافظه با بهینه‌سازی تصاویر بود. همچنین با ابزارهای Profiler، نقاط ضعف در کد مانند گلوگاه‌های CPU و I/O را شناسایی و رفع کردم. این هفته به من کمک کرد تا به درک عمیق‌تری از بهینه‌سازی کد و افزایش کارایی نرم‌افزار برسم.**
هفته هشتم **مستندسازی، ارائه نهایی و تجربه یادگیری جدید** **از تاریخ: ۰۶/۱۹** **لغایت تاریخ: ۰۶/۲۵**	**در هفته هشتم، علاوه بر مستندسازی کدها و نوشتن راهنمای استفاده برای سایر توسعه‌دهندگان، تجربه یادگیری جدیدی نیز کسب کردم. فرصتی پیش آمد تا در یک جلسه آموزشی داخلی شرکت کنم و با ابزارهای مانیتورینگ و آنالیز اپلیکیشن مانند Firebase Analytics و Crashlytics آشنا شوم. این تجربه به من نشان داد که چگونه می‌توان از داده‌های کاربران برای بهبود نرم‌افزار استفاده کرد. همچنین این هفته برای من فرصتی بود تا تمامی یادگیری‌ها و دستاوردهای خود را مرور و برای ادامه مسیر حرفه‌ای خود برنامه‌ریزی کنم.**

نام و نام خانوادگی سرپرست کارآموزی
امضاء سرپرست

فرم شماره ۲
گزارش سرپرست کارآموزی

نام و نام خانوادگی سرپرست کارآموزی: **سمت:**

نام واحد صنعتی:

نام و نام خانوادگی کارآموز: **شماره دانشجویی:**

گرایش تحصیلی: **دانشکده: برق و کامپیوتر**

ردیف	اظهار نظر سرپرست کارآموز	ضعیف	متوسط	خوب	عالی
۱	رعایت نظم و ترتیب و انضباط در محیط کار				
۲	میزان علاقه و همکاری با دیگران				
۳	علاقه به فراگیری مطالب علمی و فنی				
۴	پیگیری وظایف و میزان پشتکار				
۵	ارزش پیشنهادات کارآموز در جهت بهبود کار				
۶	کیفیت گزارشهای کارآموزی (حداقل فرمهای شماره ۱)				
۷	میزان بهره گیری از امکانات موجود جهت ارتقاء توانایی علمی و فنی				

تعداد روزهای مرخصی: **تعداد روزهای غیبت:**

پیشنهادات سرپرست کارآموزی جهت بهبود دورهٔ کارآموزی:

امضاء سرپرست کارآموزی

به نام خدا

فرم شماره ۲
گزارش سرپرست کارآموزی

نام و نام خانوادگی سرپرست کارآموزی: محمود نامدی سمت: مدیر فنی (CTO)

نام واحد صنعتی: شرکت فنی مهندسی لسرا (لیزرگران سرزمین رایانه)

نام و نام خانوادگی کارآموز: پریسان ملک زاده شماره دانشجویی: ۴۰۰۱۷۰۵۳

گرایش تحصیلی: رایانش امن دانشکده مهندسی برق و کامپیوتر

ردیف	اظهار نظر سرپرست کارآموز	ضعیف	متوسط	خوب	عالی
۱	رعایت نظم و ترتیب و انضباط در محیط کار				✓
۲	میزان علاقه و همکاری با دیگران			✓	
۳	علاقه به فراگیری مطالب علمی و فنی				✓
۴	پیگیری وظایف و میزان پشتکار				✓
۵	ارزش پیشنهادات کارآموز در جهت بهبود کار				✓
۶	کیفیت گزارشهای کارآموزی (حداقل فرمهای شماره ۱)			✓	
۷	میزان بهره گیری از امکانات موجود جهت ارتقاء توانایی علمی و فنی				✓

تعداد روزهای مرخصی: ۵ روز تعداد روزهای غیبت: ۵ روز

پیشنهادات سرپرست کارآموزی جهت بهبود دورهٔ کارآموزی:

تاریخ ۱۴۰۳/۰۶/۲۴

امضاء سرپرست کارآموزی

فرم شماره ۳
نظرات و پیشنهادات
(در پایان دوره تکمیل شود)

نام و نام خانوادگی کارآموز: پرنیان ملک‌زاده **شماره دانشجویی: ۴۰۰۱۷۰۵۳**

رشته تحصیلی: مهندسی کامپیوتر **دانشکده: برق و کامپیوتر**

شرح نظرات و پیشنهادات:

۱- در مورد دورهٔ کارآموزی و مراحل مختلف آن: **دوره کارآموزی در شرکت مهندسی کسرا تجربه‌ای بسیار ارزشمند و مفید بود. مراحل مختلف کارآموزی به‌خوبی برنامه‌ریزی شده و شامل یادگیری و استفاده از ابزارهای توسعه اندروید، آشنایی با معماری نرم‌افزار و پیاده‌سازی عملی پروژه‌های واقعی بود. این دوره با ارائه پروژه‌های متنوع، به من کمک کرد تا مهارت‌های عملی و فنی خود را در حوزه توسعه اپلیکیشن‌های اندروید بهبود بخشم و به شکل عملی با چالش‌های واقعی در توسعه نرم‌افزار روبه‌رو شوم.**
۲- در مورد امور پژوهشی، فنی و تولیدی محل کارآموزی: **شرکت مهندسی کسرا با رویکردی نوآورانه به تحقیق و توسعه، فضایی مناسب برای یادگیری و رشد فراهم می‌کند. بخش‌های فنی و تولیدی شرکت به‌خوبی سازمان‌دهی شده‌اند و از ابزارها و تکنولوژی‌های به‌روز در زمینه توسعه نرم‌افزار استفاده می‌شود. پیشنهاد می‌شود که در آینده کارآموزان فرصت بیشتری برای مشارکت در پروژه‌های پیچیده‌تر و انجام پژوهش‌های مرتبط با فناوری‌های نوین در حوزه توسعه نرم‌افزار داشته باشند تا تجربه علمی و عملی آن‌ها غنی‌تر شود.**

توجه: علاوه بر ارائه فرم نظرات و پیشنهادات در پیوست گزارش تفصیلی، در صورت تمایل یک کپی از این فرم را به دفتر ارتباط با صنعت دانشکده تحویل نمایید.

امضاء کارآموز

فهرست مطالب

چکیده

در این گزارش، به شرح فعالیت‌ها و تجربه‌های کسب‌شده در دوره کارآموزی خود در شرکت مهندسی حضور و غیاب کسرا در حوزه توسعه اپلیکیشن‌های اندروید پرداخته‌ام. در ابتدا، شرکت کسرا و زمینه‌های فعالیت آن را معرفی کرده‌ام. سپس، مراحل مختلف کارآموزی، از جمله آشنایی با محیط کاری و ابزارهای توسعه، مطالعه کدهای موجود و معماری MVVM را توضیح داده‌ام. همچنین با استفاده از کتابخانه‌هایی مانند Retrofit و Room، عملیات ارتباط با سرور و مدیریت دیتابیس را بررسی کرده‌ام.

در طول دوره، تمرکز من بر یادگیری زبان برنامه‌نویسی کاتلین و پیاده‌سازی تسک‌های مختلف، از جمله توسعه ویژگی‌های جدید و بهینه‌سازی عملکرد بود. علاوه بر این، با اصول طراحی UI/UX و پیاده‌سازی آن‌ها بر اساس Material Design آشنا شدم. همچنین تست‌های واحد و رابط کاربری را برای افزایش کیفیت کد و بهبود عملکرد اپلیکیشن انجام داده‌ام.

در نهایت، با ارائه یک موضوع خاص درباره استفاده از Coroutines برای مدیریت عملیات ناهمگام در کاتلین و مستندسازی کامل پروژه، فعالیت‌ها و دستاوردهای دوره کارآموزی خود را جمع‌بندی کرده‌ام.

فصل اول

معرفی شرکت مهندسی کسرا

۱-۱- مقدمه

شرکت مهندسی کسرا (کیمیاگران سرزمین رایانه) در سال ۱۳۷۹ تأسیس شد و با بیش از ۲۳ سال سابقه در حوزه فناوری اطلاعات، به ارائه راهکارهای نرم‌افزاری و سخت‌افزاری مدیریت کنترل تردد و منابع انسانی می‌پردازد. کسرا بر این باور است که سازمان ها و شرکت های عضو خانواده خود تعیین کنندگان اصلی خط و مشی آینده شرکت هستند و بدین منظور تلاش می کند تا با شناسایی و درک نیازهای آنان، و با به کارگیری فناوری‌های روز و ارائه راهکارهای نوین بتواند درکمترین زمان و به بهترین شکل پاسخگوی خواست کسب و کارهای بزرگ ایرانی باشد. این شرکت با بیش از ۱۰۰۰ مشتری بزرگ، از جمله سازمان‌های داخلی و بین‌المللی، توانسته است راهکارهای نوآورانه‌ای در حوزه حضور و غیاب، مدیریت تغذیه و کنترل دسترسی ارائه دهد. کسرا همواره بر بهبود بهره‌وری و ارائه خدمات سفارشی به شرکت‌های مختلف تمرکز داشته است.

۱-۲- گروه‌های فعال در شرکت

شرکت کسرا دارای چندین گروه تخصصی در زمینه‌های مختلف فناوری اطلاعات است:

- **گروه توسعه نرم‌افزارهای حضور و غیاب: این گروه به توسعه راهکارهای نرم‌افزاری جهت ثبت تردد کارکنان با استفاده از تکنولوژی‌هایی مانند اثر انگشت و تشخیص چهره می‌پردازد.**
- **گروه مدیریت منابع انسانی: این گروه ابزارهای پیشرفته‌ای برای مدیریت فرآیندهای سازمانی از جمله حقوق و دستمزد و ارزیابی عملکرد کارکنان ارائه می‌دهد.**
- **گروه تحقیق و توسعه[۱]: مسئول نوآوری و بهبود محصولات موجود و ارائه راهکارهای جدید برای رفع نیازهای مشتریان است.**

۱-۳- محصولات و خدمات

شرکت کسرا محصولات متنوعی در حوزه‌های مختلف نرم‌افزاری و سخت‌افزاری ارائه می‌دهد، از جمله:

- **سیستم‌های حضور و غیاب: این سیستم‌ها برای کنترل تردد کارکنان از روش‌های مختلفی مانند اثر انگشت و تشخیص چهره استفاده می‌کنند.**
- **نرم‌افزار مدیریت تغذیه: این سامانه برای مدیریت رستوران‌ها و تغذیه کارکنان طراحی شده است.**

---

[۱] R&D (Research and Development)

- **سامانه حراست و انتظامات: این سیستم برای مدیریت امنیت و کنترل تردد در سازمان‌ها استفاده می‌شود.**

شکل ۱-۱: محصولات نرم‌افزاری شرکت کسرا

۱-۴- پروژه‌های شاخص

کسرا پروژه‌های مهمی را در حوزه مدیریت تردد و کنترل دسترسی انجام داده است. از جمله مشتریان این شرکت می‌توان به وزارت نفت، پتروشیمی زاگرس، ایران خودرو و دانشگاه‌های علوم پزشکی اشاره کرد. سیستم‌های طراحی‌شده توسط کسرا در این شرکت‌ها برای بهبود کنترل حضور و غیاب و مدیریت منابع انسانی استفاده می‌شوند.

۱-۵- موقعیت جغرافیایی

شرکت مهندسی کسرا در تهران و اصفهان مستقر است و دفتر مرکزی آن به‌عنوان پایگاه اصلی مدیریت پروژه‌ها و ارائه خدمات به مشتریان فعالیت می‌کند. این شرکت خدمات خود را از طریق وب‌سایت رسمی خود نیز ارائه می‌دهد.

۱-۶- درآمد و بازار هدف

شرکت کسرا با ارائه محصولات نرم‌افزاری و سخت‌افزاری در زمینه مدیریت تردد و منابع انسانی، درآمد خود را از طریق فروش و پشتیبانی این محصولات به‌دست می‌آورد. بازار هدف این شرکت شامل شرکت‌های دولتی و خصوصی، دانشگاه‌ها و بانک‌ها است که به سیستم‌های پیشرفته کنترل تردد نیاز دارند.

فصل دوم

کارهای انجام شده در دوره کارآموزی

۲-۱- مقدمه

در این فصل، به شرح دقیق علمی، فنی و اجرایی کارهایی که در طول دوره کارآموزی در شرکت مهندسی حضور و غیاب کسرا انجام شده است، پرداخته می‌شود. این دوره با موضوع توسعه اپلیکیشن‌های اندرویدی بوده و تمامی فعالیت‌های مرتبط با فرآیندهای توسعه، پیاده‌سازی، تست و بهینه‌سازی را شامل می‌شود.

۲-۲- آشنایی با ابزارهای توسعه و محیط کاری

اولین مرحله در کارآموزی من، آشنایی با ابزارها و فناوری‌های مورد استفاده در تیم توسعه اندروید بود. انتخاب ابزارهای مناسب در فرآیند توسعه نرم‌افزار اندروید، نقش حیاتی در مدیریت پروژه و بهره‌وری تیم دارد. در این بخش، ابزارهای مختلف و نحوه استفاده از آن‌ها در پروژه به تفصیل بررسی می‌شود.

۲-۲-۱- Android Studio

Android Studio به‌عنوان محیط توسعه[۱] استاندارد برای برنامه‌نویسی اندروید استفاده شد. Android Studio توسط گوگل ارائه شده و با ابزارهای داخلی خود، از جمله پشتیبانی از Gradle برای ساخت پروژه و Android Profiler برای بهینه‌سازی و اشکال‌زدایی، تبدیل به انتخابی محبوب برای توسعه‌دهندگان اندروید شده است.

نصب و راه‌اندازی Android Studio شامل تنظیم SDKهای اندروید و نصب شبیه‌ساز[۲] برای آزمایش اپلیکیشن‌ها روی دستگاه‌های مجازی بود. Android Studio با قابلیت‌های پیشرفته‌ای مانند Instant Run به ما اجازه می‌داد تا تغییرات در کدها به‌سرعت در محیط شبیه‌سازی‌شده منعکس شود.

از جمله مهم‌ترین ویژگی‌های Android Studio می‌توان به IntelliSense برای تکمیل خودکار کد، Refactor برای بهینه‌سازی ساختار کدها و Lint برای شناسایی خطاهای بالقوه اشاره کرد. استفاده از این ابزارها به ما کمک کرد تا از بروز خطاهای رایج جلوگیری کرده و بهره‌وری در توسعه کد را افزایش دهیم.

---

[۱] IDE (Integrated Development Environment)

[۲] Emulator

۲-۲-۲- Git و GitHub

Git به‌عنوان سیستم کنترل نسخه و GitHub به‌عنوان پلتفرم مدیریت پروژه‌های کدنویسی، در تمامی مراحل توسعه پروژه استفاده شد. استفاده از Git به ما این امکان را داد که تغییرات کد را به‌صورت تدریجی ثبت و در صورت بروز مشکل، به نسخه‌های قبلی بازگردیم. فرآیند Branching و Merging، باعث شد تا چندین توسعه‌دهنده به‌طور هم‌زمان روی پروژه کار کنند، بدون آنکه تداخلی در کدها ایجاد شود.

فرآیند Pull Request به‌عنوان یکی از روش‌های مهم در توسعه نرم‌افزار تیمی مورد استفاده قرار گرفت. هر بار که تغییری در کدها انجام می‌دادم، با ارسال یک Pull Request درخواست می‌کردم که کدهای من توسط دیگر اعضای تیم بازبینی و ادغام شود. این فرآیند باعث شد تا بازخوردهای سازنده‌ای از دیگر توسعه‌دهندگان دریافت کرده و از خطاهای احتمالی جلوگیری شود.

Code Review یکی از مهم‌ترین بخش‌های توسعه نرم‌افزار بود. هر تغییری که توسط اعضای تیم انجام می‌شد، توسط سایر اعضا بررسی شده و نظرات آن‌ها اعمال می‌شد. این فرآیند به بهبود کیفیت کد کمک زیادی کرد و مانع از ادغام کدهای مشکل‌دار به پروژه اصلی شد.

۲-۲-۳- Postman

Postman به‌عنوان ابزاری برای تست APIها و بررسی درخواست‌های HTTP در پروژه استفاده شد. Postman به ما امکان داد که بدون نیاز به اجرای اپلیکیشن، ارتباطات شبکه و پاسخ‌های سرور را تست کنیم. با استفاده از Postman، توانستم انواع مختلف درخواست‌های GET، POST، PUT و DELETE را ارسال و پاسخ‌های سرور را به‌صورت JSON یا XML بررسی کنم. این تست‌ها به شناسایی سریع مشکلات ارتباطی و بهبود کارایی APIها کمک بسیاری کرد.

یکی از ویژگی‌های مهم Postman و مزیت استفاده از آن، امکان ذخیره و سازماندهی درخواست‌ها در قالب Collection است. این قابلیت به ما اجازه می‌داد که تست‌های مختلفی را برای هر Endpoint تنظیم کرده و در طول فرآیند توسعه، از صحت عملکرد آن‌ها اطمینان حاصل کنیم.

۲-۳- بررسی ساختار پروژه و معماری نرم‌افزار

در مرحله بعدی کارآموزی، درک معماری پروژه و نحوه تعامل بخش‌های مختلف آن با هم، اهمیت ویژه‌ای داشت. پروژه اندرویدی شرکت از الگوی MVVM (Model-View-ViewModel) استفاده می‌کرد که یکی از محبوب‌ترین معماری‌ها برای توسعه اپلیکیشن‌های اندرویدی مدرن است. این معماری به تفکیک منطقی وظایف کمک کرده و ساختار قابل نگهداری‌تری برای پروژه ایجاد می‌کند.

۲-۳-۱- معماری MVVM

معماری MVVM سه لایه اصلی دارد که در تعامل با یکدیگر به‌طور مجزا کار می‌کنند. این لایه‌ها عبارتند از:

- Model: لایه‌ای است که داده‌ها و منطق کسب‌وکار در آن ذخیره و مدیریت می‌شود. در این پروژه، داده‌های مرتبط با حضور و غیاب کارکنان از طریق APIهای سرور دریافت شده و در این لایه ذخیره می‌شد.
- ViewModel: واسطه‌ای است که داده‌ها را از Model دریافت کرده و آن‌ها را برای نمایش در View آماده می‌کند. از LiveData برای مدیریت داده‌های واکنش‌گرا در ViewModel استفاده شد که به‌محض تغییر داده‌ها، رابط کاربری نیز به‌روزرسانی می‌شد.
- View: لایه رابط کاربری است که داده‌ها را به کاربر نمایش می‌دهد و تعاملات کاربر را به ViewModel ارسال می‌کند.

استفاده از MVVM به ما این امکان را داد که داده‌ها و منطق نمایش آن‌ها را از یکدیگر جدا کنیم و این مسئله باعث شد تا نگهداری و تست کدها بسیار ساده‌تر و مؤثرتر باشد.

۲-۳-۲- بررسی فریم‌ورک‌ها و کتابخانه‌های مورد استفاده

در این پروژه از کتابخانه‌های مختلفی استفاده شد که هر کدام نقش خاصی در بهبود کارایی و تسریع فرآیند توسعه داشتند. در ادامه، برخی از مهم‌ترین این کتابخانه‌ها را بررسی می‌کنیم:

- Retrofit: **برای برقراری ارتباط با سرور و ارسال درخواست‌های HTTP از کتابخانه Retrofit استفاده شد. Retrofit به‌طور خودکار درخواست‌ها را به فرمت JSON تبدیل کرده و پاسخ‌های سرور را به مدل‌های کاتلین تبدیل می‌کرد. استفاده از Retrofit با ترکیب Gson برای Deserialization داده‌ها بسیار ساده و کارآمد بود.**
- Room: **برای ذخیره‌سازی داده‌ها به‌صورت محلی از کتابخانه Room استفاده شد. Room یک لایه انتزاعی بر روی SQLite فراهم می‌کند و به ما امکان می‌دهد که داده‌ها را به‌صورت آفلاین مدیریت کنیم. همچنین، Room با پشتیبانی از LiveData و Coroutines، امکان ذخیره‌سازی و بازیابی داده‌ها را به‌صورت هم‌زمان فراهم کرد.**
- Glide: **برای بارگذاری تصاویر از Glide استفاده شد. Glide یک کتابخانه قدرتمند برای بارگذاری و مدیریت تصاویر است که با بهینه‌سازی حافظه و کش[۱] تصاویر، عملکرد اپلیکیشن را بهبود می‌بخشد.**

۲-۴- یادگیری زبان کاتلین و ویژگی‌های آن

زبان کاتلین، به عنوان زبان رسمی توسعه اندروید توسط گوگل معرفی شده است. این زبان با ارائه ویژگی‌های پیشرفته و قابلیت‌های بهبودیافته نسبت به جاوا، تبدیل به انتخابی محبوب در جامعه توسعه‌دهندگان اندروید شده است. در این بخش، به یادگیری و پیاده‌سازی کدهای اندرویدی با استفاده از کاتلین پرداختم.

۲-۴-۱- مبانی کاتلین

یادگیری زبان کاتلین از بخش‌های اولیه مانند تعریف متغیرها و کنترل جریان شروع شد. کاتلین به ما این امکان را می‌دهد تا متغیرها را به دو صورت Immutable با استفاده از `val` و Mutable با استفاده از `var` تعریف کنیم. یکی از ویژگی‌های مهم کاتلین، پشتیبانی از Null Safety است که از بروز خطاهای مربوط به NullPointerException جلوگیری می‌کند.

یکی از مزایای کاتلین نسبت به زبان‌های دیگر، پشتیبانی از ویژگی‌های پیشرفته‌ای مانند Type Inference است. این ویژگی به ما این امکان را می‌دهد که بدون نیاز به مشخص کردن نوع متغیرها به‌صورت صریح، کاتلین به‌طور خودکار نوع داده را بر اساس مقدار انتساب داده‌شده تعیین کند. این قابلیت باعث کاهش کدنویسی و ساده‌تر شدن فرآیند تعریف متغیرها می‌شود.

همچنین، کاتلین از ساختارهای کنترلی مشابه زبان جاوا استفاده می‌کند، اما با بهبودهای زیادی همراه است. به‌عنوان مثال، در زبان جاوا از ساختار switch-case استفاده می‌شود، در حالی که در کاتلین از when به‌عنوان یک ساختار قدرتمندتر استفاده می‌شود که قابلیت مدیریت پیچیدگی‌های بیشتری را دارد:

---

[۱] Caching

```
val day = when(today) {
    1 -> "Sunday"
    2 -> "Monday"
    3 -> "Tuesday"
    else -> "Invalid day"
}
```

شکل ۱-۲: استفاده از ساختار when برای تعیین روز هفته بر اساس مقدار ورودی

۲-۴-۲- توابع Extension و Higher-Order Functions

یکی از ویژگی‌های بسیار مفید کاتلین، توابع Extension است. این قابلیت به ما امکان می‌دهد تا بدون تغییر در کدهای اصلی یک کلاس، توابع جدیدی به آن کلاس اضافه کنیم. این قابلیت در کاتلین به ما کمک کرد تا عملکردهای اضافی را به کلاس‌های استاندارد اضافه کنیم. برای مثال، در پروژه ما از Extension Functions برای اضافه‌کردن توابع کمکی به کلاس‌های اندروید مانند Activity و Fragment استفاده شد. به‌عنوان مثال:

```
fun Fragment.showToast(message: String) {
    Toast.makeText(requireContext(), message, Toast.LENGTH_SHORT).show()
}
```

شکل ۲-۲: تعریف تابع Extention در Fragment برای نمایش پیام‌های Toast

همچنین، کاتلین از Higher-Order Functions یا توابع مرتبه بالاتر پشتیبانی می‌کند که به ما این امکان را می‌دهد که توابع را به‌عنوان پارامتر به توابع دیگر ارسال کنیم یا تابعی را به‌عنوان نتیجه برگردانیم. این ویژگی به‌ویژه در توسعه اپلیکیشن‌های اندروید برای مدیریت رویدادها و بازگشت‌ها[۱] بسیار مفید بود.

۲-۴-۳- Coroutines و مدیریت عملیات ناهمگام

یکی دیگر از نقاط قوت کاتلین، Coroutines است. در اندروید، بسیاری از عملیات‌ها مانند فراخوانی APIها یا دسترسی به دیتابیس نیازمند پردازش‌های طولانی هستند که نباید در Thread اصلی[۲] انجام شوند. استفاده از Coroutines به ما این امکان را داد که این عملیات‌ها را به‌صورت ناهمگام و بدون مسدودکردن UI اجرا کنیم.

در طول پروژه، از Coroutines برای مدیریت درخواست‌های شبکه و همچنین تعامل با دیتابیس Room استفاده کردم. به‌عنوان مثال، با استفاده از `suspend` تابع‌ها و `launch` در ViewModelها، توانستم عملیات‌های طولانی‌مدت مانند دریافت داده‌ها از سرور را بدون مسدودکردن رابط کاربری مدیریت کنم:

---

[۱] Callbacks

[۲] UI Thread

```
viewModelScope.launch {
    val response = apiService.getData()
    _liveData.postValue(response)
}
```

شکل ۳-۲: اجرای درخواست ناهمگام با Coroutines و به‌روزرسانی Live Data با استفاده از ViewModel

۲-۵- توسعه و پیاده‌سازی ویژگی‌های جدید

در بخش‌های بعدی کارآموزی، مسئولیت توسعه و پیاده‌سازی ویژگی‌های جدید در اپلیکیشن به من محول شد. این بخش شامل طراحی صفحات جدید، اتصال به APIهای خارجی، مدیریت داده‌ها و بهینه‌سازی رابط کاربری بود.

۲-۵-۱- طراحی و توسعه ماژول‌های جدید

یکی از پروژه‌های اصلی که در طول دوره انجام دادم، توسعه یک ماژول جدید برای مدیریت حضور و غیاب کارکنان بود. این ماژول به کاربر امکان می‌داد که اطلاعات مربوط به حضور و غیاب خود را مشاهده و مدیریت کند. طراحی این ماژول شامل دو بخش اصلی بود:

- نمایش لیست حضور و غیاب: از RecyclerView برای نمایش لیستی از رویدادهای حضور و غیاب استفاده شد. هر آیتم شامل تاریخ، ساعت و وضعیت حضور بود.
- فیلتر کردن داده‌ها: قابلیت فیلتر کردن لیست بر اساس تاریخ و وضعیت نیز پیاده‌سازی شد که با استفاده از Spinner و DatePicker انجام گرفت.

```
recyclerView.adapter = AttendanceAdapter(attendanceList)
datePicker.setOnDateChangedListener { _, year, month, day ->
    filterAttendance(year, month, day)
}
```

شکل ۴-۲: تنظیم Adapter برای RecyclerView و استفاده از DatePicker برای فیلتر کردن داده‌های حضور و غیاب بر اساس تاریخ

۲-۵-۲- اتصال به APIها و مدیریت داده‌ها

در توسعه این ماژول، برای دریافت داده‌های حضور و غیاب از سرور، از Retrofit استفاده کردم. داده‌ها از سرور به‌صورت JSON دریافت می‌شد و با استفاده از Gson به مدل‌های کاتلین تبدیل می‌شد. در این پروژه، مدیریت صحیح خطاها و نمایش پیغام‌های مناسب به کاربر نیز از اهمیت ویژه‌ای برخوردار بود. برای مثال، در صورت بروز خطا در اتصال به سرور، یک پیغام خطا به‌صورت Toast به کاربر نمایش داده می‌شد:

```
try {
    val response = apiService.getAttendance()
    if(response.isSuccessful) {
        attendanceList = response.body() ?: listOf()
        updateRecyclerView()
    } else {
        showToast("Failed to fetch data")
    }
} catch (e: Exception) {
    showToast("Network error")
}
```

شکل ۵-۲: مدیریت درخواست API با استفاده از Try-Catch برای شناسایی خطاهای شبکه و بررسی موفقیت پاسخ

۳-۵-۲- مدیریت وضعیت آنلاین و آفلاین

یکی از چالش‌های این ماژول، مدیریت وضعیت آنلاین و آفلاین بود. برای اطمینان از عملکرد صحیح اپلیکیشن در حالت آفلاین، داده‌های دریافت‌شده از سرور در دیتابیس محلی (Room) ذخیره می‌شدند تا در زمان‌هایی که اتصال به اینترنت برقرار نبود، همچنان به این داده‌ها دسترسی داشته باشیم.
برای همگام‌سازی داده‌ها بین حالت آنلاین و آفلاین، از یک الگوریتم ساده همگام‌سازی استفاده شد که پس از اتصال مجدد به اینترنت، داده‌های جدید را از سرور دریافت و به‌روزرسانی می‌کرد.
این فرآیند با استفاده از Coroutines و LiveData انجام شد تا داده‌ها به‌صورت واکنش‌گرا در رابط کاربری نمایش داده شوند.

۶-۲- طراحی رابط کاربری[۱] و بهبود تجربه کاربری[۲]

یکی دیگر از بخش‌های مهم کارآموزی، طراحی رابط کاربری و بهینه‌سازی تجربه کاربری بود. رابط کاربری به‌عنوان نقطه تعامل اصلی کاربران با اپلیکیشن اهمیت ویژه‌ای دارد و نیازمند طراحی دقیق و بهینه است.

۱-۶-۲- طراحی بر اساس Material Design

در این پروژه، از اصول Material Design که توسط گوگل معرفی شده است، استفاده شد. Material Design استانداردهایی را برای طراحی عناصر رابط کاربری مانند دکمه‌ها، کارت‌ها و فضاهای خالی تعیین می‌کند. استفاده از Material Design باعث شد تا اپلیکیشن ظاهر مدرن‌تری داشته باشد و تجربه کاربری آن بهبود یابد.
برای پیاده‌سازی این طراحی‌ها، از عناصر Material مانند CardView و FloatingActionButton استفاده شد. همچنین از Themeing برای تطبیق رابط کاربری با تم تاریک و روشن[۳] بهره بردم. این قابلیت به کاربر اجازه می‌دهد تا به‌راحتی بین تم‌ها جابه‌جا شود و تجربه کاربری شخصی‌سازی‌شده‌ای داشته باشد.

---

۱ UI (User Interface)
۲ UX (User Experience)
۳ Dark/Light Mode

۲-۶-۲- استفاده از ConstraintLayout برای پیاده‌سازی Layoutها

برای پیاده‌سازی Layoutهای پیچیده در اندروید، از ConstraintLayout استفاده کردم. ConstraintLayout به ما امکان می‌دهد تا چیدمان‌های پیچیده‌ای را با استفاده از محدودیت‌های[۱] مختلف پیاده‌سازی کنیم و این کار را بدون نیاز به تو در تو شدن Layoutها انجام دهیم. این قابلیت به بهینه‌سازی عملکرد اپلیکیشن کمک زیادی کرد و زمان بارگذاری صفحات را به‌طور محسوسی کاهش داد.

به‌عنوان مثال، یکی از صفحاتی که با استفاده از ConstraintLayout پیاده‌سازی شد، صفحه اصلی اپلیکیشن بود که شامل چندین Button و TextView با طرح‌بندی پیچیده بود. با استفاده از Constraints، موقعیت هر عنصر نسبت به عناصر دیگر و نسبت به لبه‌های صفحه تعیین شد.

```
<Button
    android:id="@+id/buttonSubmit"
    app:layout_constraintTop_toBottomOf="@id/textView"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    android:text="Submit" />
```

شکل ۶-۲: تعریف دکمه Submit در ConstraintLayout با تنظیم موقعیت آن نسبت به سایر عناصرUI

۲-۷- مدیریت داده‌ها و ارتباط با دیتابیس

در هر اپلیکیشن اندروید، مدیریت داده‌ها و تعامل با منابع داده مانند دیتابیس محلی اهمیت ویژه‌ای دارد. در پروژه شرکت کسرا، از کتابخانه Room برای مدیریت داده‌های محلی استفاده شد که یک ابزار قدرتمند برای کار با دیتابیس SQLite است. Room با ارائه یک لایه انتزاعی بر روی SQLite، امکان مدیریت آسان‌تر و کارآمدتر داده‌ها را فراهم می‌کند.

۲-۷-۱- پیاده‌سازی Room برای ذخیره‌سازی داده‌ها

Room یکی از اجزای Android Jetpack است که امکان کار با دیتابیس SQLite را به‌صورت امن و کارآمد فراهم می‌کند. در این پروژه، Room برای مدیریت داده‌های حضور و غیاب به کار گرفته شد. ابتدا مدل‌های داده به‌عنوان Entity تعریف شدند. هر Entity نماینده یک جدول در دیتابیس بود. به‌عنوان مثال، جدول حضور و غیاب به‌صورت زیر تعریف شد:

---

[۱] Constraints

```
@Entity(tableName = "attendance_table")
data class Attendance(
    @PrimaryKey(autoGenerate = true)
    val id: Int,
    val employeeId: Int,
    val date: String,
    val status: String
)
```

شکل ۷-۲-: تعریف یک Entity در Room برای ذخیره داده‌های حضور و غیاب با مشخصات کلیدی از جمله شناسه، شناسه کارکنان، تاریخ و وضعیت

سپس برای انجام عملیات‌های CRUD (Create, Read, Update, Delete)، از DAO (Data Access Object) استفاده شد. DAOها رابطی برای انجام عملیات بر روی دیتابیس فراهم می‌کنند. به‌عنوان مثال، برای دریافت تمامی داده‌های حضور و غیاب، کدی مشابه زیر نوشته شد:

```
@Dao
interface AttendanceDao {
    @Query("SELECT * FROM attendance_table")
    fun getAllAttendance(): LiveData<List<Attendance>>
}
```

شکل ۸-۲: تعریف یک DAO در Room برای بازیابی تمامی داده‌های حضور و غیاب به‌صورت واکنش‌گرا از جدول دیتابیس

استفاده از LiveData در این مثال باعث شد که داده‌ها به‌صورت واکنش‌گرا[۱] به رابط کاربری ارسال شوند. هرگاه تغییری در دیتابیس ایجاد شود، LiveData به‌طور خودکار UI را به‌روز می‌کند.

۲-۷-۲- مدیریت داده‌های آفلاین و همگام‌سازی با سرور

یکی از مهم‌ترین بخش‌های این پروژه، امکان دسترسی به داده‌ها در حالت آفلاین بود. برای این منظور، داده‌هایی که از سرور دریافت می‌شدند، در دیتابیس محلی ذخیره می‌شدند تا در مواقعی که دسترسی به اینترنت وجود ندارد، همچنان کاربر بتواند به داده‌ها دسترسی داشته باشد. این ویژگی با استفاده از Room و قابلیت Caching به‌طور مؤثر پیاده‌سازی شد.
در زمان اتصال مجدد به اینترنت، داده‌های جدید از سرور دریافت و در دیتابیس محلی ذخیره می‌شدند. برای این کار، از یک مکانیزم ساده Synchronization استفاده شد که در آن، داده‌های جدید با داده‌های محلی مقایسه و به‌روزرسانی می‌شدند.

۳-۷-۲- بهینه‌سازی دسترسی به دیتابیس

برای بهبود عملکرد اپلیکیشن در زمان دسترسی به دیتابیس، از بهینه‌سازی‌های مختلفی استفاده شد. به‌عنوان مثال، Queryهای پیچیده‌ای که نیاز به محاسبات زیاد داشتند، بهینه‌سازی شدند تا عملکرد

[۱] Reactive

سریع‌تری داشته باشند. یکی از این بهینه‌سازی‌ها شامل استفاده از Indexes بر روی ستون‌های مهم جدول بود که سرعت جستجو را بهبود بخشید.

۸-۲- تست و رفع اشکال

تست و اطمینان از صحت عملکرد اپلیکیشن، یکی از مراحل حیاتی در فرآیند توسعه است. در این پروژه، از انواع مختلفی از تست‌ها استفاده شد تا مطمئن شویم که تمامی بخش‌های اپلیکیشن به‌درستی کار می‌کنند. این تست‌ها شامل تست واحد[۱]، تست رابط کاربری[۲] و تست یکپارچه‌سازی[۳] بود.

۱-۸-۲- تست واحد

برای اطمینان از عملکرد صحیح بخش‌های مختلف کد، از تست‌های واحد استفاده شد. این تست‌ها با استفاده از JUnit و Mockito پیاده‌سازی شدند. JUnit به ما امکان نوشتن تست‌های مستقل برای توابع و کلاس‌های مختلف را داد، در حالی که Mockito برای Mock کردن وابستگی‌ها به کار رفت. این ابزارها به ما این امکان را دادند که توابع مختلف را بدون نیاز به اجرای کامل اپلیکیشن آزمایش کنیم.
به‌عنوان مثال، برای تست تابعی که حضور و غیاب کارکنان را محاسبه می‌کرد، کد زیر نوشته شد:

```
@Test
fun calculateAttendanceTest() {
    val attendanceList = listOf(
        Attendance(1, 1001, "2024-09-10", "Present"),
        Attendance(2, 1001, "2024-09-11", "Absent")
    )

    val result = calculateAttendance(attendanceList)
    assertEquals(50, result) // Expecting 50% attendance
}
```

شکل ۹-۲: تست واحد برای تابع محاسبه حضور و غیاب با استفاده از لیست نمونه و بررسی نتیجه درصد حضور کارکنان با استفاده از assertEquals

این تست‌ها به ما کمک کرد تا مطمئن شویم که کدهای اصلی به‌درستی کار می‌کنند و در صورت تغییرات در کدها، عملکرد قبلی مختل نمی‌شود.

۲-۸-۲- تست رابط کاربری

در کنار تست‌های واحد، تست‌های رابط کاربری نیز با استفاده از Espresso پیاده‌سازی شدند. این تست‌ها به‌ویژه برای اطمینان از عملکرد صحیح تعاملات کاربر با اپلیکیشن به‌کار رفتند. با استفاده از Espresso، شبیه‌سازی تعاملات کاربر مانند کلیک کردن روی دکمه‌ها، وارد کردن اطلاعات در فرم‌ها و پیمایش بین صفحات انجام شد.
به‌عنوان مثال، برای تست عملکرد یک دکمه که لیست حضور و غیاب را بارگذاری می‌کند، از کد زیر استفاده شد:

---

[۱] Unit Testing
[۲] UI Testing
[3] Integration Testing

```
@Test
fun testAttendanceButtonClick() {
    onView(withId(R.id.buttonLoadAttendance)).perform(click())
    onView(withId(R.id.recyclerViewAttendance))
        .check(matches(isDisplayed()))
}
```

شکل ۱۰-۲: تست عملکرد دکمه بارگذاری حضور و غیاب با استفاده از Espresso و بررسی نمایش لیست حضور و غیاب در RecyclerView

۳-۸-۲- تست یکپارچه‌سازی و CI/CD

برای اطمینان از عملکرد هماهنگ بین بخش‌های مختلف اپلیکیشن، تست‌های یکپارچه‌سازی نیز انجام شد. این تست‌ها شامل بررسی تعامل بین کلاس‌های مختلف، سرویس‌ها و پایگاه داده بود. همچنین با استفاده از Jenkins به‌عنوان ابزار CI/CD)، فرآیندهای تست به‌صورت خودکار اجرا شدند. هر بار که تغییری در کدها اعمال می‌شد، Jenkins به‌طور خودکار تست‌ها را اجرا می‌کرد و گزارش آن‌ها را ارائه می‌داد.

۹-۲- بهینه‌سازی عملکرد اپلیکیشن

یکی از بخش‌های حیاتی در هر پروژه موبایلی، بهینه‌سازی عملکرد اپلیکیشن است. در طول دوره کارآموزی، من مسئول بهینه‌سازی بخش‌های مختلف پروژه بودم تا عملکرد کلی اپلیکیشن بهبود یابد و تجربه کاربری بهتری ارائه شود.

۱-۹-۲- مدیریت حافظه و بهینه‌سازی مصرف RAM

یکی از چالش‌های اصلی در توسعه اپلیکیشن‌های اندروید، مدیریت بهینه حافظه است. در این پروژه، با استفاده از ابزار Android Profiler به تحلیل مصرف حافظه پرداختم و نشت‌های حافظه[۱] را شناسایی کردم. برای مثال، برخی از نشت‌ها به دلیل عدم حذف صحیح Listenerها و Contextها از اشیاء به‌وجود آمده بود. با استفاده از ابزار LeakCanary، توانستم مشکلات مربوط به مدیریت حافظه را شناسایی کرده و آن‌ها را برطرف کنم.

۲-۹-۲- بهینه‌سازی سرعت بارگذاری صفحات

یکی از بخش‌های مهم بهینه‌سازی، کاهش زمان بارگذاری صفحات و نمایش سریع‌تر داده‌ها به کاربر بود. برای این منظور، از Lazy Loading استفاده شد که به ما امکان می‌داد تنها داده‌هایی را که نیاز است به کاربر نمایش داده شود، بارگذاری کنیم. این کار به‌خصوص در زمان بارگذاری تصاویر از سرور با استفاده از Glide انجام شد. همچنین با بهینه‌سازی Queryهای دیتابیس، زمان پاسخ‌دهی به درخواست‌های کاربر بهبود یافت.

۳-۹-۲- بهینه‌سازی عملکرد شبکه و کاهش مصرف پهنای باند

در بخش ارتباطات شبکه، از قابلیت Caching در Retrofit استفاده شد تا درخواست‌های مکرر به سرور کاهش یابد. به‌عنوان مثال، داده‌های دریافت‌شده از سرور در حافظه محلی[۲] ذخیره می‌شدند و در درخواست‌های بعدی، تنها در صورتی که داده‌ها تغییر کرده باشند، از سرور دریافت می‌شدند. این کار علاوه بر کاهش مصرف پهنای باند، باعث بهبود زمان بارگذاری داده‌ها نیز شد.

---

۱ Memory Leaks
۲ Cache

۲-۱۰- مستندسازی و جمع‌بندی فعالیت‌ها

مستندسازی یکی از بخش‌های مهم در هر پروژه نرم‌افزاری است که به توسعه‌دهندگان دیگر کمک می‌کند تا به‌سرعت با پروژه آشنا شوند و بتوانند به‌راحتی در توسعه آن مشارکت کنند.

۲-۱۰-۱- مستندسازی کد و کامنت‌گذاری

در طول دوره، تمامی کدهای نوشته‌شده با استفاده از کامنت‌ها مستندسازی شدند. کامنت‌ها توضیحاتی درباره عملکرد توابع، کلاس‌ها و بخش‌های پیچیده کد ارائه می‌دادند. این کامنت‌ها به‌خصوص در بخش‌هایی که از الگوریتم‌های پیچیده استفاده می‌شد، بسیار مفید بودند.

۲-۱۰-۲- استفاده از Dokka برای مستندسازی خودکار

برای مستندسازی خودکار کدها، از ابزار Dokka که مخصوص پروژه‌های کاتلین است، استفاده شد. Dokka به‌صورت خودکار مستندات مرتبط با کلاس‌ها و توابع پروژه را از کامنت‌های موجود استخراج کرده و آن‌ها را به‌صورت HTML یا Markdown ذخیره می‌کرد. این مستندات در اختیار توسعه‌دهندگان جدید قرار گرفت تا به‌سرعت با پروژه آشنا شوند.

۲-۱۰-۳- تهیه راهنمای استفاده از اپلیکیشن

علاوه بر مستندسازی کد، یک راهنمای استفاده برای توسعه‌دهندگان و کاربران تهیه شد که شامل توضیحات کامل در مورد نحوه نصب، راه‌اندازی و استفاده از اپلیکیشن بود. این مستند شامل توضیحات مربوط به استفاده از Git برای مدیریت نسخه‌ها، Android Studio برای توسعه و همچنین روش‌های تست و اشکال‌زدایی بود.

۲-۱۱- جمع بندی

در این فصل، به‌طور جامع به شرح تمامی فعالیت‌های انجام‌شده در دوره کارآموزی پرداخته شد. از مراحل ابتدایی شامل یادگیری ابزارهای توسعه و زبان کاتلین تا مراحل پیشرفته‌تر شامل پیاده‌سازی ویژگی‌های جدید، بهینه‌سازی عملکرد و مستندسازی پروژه، تمامی بخش‌ها به‌تفصیل مورد بررسی قرار گرفتند. این دوره کارآموزی فرصتی برای یادگیری عملی و به‌کارگیری مفاهیم پیشرفته توسعه نرم‌افزار اندروید بود که در نهایت به بهبود مهارت‌ها و تجربه من در این حوزه کمک شایانی کرد.

فصل سوم

مدیریت عملیات ناهمگام با Coroutines در کاتلین

۳-۱- مقدمه

در فصل سوم این گزارش، به بررسی یکی از موضوعات مهم و کلیدی در توسعه اپلیکیشن‌های اندروید با زبان کاتلین، یعنی مدیریت عملیات ناهمگام[۱] با استفاده از Coroutines پرداخته می‌شود. عملیات ناهمگام یکی از مباحث حیاتی در توسعه نرم‌افزارهای موبایل است، زیرا بسیاری از وظایف در اپلیکیشن‌ها نیاز به پردازش‌هایی دارند که نمی‌توانند در Thread اصلی انجام شوند. این فصل به بررسی مفاهیم Coroutines، ساختار و نحوه پیاده‌سازی آن‌ها و همچنین مزایای استفاده از این تکنیک در مقایسه با سایر روش‌های مدیریت عملیات ناهمگام می‌پردازد.

۳-۲- مقدمه‌ای بر عملیات ناهمگام

در هر اپلیکیشن موبایل، بسیاری از وظایف مانند فراخوانی APIها، دسترسی به دیتابیس، دانلود فایل‌ها و سایر عملیات‌های سنگین، به زمان زیادی برای اجرا نیاز دارند. اگر این وظایف در Thread اصلی اجرا شوند، باعث مسدود شدن رابط کاربری و تجربه کاربری نامطلوبی خواهند شد. در نتیجه، باید این عملیات‌ها به‌صورت ناهمگام انجام شوند تا به‌طور هم‌زمان بتوان از منابع دیگر مانند سرور یا حافظه داخلی استفاده کرد، بدون اینکه کاربر احساس کند که اپلیکیشن متوقف شده است.

در گذشته، برای مدیریت این عملیات از تکنیک‌هایی مانند Threads و AsyncTask استفاده می‌شد، اما این روش‌ها به دلیل پیچیدگی در مدیریت و نگهداری، مشکلاتی ایجاد می‌کردند. Coroutines در کاتلین به‌عنوان یک ابزار قدرتمند و ساده برای مدیریت عملیات ناهمگام معرفی شد که مزایای زیادی نسبت به روش‌های قبلی دارد.

۳-۳- معرفی Coroutines

Coroutines یک مکانیسم سبک و بهینه برای مدیریت عملیات ناهمگام و هم‌زمانی[۲] است که به توسعه‌دهندگان این امکان را می‌دهد که وظایف پیچیده و سنگین را بدون مسدود کردن Thread اصلی اجرا کنند. Coroutines به‌صورت یکپارچه با زبان کاتلین طراحی شده و امکان مدیریت عملیات طولانی را با کدنویسی ساده و بدون پیچیدگی‌های معمول در برنامه‌نویسی ناهمگام فراهم می‌کند.

---

[۱] Asynchronous Operation

[۲] Concurrecny

در واقع، Coroutines در کاتلین همانند یک Thread عمل می‌کنند، اما سبک‌تر و بهینه‌تر هستند. هر Coroutine می‌تواند در پس‌زمینه کار کند، در حالی که Thread اصلی به اجرای عملیات‌های دیگر می‌پردازد. به دلیل سبک بودن و عدم نیاز به منابع سنگین برای مدیریت، Coroutines بسیار کارآمدتر از Threadها هستند.

ویژگی‌های کلیدی Coroutines عبارتند از:

- **سبک بودن[۱]: برخلاف Threadها که منابع سنگینی نیاز دارند، Coroutines بسیار سبک هستند و می‌توان صدها Coroutine را در یک Thread اجرا کرد.**
- **تعلیق و ازسرگیری[۲] Coroutines: می‌توانند به‌راحتی تعلیق شوند و پس از انجام سایر وظایف دوباره از سر گرفته شوند.**
- **ادغام با کاتلین: Coroutines به‌طور کامل با زبان کاتلین ادغام شده و می‌توان آن‌ها را به‌سادگی با استفاده از کلمه کلیدی `suspend` تعریف کرد.**

۴-۳- ساختار Coroutines در کاتلین

Coroutines با استفاده از دو مفهوم اساسی Coroutine Builders و Suspend Functions در کاتلین پیاده‌سازی می‌شوند. در این بخش، به بررسی این دو مفهوم پرداخته و ساختار یک Coroutine ساده را توضیح می‌دهیم.

۱-۴-۳- Coroutine Builders

در کاتلین، برای راه‌اندازی یک Coroutine از Coroutine Builders استفاده می‌شود. دو نوع رایج از Coroutine Builders عبارتند از `launch` و `async`.

- **launch: این Builder برای راه‌اندازی یک Coroutine استفاده می‌شود که نتیجه‌ای را بازنمی‌گرداند و معمولاً برای اجرای عملیات‌هایی که نتیجه آن‌ها به کاربر بازگردانده نمی‌شود، استفاده می‌شود. به‌عنوان مثال:**

```
GlobalScope.launch {
    // Some background task
    delay(1000L) // Simulate long-running task
    println("Task completed")
}
```

شکل ۱-۳: اجرای تسک پس‌زمینه با استفاده از launch و تاخیر

- **async: این Builder برای راه‌اندازی Coroutinesی استفاده می‌شود که نتیجه‌ای را بازمی‌گردانند. `async` اغلب با `await` استفاده می‌شود تا نتیجه اجرای Coroutine به دست آید. برای مثال:**

---

[۱] Lightweight
[۲] Suspending and Resuming

```
val deferred = GlobalScope.async {
    // Some computation
    42 // Result of the computation
}

runBlocking {
    println("Result: ${deferred.await()}")
}
```

شکل ۲-۳: استفاده از async برای اجرای عملیات ناهمگام و دریافت نتیجه با await

۲-۳-۴- Suspend Functions

در کاتلین، توابعی که ممکن است عملیات ناهمگام انجام دهند، با استفاده از کلمه کلیدی `suspend` تعریف می‌شوند. توابع Suspend می‌توانند از Coroutines استفاده کنند و این امکان را فراهم می‌کنند که عملیات طولانی‌مدت بدون مسدود کردن Thread اجرا شود.

یک تابع `suspend` فقط می‌تواند از طریق Coroutine یا تابع `suspend` دیگری فراخوانی شود و به ما این امکان را می‌دهد که عملیات‌هایی مانند دسترسی به سرور یا دیتابیس را به‌صورت ناهمگام و کارآمد انجام دهیم.

۵-۳- مزایای استفاده از Coroutines در مقایسه با روش‌های سنتی

استفاده از Coroutines نسبت به روش‌های سنتی مانند Threadها یا AsyncTaskها مزایای بسیاری دارد. در ادامه به برخی از این مزایا اشاره می‌کنیم:

- سبک بودن و مدیریت منابع: یکی از مهم‌ترین مزایای Coroutines، سبک بودن آن‌ها است. هر Thread در یک سیستم، منابع سنگینی مصرف می‌کند و به همین دلیل نمی‌توان تعداد زیادی Thread را به‌طور هم‌زمان اجرا کرد. در مقابل، Coroutines بسیار سبک‌تر هستند و می‌توان صدها Coroutine را در یک Thread اجرا کرد.
- ساده‌تر شدن مدیریت ناهمگام: با استفاده از Threadها و AsyncTaskها، مدیریت عملیات ناهمگام معمولاً با پیچیدگی‌های زیادی همراه است. برای مثال، باید به دقت مدیریت Threadهای مختلف، ارتباط بین آن‌ها و همچنین برخورد با استثناهای احتمالی را انجام داد. Coroutines این فرآیندها را بسیار ساده‌تر می‌کند و با استفاده از کدهای ساده و خوانا می‌توان عملیات‌های ناهمگام را مدیریت کرد.
- کاهش خطاها و افزایش اطمینان: استفاده از Coroutines خطر بروز خطاهای رایجی مانند Race Conditions و Deadlocks را به میزان قابل توجهی کاهش می‌دهد. به دلیل اینکه Coroutineها به‌صورت منظم و بهینه مدیریت می‌شوند، احتمال بروز خطاهای هم‌زمانی کاهش پیدا می‌کند.
- پشتیبانی از مدیریت پیشرفته ناهمگام: Coroutines از Structured Concurrency پشتیبانی می‌کنند، به این معنی که Coroutineها به‌صورت ساختاریافته و منظم مدیریت می‌شوند. این ویژگی به ما امکان می‌دهد که Coroutineهای ایجادشده را به‌صورت ساختاریافته نگهداری و مدیریت کنیم و از بروز خطاهای مرتبط با هم‌زمانی جلوگیری کنیم.

۳-۶- استفاده عملی از Coroutines در پروژه حضور و غیاب

در پروژه حضور و غیاب شرکت کسرا، از Coroutines برای مدیریت درخواست‌های شبکه و دسترسی به دیتابیس استفاده شد. به دلیل این‌که بسیاری از عملیات‌ها مانند دریافت داده‌ها از سرور زمان‌بر هستند، استفاده از Coroutines برای مدیریت این وظایف اهمیت زیادی داشت.

۳-۶-۱- دریافت داده‌ها از API به‌صورت ناهمگام

در این پروژه، داده‌های حضور و غیاب کارکنان از طریق APIهای REST از سرور دریافت می‌شدند. برای جلوگیری از مسدود شدن رابط کاربری در هنگام دریافت داده‌ها، از Coroutines استفاده شد. به‌عنوان مثال، درخواست به سرور با استفاده از Retrofit در داخل یک Coroutine انجام شد:

```
fun fetchAttendance() {
    viewModelScope.launch {
        try {
            val response = apiService.getAttendance()
            if (response.isSuccessful) {
                _attendanceData.value = response.body()
            } else {
                showError("Error fetching data")
            }
        } catch (e: Exception) {
            showError("Network error")
        }
    }
}
```

شکل ۳-۳: درخواست API ناهمگام برای دریافت داده‌های حضور و غیاب و مدیریت خطا با Coroutines در ViewModel

در این مثال، از Coroutine Builder `launch` برای اجرای درخواست به‌صورت ناهمگام استفاده شده است. این کد به‌صورت روان اجرا می‌شود و حتی در صورت طولانی بودن زمان پاسخ سرور، UI مسدود نمی‌شود.

۳-۶-۲- تعامل با دیتابیس به‌صورت ناهمگام

علاوه بر ارتباط با سرور، در این پروژه از Coroutines برای دسترسی به دیتابیس محلی (Room) نیز استفاده شد. عملیات‌هایی مانند ذخیره و بازیابی داده‌ها از دیتابیس که نیازمند زمان بیشتری هستند، با استفاده از توابع `suspend` و Coroutines به‌صورت ناهمگام انجام شدند.

```
suspend fun insertAttendance(attendance: Attendance) {
    attendanceDao.insert(attendance)
}
```

شکل ۳-۴: تابع suspend برای درج داده‌های حضور و غیاب در دیتابیس با استفاده از Coroutines

با استفاده از این روش، عملیات‌هایی مانند ذخیره و بارگذاری داده‌ها به‌صورت ناهمگام و بدون اختلال در رابط کاربری انجام شدند.

۷-۳- جمع بندی

استفاده از Coroutines در کاتلین یک روش مؤثر و بهینه برای مدیریت عملیات ناهمگام و هم‌زمانی است. Coroutines به توسعه‌دهندگان اندروید امکان می‌دهد که وظایف سنگین و طولانی‌مدت را بدون مسدود کردن رابط کاربری و با استفاده از منابع کمتر اجرا کنند. در پروژه حضور و غیاب شرکت کسرا، استفاده از Coroutines باعث افزایش کارایی، کاهش پیچیدگی‌های کدنویسی و بهبود تجربه کاربری شد. با استفاده از این تکنیک، می‌توان به‌صورت بهینه و کارآمد عملیات‌های سنگین را در پس‌زمینه اجرا کرد و در عین حال رابط کاربری را بدون هیچ‌گونه کندی یا توقفی نگه داشت.

پیوست ۱

بهترین روش‌های بهینه‌سازی عملکرد در اندروید

در زیر به برخی از نکات کلیدی برای بهینه‌سازی عملکرد اپلیکیشن‌های اندروید پرداخته شده است. این نکات در طول دوره کارآموزی به‌کار گرفته شدند تا اپلیکیشن بهینه‌تر و سریع‌تر شود.

مدیریت حافظه:

- **استفاده از Glide برای مدیریت تصاویر: برای بهبود عملکرد و جلوگیری از مصرف بیش‌ازحد حافظه، از کتابخانه Glide برای بارگذاری و نمایش تصاویر استفاده شد.**
- **استفاده از LeakCanary برای شناسایی Memory Leaks: ابزار LeakCanary برای شناسایی مشکلات نشت حافظه به کار گرفته شد. این ابزار به‌صورت خودکار نشت حافظه را تشخیص داده و گزارش آن را ارائه می‌دهد.**

بهینه‌سازی Queryهای دیتابیس:

- **استفاده از Indexها: برای بهبود سرعت دسترسی به داده‌ها، از **Index** بر روی ستون‌های مهم دیتابیس استفاده شد.**
- **Lazy Loading: برای جلوگیری از بارگذاری بی‌مورد داده‌ها، تنها داده‌هایی که کاربر به آن‌ها نیاز داشت بارگذاری شدند.**

بهینه‌سازی عملکرد شبکه:

- **Caching در Retrofit: استفاده از قابلیت Caching در Retrofit برای کاهش تعداد درخواست‌ها به سرور و افزایش سرعت بارگذاری داده‌ها.**
- **Timeout تنظیم شده: برای بهبود تجربه کاربری، زمان‌های Timeout برای درخواست‌های شبکه بهینه‌سازی شد تا در صورت بروز خطا، اپلیکیشن به‌سرعت پاسخ مناسبی ارائه دهد.**

مراجع

[1] Android Developers. (n.d.). Android Studio. Retrieved from https://developer.android.com/studio

[2] Git. (n.d.). Git و GitHub. Retrieved from https://git-scm.com

[3] Postman. (n.d.). Postman API Development. Retrieved from https://www.postman.com

[4] Google. (n.d.). MVVM Architecture. Retrieved from https://developer.android.com/jetpack/guide

[5] Square. (n.d.). Retrofit: Type-safe HTTP client for Android and Java. Retrieved from https://square.github.io/retrofit

[6] Android Developers. (n.d.). Room: Persistence Library. Retrieved from https://developer.android.com/training/data-storage/room

[7] BumpTech. (n.d.). Glide Image Loading Library. Retrieved from https://bumptech.github.io/glide

[8] Kotlin. (n.d.). Kotlin Language Documentation. Retrieved from https://kotlinlang.org/docs/home.html

[9] Kotlin. (n.d.). Extension and Higher-Order Functions. Retrieved from https://kotlinlang.org/docs/functions.html

[10] Google. (n.d.). Kotlin Coroutines. Retrieved from https://kotlinlang.org/docs/coroutines-overview.html

[11] Google. (n.d.). Material Design Guidelines. Retrieved from https://material.io/design

[12] Google. (n.d.). Unit Testing. Retrieved from https://developer.android.com/training/testing/unit-testing

[13] Google. (n.d.). UI Testing. Retrieved from https://developer.android.com/training/testing/ui-testing

[14] Android Developers. (n.d.). Android Profiler. Retrieved from https://developer.android.com/studio/profile/android-profiler

[15] Google. (n.d.). LeakCanary. Retrieved from https://square.github.io/leakcanary

[16] Google. (n.d.). AsyncTask. Retrieved from https://developer.android.com/reference/android/os/AsyncTask

[17] Google. (n.d.). Best Practices for Memory Management. Retrieved from https://developer.android.com/topic/performance/memory