



دانشگاه شهرد

دانشکده مهندسی کامپیوتر

پایان نامه کارشناسی مهندسی کامپیوتر

گرایش نرم افزار

اپلیکیشن جستجوی بهینه در اسناد

نگارش:

احمد رضا گل کارنور

استاد راهنما:

استاد علی محمد زارع بیدکی

آذر ۱۴۰۰

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

کلیه حقوق مادی مترتب بر نتایج
مطالعات، ابتکارات و نوآوری‌های
ناشی از تحقیق موضوع این پایان‌نامه
متعلق به دانشکده مهندسی کامپیوتر دانشگاه یزد است.

تقدیم

تقدیم به پدر و مادر مهربانم که در مسیر ناهموار زندگی همواره مشوق و پشتیبان راهم بوده‌اند و همچنین تقدیم به استاد ارجمندم، دکتر زارع بیدکی به پاس قدردانی از راهنمایی‌های ارزشمندشان.

تشکر و قدردانی

از اساتید بزرگ دانشگاه یزد که در این چهار سال مرا از تجربیات و دانش خود بهره‌مند ساختند تشکر و قدردانی می‌نمایم.

همچنین از استاد راهنمای خود، دکتر زارع بیدکی، که در پیاده‌سازی پروژه و در جهت‌گیری بنده در گرایش نرم‌افزار به دلیل تدریس فوق‌العاده کامل ایشان در دروس برنامه‌نویسی بسیار تشکر و قدردانی می‌کنم. بدون کمک شما، یادگیری دروس برنامه‌نویسی به این سادگی ممکن نبود. از تدریس کامل و بدون نقص شما در گروه مهندسی کامپیوتر دانشگاه یزد بسیار تشکر و قدردانی می‌نمایم.

در آخر از مهم‌ترین افراد زندگی‌ام، پدر و مادرم که همواره حامی و پشتیبان من بوده و در مسیر یادگیری همیشه در کنار من ایستاده‌اند، بسیار سپاس‌گذارم.

چکیده

اسناد در گوشی‌های هوشمند از داده‌های بسیار ارزشمند می‌باشد و وجود این داده‌ها در گوشی مانند وجود کارت بانکی در کیف پول ارزشمند است. بسیاری از اسناد دارای داده‌های مهمی هستند که کاربر ممکن است هر هفته به آن‌ها نیاز داشته باشد؛ یا ممکن است اسنادی وجود داشته باشند که سالیانه تنها یک بار استفاده داشته باشند ولی وجود آن‌ها در دستگاه هوشمند و همراه کاربر بسیار ضروری است. در بسیاری از حالت‌ها و شرایط پیچیده، کاربر اطلاعی از اینکه سند مورد نظر خود را کجا ذخیره کرده است و یا آیا اصلاً این سند را در دستگاه هوشمند خود نگه داشته است یا پاک کرده است اطلاعی ندارد. و برنامه‌ای بصورت پیش فرض در دستگاه‌های هوشمند وجود ندارد که بتواند براساس فرمت خاص، اسناد مورد نظر را جستجو کند یا بتواند داخل اسناد را بخواند و براساس پرس و جو خود اسناد را رتبه‌بندی کند تا به سند مورد نظر خود برسد. پلیکیشن سند یاب گلی سعی دارد با بهینه ترین روش ممکن و کمترین فشار روی هسته‌های پردازنده دستگاه، به جستجوی داخل اسناد موجود در دستگاه اندرویدی براساس پرس و جو مورد نظر کاربر بپردازد و سندهایی که به کوئری مورد نظر کاربر نزدیک تر هستند را از سندهای دیگر جدا کرده و به کاربر نمایش دهد.

کلید واژه: اپلیکیشن سند یاب گلی، جستجوی اسناد براساس نام و فرمت و پرس و جو، یافتن اسناد.

فهرست مطالب

عنوان	صفحه
فهرست کدها.....	ز
فهرست شکل ها.....	د
فهرست علامت‌های اختصاری.....	و
فصل ۱: مقدمه.....	۱
۱-۱ پیشگفتار.....	۱
۲-۱ هدف از طراحی این اپلیکیشن.....	۱
۳-۱ ساختار بیان مطالب.....	۱
۱-۳-۱ بخش‌های اصلی.....	۱
۲-۳-۱ بخش‌های جزئی.....	۲
فصل ۲: روش حل مسئله برای امتیازدهی و جستجوی اسناد.....	۳
۱-۲ مقدمه.....	۳
۲-۲ داده‌های مورد نیاز.....	۳
۱-۲-۲ نام سند (اختیاری).....	۳
۲-۲-۲ آدرس اولیه.....	۳
۳-۲-۲ فرمت فایل مورد نظر.....	۴
۴-۲-۲ پرس‌وجو.....	۴
۳-۲ جاوا - جستجوی اسناد بر اساس نام.....	۴
۱-۳-۲ نحوه اجرای Search By Name.....	۴
۲-۳-۲ خروجی برنامه.....	۶
۴-۲ جاوا - جستجوی اسناد بر اساس پرس‌وجو.....	۶
۱-۴-۲ نحوه اجرای Search By Text.....	۶
۲-۴-۲ نحوه محاسبه امتیاز اسناد.....	۸
۳-۴-۲ کلاس WordSearchData.....	۸
۴-۴-۲ خروجی برنامه.....	۹

فصل ۳:	پایه‌سازی کدهای اندروید و تعریف اکتیویتی‌ها.....	۱۰
۱-۳	مقدمه.....	۱۰
۲-۳	دسترسی‌ها و حداقل نسخه اندروید مورد نیاز.....	۱۰
۲-۳	مفاهیم اولیه در برنامه‌نویسی اندروید.....	۱۰
۳-۳	کتابخانه‌ها و وابستگی‌های استفاده شده.....	۱۱
	Apache POI ۱-۳-۳.....	۱۲
	Apache PdfBox ۲-۳-۳.....	۱۲
	FancyToast Dependency ۳-۳-۳.....	۱۲
	UnicornFilePicker Dependency ۴-۳-۳.....	۱۲
	MaterialEditText Dependency ۵-۳-۳.....	۱۳
۴-۳	اکتیویتی‌ها و کلاس‌ها.....	۱۳
	MainActivity Class ۱-۴-۳.....	۱۴
	۱-۱-۴-۳ ساختار کد.....	۱۴
	SearchByNameTab Fragment ۲-۱-۴-۳.....	۱۶
	SearchByTextTab Fragment ۳-۱-۴-۳.....	۲۰
	AdapterTab ۴-۱-۴-۳.....	۲۳
	SearchByNameStartActivity ۲-۴-۳.....	۲۴
	۱-۲-۴-۳ ساختار کد.....	۲۴
	SearchByName_RecycleAdapter ۲-۲-۴-۳.....	۲۶
	SearchByName_ViewHolder ۳-۲-۴-۳.....	۲۷
	SearchByTextStartActivity ۳-۴-۳.....	۲۸
	۱-۳-۴-۳ ساختار کد.....	۲۸
	SearchByText_RecyclerAdapter ۲-۳-۴-۳.....	۳۳
	SeachByText_ViewHolder ۳-۳-۴-۳.....	۳۴
	ShowFileInfoActivity Class ۴-۴-۳.....	۳۵
	AskPermissionActivity Class ۵-۴-۳.....	۳۷
۵-۳	طراحی بخش UI.....	۳۸
	Layouts ۱-۵-۳.....	۳۸
	activity_main.xml ۱-۱-۵-۳.....	۳۸
	fragment_search_by_name_tab.xml ۲-۱-۵-۳.....	۳۹

۳۹.....	fragment_search_by_text_tab.xml	۳-۱-۵-۳
۴۰.....	my_toolbar.xml	۴-۱-۵-۳
۴۰.....	activity_search_by_name_start.xml	۵-۱-۵-۳
۴۱.....	search_by_name_viewholder.xml	۶-۱-۵-۳
۴۱.....	activity_search_by_text_start.xml	۷-۱-۵-۳
۴۲.....	search_by_text_viewholder.xml	۸-۱-۵-۳
۴۲.....	activity_show_file_info.xml	۹-۱-۵-۳
۴۳.....	activity_ask_permission.xml	۱۰-۱-۵-۳
۴۴.....	drawable	۲-۵-۳
۴۴.....	mipmap	۳-۵-۳
۴۴.....	values	۲-۵-۳
۴۵.....	بررسی کارکرد اپلیکیشن و اشکالات رفع شده.....	فصل ۴:
۴۵.....	مشکلات رفع شده.....	۱-۴
۴۶.....	نمونه خروجی اپلیکیشن.....	۲-۴
۴۸.....	فهرست مراجع.....	

فهرست کدها

عنوان	صفحه
کد ۱-۲: متد SreachForFilesOnly	۵
کد ۲-۲: متد doInBackground از کلاس startLongSreaching	۷
کد ۲-۳: کلاس WordSearchData	۸
کد ۳-۱: دسترسی خواندن حافظه	۱۰
کد ۳-۲: MainActivity Class بخش اول	۱۴
کد ۳-۳: MainActivity Class بخش دوم	۱۵
کد ۳-۴: MainActivity Class – addOnPageChangeListener	۱۵
کد ۳-۵: MainActivityClass – بررسی دسترسی خواندن حافظه	۱۶
کد ۳-۶: SearchByNameTab بخش اول	۱۶
کد ۳-۷: SearchByNameTab بخش دوم	۱۷
کد ۳-۸: SearchByNameTab بخش سوم	۱۷
کد ۳-۹: SearchByNameTab بخش چهارم	۱۸
کد ۳-۱۰: SearchByNameTab بخش پنجم	۱۹
کد ۳-۱۱: SearchByNameTab بخش ششم	۱۹
کد ۳-۱۲: SearchByTextTab بخش اول	۲۰
کد ۳-۱۳: SearchByTextTab بخش دوم	۲۰
کد ۳-۱۴: SearchByTextTab بخش سوم	۲۱
کد ۳-۱۵: SearchByTextTab بخش چهارم	۲۲
کد ۳-۱۶: SearchByTextTab بخش پنجم	۲۲
کد ۳-۱۷: AdapterTab Class	۲۳
کد ۳-۱۸: SearchByNameStartActivity Class بخش اول	۲۴
کد ۳-۱۹: SearchByNameStartActivity Class بخش دوم	۲۵
کد ۳-۲۰: SearchByNameStartActivity Class بخش سوم	۲۶
کد ۳-۲۱: SearchByName_RecyclerAdapter بخش اول	۲۶
کد ۳-۲۲: SearchByName_RecyclerAdapter بخش دوم	۲۷
کد ۳-۲۳: SearchByName_ViewHolder	۲۸
کد ۳-۲۴: SearchByTextActivityClass بخش اول	۲۹

۳۰	بخش دوم	SearchByTextActivityClass	: ۳-۲۵	کد
۳۰	بخش سوم	SearchByTextActivityClass	: ۳-۲۶	کد
۳۱	بخش چهارم	SearchByTextActivityClass	: ۳-۲۷	کد
۳۱	بخش پنجم	SearchByTextActivityClass	: ۳-۲۸	کد
۳۲	بخش ششم	SearchByTextActivityClass	: ۳-۲۹	کد
۳۳		SearchByText_RecyclerViewAdapter	: ۳-۳۰	کد
۳۴		SearchByText_ViewHolder	: ۳-۳۱	کد
۳۵	بخش اول	ShowFileInfoActivity Class	: ۳-۳۲	کد
۳۶	بخش دوم	ShowFileInfoActivity Class	: ۳-۳۳	کد
۳۶	بخش سوم	ShowFileInfoActivity Class	: ۳-۳۴	کد
۳۷		AskPermissionActivity Class	: ۳-۳۵	کد

فهرست شکل‌ها

عنوان	صفحه
شکل ۳-۱: نمونه خروجی FancyToast	۱۲
شکل ۳-۲: نمونه خروجی UnicornFilePicker	۱۳
شکل ۳-۳: نمونه خروجی MaterialEditText	۱۳
شکل ۳-۴: لیست کلاس‌ها	۱۴
شکل ۳-۵: activity_main.xml SBN	۳۸
شکل ۳-۶: activity_main.xml SBT	۳۸
شکل ۳-۷: fragment_search_by_name_tab.xml	۳۹
شکل ۳-۸: fragment_search_by_text_tab.xml	۴۰
شکل ۳-۹: activity_search_by_name_start.xml	۴۱
شکل ۳-۱۰: search_by_name_viewholder	۴۱
شکل ۳-۱۱: activity_search_by_text_start.xml	۴۲
شکل ۳-۱۲: search_by_text_viewholder.xml	۴۲
شکل ۳-۱۳: activity_show_file_info.xml	۴۳
شکل ۳-۱۴: activity_ask_permission.xml	۴۳
شکل ۴-۱: اسناد اولیه برای بررسی اپلیکیشن	۴۶
شکل ۴-۲: نمونه خروجی ۱	۴۶
شکل ۴-۳: نمونه خروجی ۲	۴۶
شکل ۴-۴: نمونه خروجی ۳	۴۷
شکل ۴-۵: نمونه خروجی ۴	۴۷
شکل ۴-۶: نتیجه جستجو در نرم‌افزار Microsoft Word	۴۷
شکل ۴-۷: نمونه خروجی ۵	۴۷

فهرست علامت‌های اختصاری

عنوان	علامت اختصاری
واحد پردازش مرکزی (Central Processing Unit)	CPU
فرمت سند قابل حمل (Portable Document Format)	PDF
فرمت قدیم سند نوشته‌ها (Word Document Format)	DOC
فرمت جدید سند نوشته‌ها (Word Document Format)	DOCX
سند ساده متن (Text File Format)	TXT
جستجو براساس نام اسناد (Search By Name)	SBN
جستجو براساس پرس‌وجو (Search By Text)	SBT

مقدمه

۱-۱- پیش‌گفتار

اسناد^۱ موجود در دستگاه‌های اندروید معمولاً از اهمیت بالایی برخوردار اند و کاربر به منظور استفاده از این اسناد در آینده، این اسناد را در دستگاه هوشمند خود نگهداری می‌کند. برای بسیاری از کاربران این اتفاق رخ داده است که فراموش می‌کنند سند موردنظر خود را کجا ذخیره کرده‌اند و یا نام سند مورد نظرشان چه بوده است. این اپلیکیشن^۲ به کاربران کمک می‌کند اسناد خود را با استفاده از جستجوی درون تک‌تک سند های موجود در دستگاه بیابند.

۲-۱- هدف از طراحی این اپلیکیشن

هدف اصلی از طراحی اپلیکیشن Goly Document Finder^۳، یافتن اسناد گم‌شده در دستگاه و سرعت بخشیدن به انجام امور کاربر در جهت یافتن داده‌های مورد نیاز او می‌باشد.

۳-۱- ساختار بیان مطالب

۱-۳-۱- بخش‌های اصلی

نحوه پیاده‌سازی برنامه و توسعه این اپلیکیشن در این چند بخش بصورت خلاصه بیان شده است:

۱. یافتن روش حل مسئله برای امتیازدهی و جستجوی اسناد (برنامه‌نویسی جاوا)

۲. یافتن کتابخانه‌های موردنظر برای بهینه‌سازی برنامه

۳. طراحی بخش اندروید: طراحی UI^۴ و کدنویسی اکتیویتی^۵ها

۴. بررسی کارکرد اپلیکیشن و رفع باگ‌های برنامه

^۱ Documents

^۲ Application

^۳ سندیاب گلی

^۴ User Interface

^۵ Activity

۱. یافتن روش حل مسئله برای امتیازدهی و جستجوی اسناد (برنامه‌نویسی جاوا): بررسی الگوریتم‌های موجود برای خواندن اسناد و امتیازدهی مناسب براساس پرس‌وجو^۱ی کاربر.
۲. یافتن کتابخانه‌های موردنظر برای بهینه‌سازی برنامه: یافتن بهینه‌ترین کتابخانه‌ها برای باز کردن اسناد در برنامه‌نویسی جاوا و همچنین پشتیبانی کتابخانه‌ها از زبان فارسی. همچنین یافتن کتابخانه‌ها و وابستگی^۲های مورد نیاز در بخش طراحی اندروید.
۳. طراحی بخش اندروید - طراحی UI و کدنویسی اکتیویتی‌ها: طراحی ظاهری زیبا و کدنویسی بهینه برای برنامه به منظور جلوگیری از کند شدن برنامه و سرعت بالای نمایش نتیجه به کاربر.
۴. بررسی کارکرد اپلیکیشن و رفع باگ‌های برنامه: بررسی نهایی برنامه به منظور یافتن باگ‌های احتمالی و بررسی سرعت حل مسئله.

¹ Query

² Dependency

فصل ۲- روش حل مسئله برای امتیازدهی و جستجوی اسناد

۲-۱- مقدمه

برنامه سندiyاب گلی بطور کلی دو بخش را شامل می‌شود. بخش اول مربوط به جستجوی اسناد تنها با استفاده از نام آن‌ها می‌باشد (SBN^1). این بخش بصورت خودکار در تمامی دستگاه‌های اندروید وجود دارد اما با این تفاوت که تمامی داده‌های دستگاه را بررسی می‌کند. برنامه سندiyاب گلی بسیاری از داده‌ها که سند محسوب نمی‌شوند را نادیده می‌گیرد و این بخش به جستجوی سریع‌تر اسناد کمک می‌کند.

بخش دوم برنامه، جستجوی اسناد با استفاده از پرس‌وجو از کاربر است (SBT^2). بدین صورت که کاربر بخشی از اطلاعات داخل سند مورد نظر خود را وارد می‌کند و برنامه سندiyاب به بررسی داخل اسناد می‌پردازد.

۲-۲- داده‌های مورد نیاز

هر برنامه و اپلیکیشنی برای اجرا نیاز به داده‌های اولیه به عنوان ورودی دارد؛ و پس از انجام پردازش محاسبات بر روی داده‌های اولیه، خروجی خود را به عنوان نتیجه به کاربر تحویل می‌دهد. اپلیکیشن سندiyاب نیز نیاز به داده‌های اولیه برای اجرا شدن دارد که در این بخش بیان شده‌اند.

۲-۲-۱- نام سند (اختیاری)

در صورت به یاد آوردن نام سند توسط کاربر، می‌توان نام سند را بصورت رشته^۳ از کاربر دریافت کرد.

۲-۲-۲- آدرس اولیه

آدرس اولیه که به عنوان آدرس ریشه^۴ نیز شناخته می‌شود، آدرسی است که جستجو در آن انجام می‌شود. این متغیر در برنامه سندiyاب گلی مقدار اولیه دارد ولی می‌تواند از کاربر مقدار جدیدی دریافت کند.

¹ Search By Name

² Search By Text

³ String

⁴ Root

۲-۲-۳- فرمت فایل موردنظر

برنامه سندپاب درحال حاضر تنها سه فرمت را پشتیبانی می‌کند. فرمت پی دی اف^۱، فرمت سندهای مایکروسافت ورد^۲ و فرمت تکست. حداقل یکی از این سه فرمت باید بصورت بولین^۳ مقدار درست داشته باشند.

۲-۲-۴- پرس‌وجو

پرس‌وجو تنها در بخش SBT مورد استفاده قرار می‌گیرد. پرس‌وجو یک متغیر رشته‌ای است که از کاربر دریافت می‌شود و داخل اسناد براساس آن جستجو انجام می‌شود.

۲-۳- جاوا – جستجوی اسناد بر اساس نام

بخش اول برنامه که جستجوی اسناد بر اساس نام یا Search By Name نام دارد، برای جستجوی اسناد براساس نام آن‌ها است و متغیر پرس‌وجو از کاربر دریافت نمی‌شود. کد و نحوه اجرای آن بصورت زیر می‌باشد.

۲-۳-۱- نحوه اجرای Search By Name

نحوه جستجوی اسناد بر اساس نام در متد searchForFilesOnly در کلاس SearchByNameStartActivity انجام می‌شود. کد این متد بصورت زیر است.

¹ PDF

² Microsoft Word

³ Boolean

```

public void searchForFilesOnly
    (File path, String [] name, boolean isWord, boolean isPDF, boolean isTxt) {
    File[] list = path.listFiles();
    boolean ok = true;
    for(int i = 0 ; i<list.length ; i++){

        if ( list[i].isDirectory() ) {
            searchForFilesOnly(list[i],name,isWord,isPDF,isTxt);
        }
        else {
            try{
                if(isWord){
                    String x = list[i].getAbsolutePath().toString().toLowerCase();
                    if(x.endsWith(".doc") || x.endsWith(".docx")) {
                        ok = true;
                        for(int j = 0 ;j<name.length ; j++) if (!x.toLowerCase().contains(name[j])) {ok = false;break;}
                        if(ok)
                        {
                            adptr = (SearchByName_RecyclerAdapter) showFilesRecyclerView.getAdapter();
                            adptr.addFile(list[i]);
                            totalItemsInserted++;
                        }
                    }
                }
                if(isPDF)
                    if(list[i].getAbsolutePath().toString().toLowerCase().endsWith(".pdf")) {
                        ok = true;
                        for(int j = 0 ;j<name.length ; j++) if (!list[i].getAbsolutePath().toString().toLowerCase().contains(name[j]))
                            {ok = false;break;}
                        if(ok)
                        {
                            adptr = (SearchByName_RecyclerAdapter) showFilesRecyclerView.getAdapter();
                            adptr.addFile(list[i]);
                            totalItemsInserted++;
                        }
                    }
                if(isTxt)
                    if(list[i].getAbsolutePath().toString().toLowerCase().endsWith(".txt")){
                        ok = true;
                        for(int j = 0 ;j<name.length ; j++)
                            if (!list[i].getAbsolutePath().toString().toLowerCase().contains(name[j])){ok = false;break;}
                        if(ok)
                        {
                            adptr = (SearchByName_RecyclerAdapter) showFilesRecyclerView.getAdapter();
                            adptr.addFile(list[i]);
                            totalItemsInserted++;
                        }
                    }
            }
            catch(Exception e){
                Log.i( tag: "searchForFilesOnlyClass", msg: "Error in reading files.");
            }
        }
    }
}

```

کد ۱-۲: متد SreachForFilesOnly

بصورت خلاصه، این متد بصورت بازگشتی از آدرس اولیه‌ای که بصورت ورودی به آن داده شده شروع به بررسی تمامی فایل‌ها می‌کند. در صورت پوشه بودن آن فایل، دوباره همین متد را برای آن پوشه صدا می‌زند و در صورتی که پوشه نباشد به بررسی آن براساس ورودی‌ها می‌پردازد. و در انتها لیستی از فایل‌ها که نام آن‌ها مانند نام ورودی است و دارای ویژگی‌های درخواست شده توسط کاربر است را برمی‌گرداند.

۲-۳-۲- خروجی برنامه

خروجی این متد در بخش جاوا در ابتدا آرایه‌ای از فایل‌ها بود که مشخصات ورودی کاربر را داشتند اما پس از وارد کردن کد به اندروید، بنده برای اجرای کد بصورت چندنخی^۱ مجبور به تعریف آرایه‌ای استاتیک^۲ در کد و تبدیل این متد به یک متد بدون خروجی (Void) شدم.

۲-۴-۱- جاوا – جستجوی اسناد بر اساس پرس‌وجو

بخش دوم برنامه که بخش اصلی این اپلیکیشن است، جستجوی اسناد با استفاده از پرس‌وجو می‌باشد. بدین صورت که داخل تمامی اسناد که نام مشابه با نام ورودی کاربر دارند و فرمت درخواستی کاربر است بررسی شده و هر سندی که داده‌های مشابه‌تری با پرس‌وجوی کاربر داشته باشند دارای امتیاز بیشتری می‌شوند و اولویت بالاتری برای نمایش به کاربر دارند.

۲-۴-۱- نحوه اجرای Search By Text

نحوه اجرای الگوریتم SBT، در کلاس startLongSearching که خود در کلاس SearchByTextStartActivity وجود دارد انجام می‌شود. در ابتدا برنامه به جستجوی اسناد در بخش مورد نظر مشغول می‌شود و پس از یافتن اسناد با مشخصات ورودی کاربر، شروع به سرچ عمیق در آن‌ها بصورت Background می‌شود (بدین صورت که نخ اصلی را درگیر نمی‌کند و باعث کندی دستگاه اندرویدی نمی‌شود).

¹ Thread

² Static Array

کدهای این بخش بصورت زیر می باشد

```
@Override
protected Void doInBackground(Void... voids) {
    myFiles = searchForFilesOnly(path,nameArray,word,pdf,txt);
    PB.setMax(myFiles.size());
    try {
        query = query.toLowerCase();
        Scanner input;
        for(int i = 0 ; i<myFiles.size() ; i++){
            try {
                boolean newFound;
                boolean found = false;
                int score = 0;
                int scoreMultiple = 0;
                publishProgress( _values: i, 0);
                input = new Scanner(loadString(myFiles.get(i)));
                //Give Score to each file:
                while (input.hasNext()) {
                    newFound = false;
                    String wordToSearch = input.next().toLowerCase();
                    for (int j = 0; j < queryArray.length; j++) {
                        if (wordToSearch.contains(queryArray[j])) {
                            newFound = true;
                            if (found) score = score + (scoreMultiple);
                            else score = score + 1;
                            break;
                        }
                    }
                    if (!newFound) {
                        found = false;
                        scoreMultiple = 0;
                    } else {
                        found = true;
                        scoreMultiple += 4;
                    }
                }
                if (score > 0) {
                    myData.path.add(myFiles.get(i));
                    myData.score.add(score);
                }
            } catch (Exception e){...}
        }
    } catch (Exception e) {...}
    publishProgress( _values: 0,1);
    SortMyData();
    publishProgress( _values: 0,2);
    return null;
}
```

کد ۲-۲: متد doInBackground از کلاس startLongSreaching

۲-۴-۲- نحوه محاسبه امتیاز اسناد

محاسبه امتیاز هر سند بدین صورت انجام می‌شود:

۱. تکرار هر کلمه از پرس‌وجوی کاربر یک امتیاز دارد.
 ۲. در صورتی که کلمات پرس‌وجوی کاربر در سند مورد نظر پشت سر هم باشند، ۱۰ امتیاز به آن سند اضافه می‌شود.
 ۳. بررسی حروف اول کلمات و در صورت شبیه بودن ۸۰ درصد از حروف کلمه ای با یکی از کلمات پرس‌وجو، نیم امتیاز به آن سند تعلق می‌گیرد. (این بخش همچنان در حال توسعه است و در نسخه اصلی وجود ندارد).
- در آخر اسنادی که امتیازی بدست نیاورده‌اند در کل به لیست نهایی اضافه نمی‌شوند و به کاربر نمایش داده نمی‌شوند.

۲-۴-۳- کلاس WordSearchData

این کلاس برای بخش SBT طراحی شده است. از آنجایی که برای هر سند باید امتیاز آن سند را نیز ذخیره کرد، این کلاس به تنهایی برای هر سند امتیازی را نیز در نظر می‌گیرد. امتیاز هر سند متناسب با اندیس آن سند ذخیره شده است.

```
package com.goly.golydocumentfinder;

import ...

public class WordSearchData {
    public ArrayList<File> path;
    public ArrayList<Integer> score;

    public WordSearchData() {
        this.path = new ArrayList<>();
        this.score = new ArrayList<>();
    }

    public ArrayList<File> getPath() { return path; }
    public ArrayList<Integer> getScore() { return score; }
    public void setPath(ArrayList<File> path) { this.path = path; }
    public void setScore(ArrayList<Integer> score) { this.score = score; }
}
```

کد ۲-۳: کلاس WordSearchData

۲-۴-۴- خروجی برنامه

در آخر، پس از بررسی اسناد متناسب با ورودی کاربر، اسنادی که دارای امتیاز بودند بصورت آرایه ذخیره شده و پس از به ترتیب قرار دادن آنها در همان آرایه، به کاربر نمایش داده می شوند.

فصل ۳ - پیاده‌سازی کدهای اندروید و تعریف اکتیویتی‌ها

۳-۱- مقدمه

پس از نوشتن کدهای روش حل مسئله برای امتیازدهی و جستجوی اسناد، حال باید به ساخت طراحی ظاهری برنامه در بخش اندروید با استفاده از نرم‌افزار اندروید استدیو^۱ پردازیم.

۳-۲- دسترسی‌ها و حداقل نسخه اندروید مورد نیاز

دسترسی^۲‌ها مواردی هستند که در ابتدای برنامه در بخش AndroidManifest.xml تعریف می‌شوند و به منظور آگاه‌سازی دستگاه اندروید کاربر برای دسترسی به بخش‌های مختلف دستگاه بیان می‌شوند. برنامه سندیاب تنها نیاز به دسترسی خواندن داده‌ها و اطلاعات حافظه دستگاه دارد تا بتواند پرس‌وجو را انجام دهد. این دسترسی در بخش AndroidManifest.xml بصورت زیر نوشته می‌شود.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.goly.golydocumentfinder">

    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

کد ۳-۱: دسترسی خواندن حافظه

این اپلیکیشن حداقل نیازمند به اندروید نسخه ۵ برای اجرا می‌باشد.

۳-۳- مفاهیم اولیه در برنامه‌نویسی اندروید

برای درک مفهوم بهتر برنامه‌نویسی اندروید، مفاهیم اولیه زیر برای درک بهتر نحوه عملکرد کدهای اندروید طراحی شده‌اند:

- View: به هریک از اجزاء موجود در بخش طراحی چیدمان، یک ویو^۳ گفته می‌شود.
- TextView: یکی ویو که وظیفه آن نمایش نوشته‌ها و متون می‌باشد.
- EditText: یک ویو که وظیفه آن دریافت متن از کاربر می‌باشد.

^۱ Android Studio

^۲ Permission

^۳ View

- **RecyclerView**: یک کلاس قدرتمند در اندروید جهت نمایش اطلاعات بصورت خطی. به عنوان مثال در این برنامه، پس از انجام جستجو باید تعدادی سند را به کاربر نمایش داد. اگر تعداد اسناد بالاتر از ۱۰۰۰ باشد هیچ مدل نمایشی نمی‌تواند به راحتی آن را اداره کند و به کاربر نمایش دهد. تفاوت **RecyclerView** با دیگر روش‌های نمایش این است که به عنوان مثال از ۱۰۰۰ سند یافت شده تنها تعدادی را بر روی حافظه موقت^۱ بارگذاری می‌کند که بر روی صفحه نمایش دستگاه قابل نمایش هستند. برای استفاده از این کلاس، باید از دو کلاس **ViewHolder** و **RecyclerViewAdapter** نیز استفاده کرد که به ترتیب هدف آن‌ها، ساخت ظاهر هر سطر و اداره کلی یک **RecyclerView** می‌باشد.
- **ImageView**: یک ویو که وظیفه آن نمایش تصاویر در اندروید می‌باشد.
- تابع **onCreate()**: اولین تابعی که پس از اجرای یک اکتیویتی اجرا می‌شود. اگر چه توابع دیگری نیز قبل از این تابع اجرا می‌شوند ولی این تابع بهترین مکان برای تعریف و ساخت متغیرها می‌باشد.
- تابع **onCreateView()**: اولین تابعی که پس از اجرای یک فرگمنت اجرا می‌شود. اگر چه توابع دیگری نیز قبل از این تابع اجرا می‌شوند ولی این تابع بهترین مکان برای تعریف و ساخت متغیرها می‌باشد.
- **ProgressBar**: یک ویو که وظیفه آن نمایش وضعیت کنونی پردازش در حال انجام می‌باشد. این ویو به منظور اطلاع‌رسانی به کاربر برای مشاهده زمان مورد نیاز برای انجام یک پردازش طراحی شده است.
- **AsyncTask**: یک کلاس پیش‌فرض در اندروید که دیگر کلاس‌ها می‌توانند از این کلاس گسترش یابند. هدف این کلاس اجرای توابع سنگین بصورت پس‌زمینه و بر روی نخ‌های پس‌زمینه است و باعث می‌شود برنامه در هنگام اجرای فعالیت‌های سنگین دچار مشکل نشود.

۳-۴ - کتابخانه‌ها و وابستگی‌های استفاده شده

به منظور ساخت یک اپلیکیشن کامل، مجموعه‌ای از وابستگی‌ها در بخش اندروید و کتابخانه‌ها در بخش جاوا به برنامه اضافه شده‌اند که می‌توان به موارد زیر اشاره کرد.

^۱ RAM (Random Access Memory)

^۲ Thread

۳-۴-۱- Apache POI

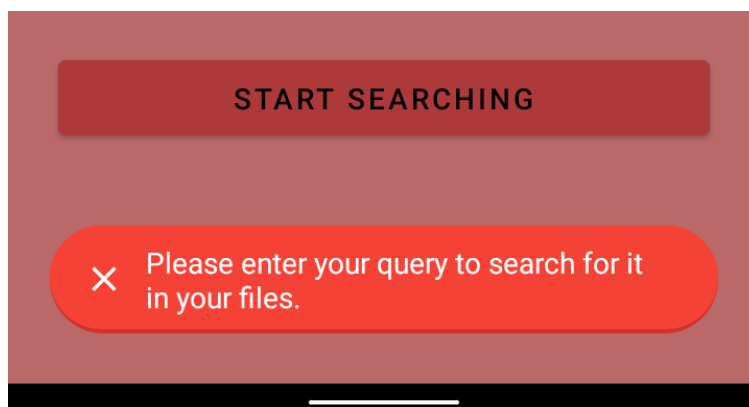
کتابخانه Apache POI به منظور کارکردن با داده‌ها و اسناد مایکروسافت^۱ طراحی شده است. در این برنامه برای خواندن اسناد Microsoft Word با فرمت‌های Doc و Docx استفاده می‌شود.

۳-۴-۲- Apache Pdfbox

کتابخانه Apache Pdfbox به منظور کارکردن با اسناد پی‌دی‌اف طراحی و ساخته شده است و یک کتابخانه بسیار سریع جهت خواندن از اسناد پی‌دی‌اف محسوب می‌شود. در این برنامه از این کتابخانه به منظور استخراج متون اسناد پی‌دی‌اف استفاده می‌شود.

۳-۴-۳- FancyToast Dependency

وابستگی FancyToast یک وابستگی در بخش اندروید به منظور طراحی ظاهری بهتر هنگام نمایش Toast Message استفاده می‌شود. به عنوان مثال در صورتی که کاربر پرس‌وجو را وارد نکرده باشد و گزینه شروع جستجو را انتخاب کند نمونه خروجی زیر را با ظاهری متفاوت ایجاد می‌کند.

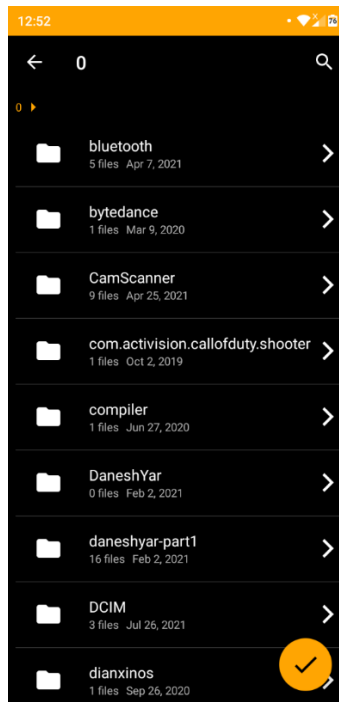


شکل ۱-۳: نمونه خروجی FancyToast

۳-۴-۴- UnicornFilePicker Dependency

وابستگی UnicornFilePicker یک وابستگی در بخش اندروید است که به ما کمک می‌کند مسیر و آدرس اولیه برای شروع جستجو را از کاربر دریافت کنیم. این وابستگی هنگامی اجرا می‌شود که کاربر گزینه Directory را انتخاب کند. سپس پنجره‌ای بصورت زیر باز می‌شود و از کاربر آدرس و مسیر جدید را می‌گیرد.

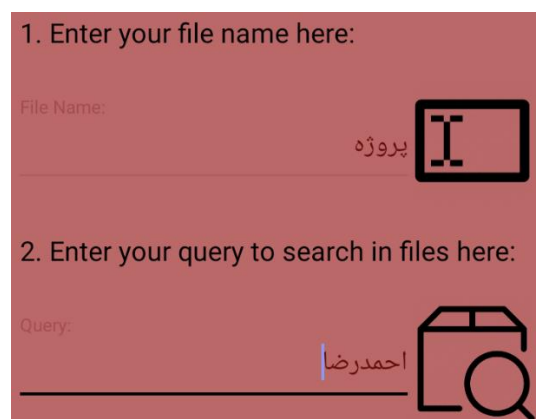
¹ Microsoft



شکل ۲-۳: نمونه خروجی UnicornFilePicker

۳-۴-۵- MaterialEditText Dependency

وابستگی MaterialEditText یک وابستگی در بخش اندروید است که باعث می‌شود فیلدهایی که کاربر مشغول به پر کردن آن است دارای محیط کاربری بهتر و طراحی و انیمیشن‌های زیباتری باشد. نمونه خروجی آن بصورت زیر می‌باشد.



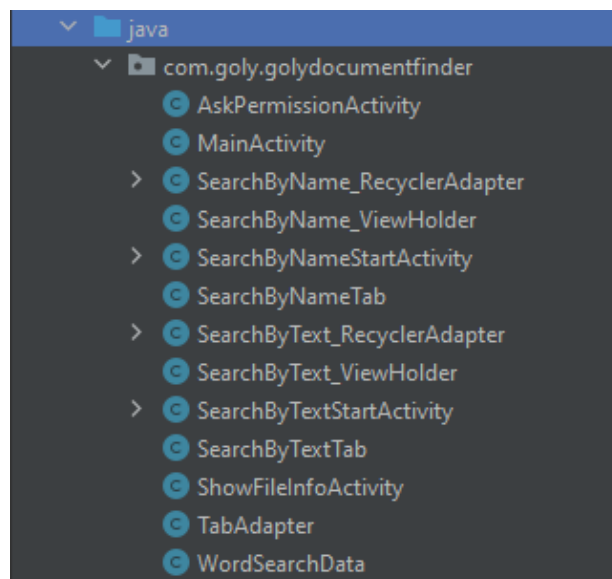
شکل ۳-۳: نمونه خروجی MaterialEditText

۳-۵- اکتیویته‌ها و کلاس‌ها

اکتیویته^۱ها در اندروید بخش اصلی برنامه را شامل می‌شوند و وابستگی بین کد و رابط کاربری در بخش اکتیویته‌ها وجود دارد. تمامی کلاس‌ها و بسیاری از کدهای جاوا در این بخش نوشته شده تا برنامه بصورت

¹ Activity

یکنواخت و بدون مشکل اجرا شود. لیست کل کلاس‌های تعریف شده در این برنامه در تصویر زیر مشخص شده‌اند.



شکل ۴-۳: لیست کلاس‌ها

۳-۵-۱- MainActivity Class

کلاس MainActivity که اکتیویته برای چیدمان activity_main.xml محسوب می‌شود، اولین کلاسی است که هنگام اجرای برنامه اجرا می‌شود و چیدمان آن به کاربر نمایش داده می‌شود.

۳-۵-۱- ساختار کد

در ابتدا چهار متغیر بصورت زیر تعریف می‌شوند.

```
package com.goly.golydocumentfinder;

import ...

public class MainActivity extends AppCompatActivity {
    private Toolbar toolbar;
    private ViewPager viewPager;
    private TabLayout tabLayout;
    private TabAdapter tabAdapter;
```

کد ۲-۳: MainActivity Class بخش اول

تمامی این متغیرها در هدف تشکیل یک صفحه که دارای دو بخش Search By Text و Search By Name است ساخته شده و بصورت زیر تعریف می‌شوند. نمونه خروجی آن در بخش ۳-۱-۵ وجود دارد. پس از تعریف این دو بخش، کدهای هر بخش باید بصورت فرگمنت^۱ تعریف شوند.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    setTitle("Goly Document Finder");
    toolbar = findViewById(R.id.myToolbar);
    setSupportActionBar(toolbar);
    viewPager = findViewById(R.id.viewPager);
    tabAdapter = new TabAdapter(getSupportFragmentManager());
    viewPager.setAdapter(tabAdapter);
    tabLayout = findViewById(R.id.tabLayout);
    tabLayout.setupWithViewPager(viewPager, autoRefresh: false);
}
```

کد ۳-۳: MainActivity Class بخش دوم

پس از تعریف این چهار متغیر، برای متغیر viewPager یک onPageChangeListener اضافه می‌کنیم. این کار باعث می‌شود هر موقع کاربر بین دو صفحه Search By Text و Search By Name جابجا شد، این بخش از کد اجرا شود. هدف بنده از این کار تنها تغییر رنگ سربزرگ به آبی یا قرمز می‌باشد.

```
viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
    // This method will be invoked when a new page becomes selected.
    @Override
    public void onPageSelected(int position) {
        if(position==0)
            getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.parseColor( colorString: "#374CBF")));
        else
            getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.parseColor( colorString: "#BF3737")));
    }
    //Useless but should be implemented:
    @Override
    public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {
    }
    @Override
    public void onPageScrollStateChanged(int state) {
    }
});
```

کد ۳-۴: MainActivity Class – addOnPageChangeListener

در انتها، پس از هر دفعه اجرای برنامه، باید بررسی شود که آیا این برنامه دسترسی به خواندن حافظه را قبلاً از کاربر دریافت کرده است یا خیر. اگر دسترسی به حافظه نداشته باشد در عمل به مشکل برمی‌خورد

¹ Fragment

و برنامه کار نمی‌کند. در نتیجه دسترسی برنامه بررسی شده و در صورت نداشتن اجازه کاربر، برنامه کاربر را به صفحه دریافت اجازه خواندن حافظه (AskPermissionActivity.class) هدایت می‌کند.

```
if (Build.VERSION.SDK_INT >= 23 && //android marshmello. if it goes under 23 we don't need permission
    (
        ActivityCompat.checkSelfPermission( context: MainActivity.this, Manifest.permission.READ_EXTERNAL_STORAGE)
        != PackageManager.PERMISSION_GRANTED ||
        ActivityCompat.checkSelfPermission( context: MainActivity.this, Manifest.permission.WRITE_EXTERNAL_STORAGE)
        != PackageManager.PERMISSION_GRANTED ) ){
    Intent intent = new Intent( packageContext: this, AskPermissionActivity.class);
    startActivity(intent);
    finish();
}
```

کد ۵-۳: MainActivityClass - بررسی دسترسی خواندن حافظه

۳-۵-۱-۲ SearchByNameTab Fragment

صفحه اول در کلاس MainActivity، Search By Name نام دارد. این صفحه بصورت فرگمنت در کلاس SearchByNameTab که به چیدمان fragment_search_by_name_tab.xml وابسته است تعریف شده است. تنها وظیفه این فرگمنت، دریافت تمامی داده‌های لازم از کاربر برای پاس دادن این داده‌ها به اکتیویتی SearchByNameStart به منظور شروع جستجو بر اساس نام است. کد این فرگمنت بطور خلاصه بصورت زیر می‌باشد.

در ابتدا، متغیرهای اصلی بصورت زیر تعریف می‌شوند. متغیرهای Button در بخش چیدمان همان دکمه‌های انتخاب آدرس جستجو و شروع جستجو می‌باشند. متغیر TextView برای نمایش یک نوشته استفاده می‌شود. متغیر EditText به منظور گرفتن یک نوشته از کاربر استفاده می‌شوند و در بخش چیدمان همان فیلدهای خالی هستند. متغیرهای Switch کلیدهایی هستند که تنها مقدار ۱ یا مقدار ۰ را از کاربر می‌گیرند.

```
package com.goly.golydocumentfinder;

import ...

public class SearchByNameTab extends Fragment implements View.OnClickListener {
    private Button directoryButton, startSearchingButton;
    private TextView directoryTextView;
    private EditText fileNameEdt;
    private Switch pdf, word, txt;
    private LinearLayout LL;
    String directory;
```

کد ۶-۳: SearchByNameTab بخش اول

¹ Field

سپس توابع اصلی کلاس فرگمنت بصورت خودکار تعریف می‌شوند.

```
public SearchByNameTab() {  
    // Required empty public constructor  
}  
  
public static SearchByNameTab newInstance(String param1, String param2) {  
    SearchByNameTab fragment = new SearchByNameTab();  
    Bundle args = new Bundle();  
    fragment.setArguments(args);  
    return fragment;  
}  
  
@Override  
public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); }
```

کد ۷-۳: SearchByNameTab بخش دوم

پس از آن تمامی متغیرها در تابع onCreateView به چیدمان متصل می‌شوند. برای دو متغیر directoryButton و startSearchingButton که هر دو دکمه هستند یک OnClickListener تعریف می‌شود که در ادامه به تعریف آن‌ها می‌پردازیم. سپس تابع SetupUI اجرا می‌شود که هدف این تابع، بسته شدن صفحه کلید دستگاه اندروید کاربر در صورت ضربه زدن به هرجایی بجز فیلدها است. در نهایت این View به Tab ساخته شده در MainActivity برگردانده می‌شود.

```
@Override  
public View onCreateView(LayoutInflater inflater, ViewGroup container,  
    Bundle savedInstanceState) {  
    View myview = inflater.inflate(R.layout.fragment_search_by_name_tab, container, attachToRoot: false);  
    directory = Environment.getExternalStorageDirectory().getAbsolutePath();  
    directoryButton = myview.findViewById(R.id.SBN_selectPathButton);  
    startSearchingButton = myview.findViewById(R.id.SBN_startSearchingButton);  
    directoryTextView = myview.findViewById(R.id.SBN_selectPathTextView);  
    directoryTextView.setText(directory);  
    fileNameEdt = myview.findViewById(R.id.SBN_fileNameEdt);  
    pdf = myview.findViewById(R.id.SBN_PDFSwitch);  
    word = myview.findViewById(R.id.SBN_WordSwitch);  
    txt = myview.findViewById(R.id.SBN_TextSwitch);  
    LL = myview.findViewById(R.id.SBN_LinearLayout);  
  
    directoryButton.setOnClickListener(this);  
    startSearchingButton.setOnClickListener(this);  
    setupUI(LL);  
    return myview;  
}
```

کد ۸-۳: SearchByNameTab بخش سوم

حال به تعریف دکمه‌ها می‌پردازیم. دکمه directoryButton وظیفه دارد آدرس مکان جستجو را از کاربر دریافت کند. این آدرس باید با استفاده از وابستگی UnicornFilePicker بدست آید. مشکلی که در این بخش وجود دارد تفاوت آدرس روت در اندرویدهای نسخه بعد از R و قبل از R می‌باشد. پس نیاز به چند

شرط بررسی دارد که بدون مشکل آدرس را از کاربر دریافت کند. دکمه `directoryButton` بصورت زیر تعریف می‌شود.

```
@Override
public void onClick(View view) {
    switch(view.getId()) {
        case R.id.SBN_selectPathButton:
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
                UnicornFilePicker.from(SearchByNameTab.this) UnicornFilePicker
                    .addConfigBuilder() ConfigBuilder
                    .selectMultipleFiles(false)
                    .showOnlyDirectory(true)
                    .setRootDirectory(Environment.getExternalStorageDirectory().getAbsolutePath())
                    .showHiddenFiles(true)
                    .addItemDivider(true)
                    .theme(R.style.UnicornFilePicker_Dracula)
                    .build() UnicornFilePicker
                    .forResult(Constants.REQ_UNICORN_FILE);
            }
            else new AlertDialog.Builder(getContext())
                .setMessage("Choose your directory:")

                .setPositiveButton(text: "External", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        UnicornFilePicker.from(SearchByNameTab.this) UnicornFilePicker
                            .addConfigBuilder() ConfigBuilder
                            .selectMultipleFiles(false)
                            .showOnlyDirectory(true)
                            .setRootDirectory("storage/")
                            .showHiddenFiles(true)
                            .addItemDivider(true)
                            .theme(R.style.UnicornFilePicker_Dracula)
                            .build() UnicornFilePicker
                            .forResult(Constants.REQ_UNICORN_FILE);
                    }
                })
                .setNegativeButton(text: "Internal", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        UnicornFilePicker.from(SearchByNameTab.this) UnicornFilePicker
                            .addConfigBuilder() ConfigBuilder
                            .selectMultipleFiles(false)
                            .showOnlyDirectory(true)
                            .setRootDirectory(Environment.getExternalStorageDirectory().getAbsolutePath())
                            .showHiddenFiles(true)
                            .addItemDivider(true)
                            .theme(R.style.UnicornFilePicker_Dracula)
                            .build() UnicornFilePicker
                            .forResult(Constants.REQ_UNICORN_FILE);
                    }
                })
                .show();
            break;
    }
}
```

کد ۹-۳: SearchByNameTab بخش چهارم

سپس دکمه `startSearchingButton` بصورت زیر تعریف می‌شود. پس از زدن بر روی این دکمه، ابتدا باید بررسی شود که آیا فیلدهای لازم از کاربر گرفته شده است یا خیر. پس از آن می‌توان تمامی متغیرها را داخل یک متغیر `Instant` گذاشت و برای اکتیویتی `SearchByNameStartActivity` ارسال کرد. همانگونه که مشاهده می‌شود، وابستگی `FancyToast` نیز در این بخش استفاده شده است. این دکمه بصورت زیر تعریف می‌شود.

```

case R.id.SBN_startSearchingButton:
    if(directory.equals(""))
        FancyToast.makeText(getContext(), message: "Please select a path to search.",
            Toast.LENGTH_SHORT, FancyToast.ERROR, androidIcon: false).show();

    else if(!pdf.isChecked() && !word.isChecked() && !txt.isChecked() )
        FancyToast.makeText(getContext(), message: "Please turn on one of the switches to search for Pdf or Word or Text Files.",
            Toast.LENGTH_SHORT, FancyToast.ERROR, androidIcon: false).show();
    else{
        Intent i = new Intent(getContext(), SearchByNameStartActivity.class);
        i.putExtra( name: "filename", fileNameEdt.getText().toString());
        i.putExtra( name: "path", directory);
        i.putExtra( name: "pdf", pdf.isChecked());
        i.putExtra( name: "word", word.isChecked());
        i.putExtra( name: "txt", txt.isChecked());
        startActivity(i);
    }
    break;

```

کد ۱۰-۳: SearchByNameTab بخش پنجم

توابع hideSoftKeyboard و SetupUI برای بستن کیبورد پس از ضربه زدن کاربر به هرجایی بجز فیلدهای درخواستی از کاربر می‌باشد. کد این دو تابع بصورت زیر نوشته شده است.

```

public static void hideSoftKeyboard(Activity activity) {
    try {
        InputMethodManager inputMethodManager =
            (InputMethodManager) activity.getSystemService(
                Activity.INPUT_METHOD_SERVICE);
        if (inputMethodManager.isAcceptingText()) {
            inputMethodManager.hideSoftInputFromWindow(
                activity.getCurrentFocus().getWindowToken(),
                flags: 0
            );
        }
    } catch (Exception e){
        Log.i( tag: "hideSoftKeyBoard", msg: "Can't hide keyboard!");
    }
}

public void setupUI(View view) {

    // Set up touch listener for non-text box views to hide keyboard.
    if (!(view instanceof EditText)) {
        view.setOnTouchListener(new View.OnTouchListener() {
            public boolean onTouch(View v, MotionEvent event) {
                hideSoftKeyboard(getActivity());
                return false;
            }
        });
    }

    //If a layout container, iterate over children and seed recursion.
    if (view instanceof ViewGroup) {
        for (int i = 0; i < ((ViewGroup) view).getChildCount(); i++) {
            View innerView = ((ViewGroup) view).getChildAt(i);
            setupUI(innerView);
        }
    }
}

```

کد ۱۱-۳: SearchByNameTab بخش ششم

SearchByTextTab Fragment ۳-۱-۵-۳

صفحه دوم در کلاس MainActivity، Search By Text نام دارد. این صفحه نیز بصورت فرگمنت در کلاس SearchByTextTab که به چیدمان fragment_search_by_Text_tab.xml وابسته است تعریف شده است. وظیفه این فرگمنت تقریباً همانند فرگمنت SearchByNameTab، دریافت تمامی داده‌های لازم از کاربر برای پاس دادن این داده‌ها به اکتیویتی SearchByTextStart به منظور شروع جستجو بر اساس پرس‌وجو است. کد این فرگمنت بطور خلاصه بصورت زیر می‌باشد.

مشابه SearchByNameTab تمامی متغیرها در ابتدای کلاس تعریف می‌شوند و توابع اولیه فرگمنت نیز بصورت پیش‌فرض وجود دارند.

```
package com.goly.golydocumentfinder;

import ...

public class SearchByTextTab extends Fragment implements View.OnClickListener{
    private Button directoryButton, startSearchingButton;
    private TextView directoryTextView;
    private EditText fileNameEdt, queryEdt;
    private Switch pdf, word, txt;
    String directory;
    public SearchByTextTab() {

    }

    public static SearchByTextTab newInstance(String param1, String param2) {...}

    @Override
    public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); }
```

کد ۳-۱۲: SearchByTextTab بخش اول

سپس تمامی متغیرها تعریف شده و همانند SearchByNameTab برای دو دکمه موجود onClickListner باید تعریف شود. و در آخر تابع صفحه کلید صدا زده می‌شود.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    View myview = inflater.inflate(R.layout.fragment_search_by_text_tab, container, attachToRoot: false);
    directory = Environment.getExternalStorageDirectory().getAbsolutePath();
    directoryButton = myview.findViewById(R.id.SBT_selectPathButton);
    startSearchingButton = myview.findViewById(R.id.SBT_startSearchingButton);
    directoryTextView = myview.findViewById(R.id.SBT_selectPathTextView);
    directoryTextView.setText(directory);
    fileNameEdt = myview.findViewById(R.id.SBT_fileNameEdt);
    queryEdt = myview.findViewById(R.id.SBT_queryEdt);
    pdf = myview.findViewById(R.id.SBT_PDFSwitch);
    word = myview.findViewById(R.id.SBT_WordSwitch);
    txt = myview.findViewById(R.id.SBT_TextSwitch);

    directoryButton.setOnClickListener(this);
    startSearchingButton.setOnClickListener(this);
    setUpUI(myview.findViewById(R.id.SBT_LinearLayout));
    return myview;
}
```

کد ۳-۱۳: SearchByTextTab بخش دوم

سپس دکمه‌ها یکی یکی تعریف می‌شوند.

```
@Override
public void onClick(View view) {
    switch(view.getId()) {
        case R.id.SBT_selectPathButton:
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
                UnicornFilePicker.from(SearchByTextTab.this) UnicornFilePicker
                    .addConfigBuilder() ConfigBuilder
                    .selectMultipleFiles(false)
                    .showOnlyDirectory(true)
                    .setRootDirectory(Environment.getExternalStorageDirectory().getAbsolutePath())
                    .showHiddenFiles(true)
                    .addItemDivider(true)
                    .theme(R.style.UnicornFilePicker_Dracula)
                    .build() UnicornFilePicker
                    .forResult(Constants.REQ_UNICORN_FILE);
            }
            else
            {
                new AlertDialog.Builder(getContext())
                    .setMessage("Choose your directory:")

                    .setPositiveButton( text: "External", new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int which) {
                            UnicornFilePicker.from(SearchByTextTab.this) UnicornFilePicker
                                .addConfigBuilder() ConfigBuilder
                                .selectMultipleFiles(false)
                                .showOnlyDirectory(true)
                                .setRootDirectory("storage/")
                                .showHiddenFiles(true)
                                .addItemDivider(true)
                                .theme(R.style.UnicornFilePicker_Dracula)
                                .build() UnicornFilePicker
                                .forResult(Constants.REQ_UNICORN_FILE);
                        }
                    })
                    .setNegativeButton( text: "Internal", new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int which) {
                            UnicornFilePicker.from(SearchByTextTab.this) UnicornFilePicker
                                .addConfigBuilder() ConfigBuilder
                                .selectMultipleFiles(false)
                                .showOnlyDirectory(true)
                                .setRootDirectory(Environment.getExternalStorageDirectory().getAbsolutePath())
                                .showHiddenFiles(true)
                                .addItemDivider(true)
                                .theme(R.style.UnicornFilePicker_Dracula)
                                .build() UnicornFilePicker
                                .forResult(Constants.REQ_UNICORN_FILE);
                        }
                    })
                    .show();
            }
        }
    }
    break;
```

کد ۱۴-۳: SearchByTextTab بخش سوم

دکمه شروع جستجو نیز به همین صورت تعریف می شود.

```
case R.id.SBT_startSearchingButton:
    if(directory.equals(""))
        FancyToast.makeText(getContext(), message: "Please select a path to search.",
            Toast.LENGTH_SHORT, FancyToast.ERROR, androidicon: false).show();

    else if(!pdf.isChecked() && !word.isChecked() && !txt.isChecked() )
        FancyToast.makeText(getContext(), message: "Please turn on one of the switches to search for Pdf or Word or Text Files.",
            Toast.LENGTH_SHORT, FancyToast.ERROR, androidicon: false).show();
    else if(queryEdt.getText().toString().equals(""))FancyToast.makeText(getContext(), message: "Please enter your query to search for it in your files.",
        Toast.LENGTH_SHORT, FancyToast.ERROR, androidicon: false).show();
    else{
        Intent i = new Intent(getContext(), SearchByTextStartActivity.class);
        i.putExtra( name: "filename",fileNameEdt.getText().toString());
        i.putExtra( name: "query",queryEdt.getText().toString());
        i.putExtra( name: "path",directory);
        i.putExtra( name: "pdf",pdf.isChecked());
        i.putExtra( name: "word",word.isChecked());
        i.putExtra( name: "txt",txt.isChecked());
        startActivity(i);
    }
    break;
}
```

کد ۱۵-۳: SearchByTextTab بخش چهارم

در اخر توابع لازم برای بسته شدن صفحه کلید تعریف می شوند.

```
public static void hideSoftKeyboard(Activity activity) {
    try {
        InputMethodManager inputMethodManager =
            (InputMethodManager) activity.getSystemService(
                Activity.INPUT_METHOD_SERVICE);

        if (inputMethodManager.isAcceptingText()) {
            inputMethodManager.hideSoftInputFromWindow(
                activity.getCurrentFocus().getWindowToken(),
                flags: 0
            );
        }
    }catch(Exception e){
        Log.i( tag: "hideSoftKeyBoard", msg: "Can't hide keyboard!");
    }
}

public void setupUI(View view) {

    // Set up touch listener for non-text box views to hide keyboard.
    if (!(view instanceof EditText)) {
        view.setOnTouchListener(new View.OnTouchListener() {
            public boolean onTouch(View v, MotionEvent event) {
                hideSoftKeyboard(getActivity());
                return false;
            }
        });
    }

    //If a layout container, iterate over children and seed recursion.
    if (view instanceof ViewGroup) {
        for (int i = 0; i < ((ViewGroup) view).getChildCount(); i++) {
            View innerView = ((ViewGroup) view).getChildAt(i);
            setupUI(innerView);
        }
    }
}
```

کد ۱۶-۳: SearchByTextTab بخش پنجم

این کلاس بسیار شبیه به کلاس SearchByNameTab می باشد. با این تفاوت که باید پرس و جو را نیز از کاربر دریافت کند.

AdapterTab Class - ۳-۵-۱-۴

کلاس AdapterTab وظیفه کنترل دو فرگمنت SearchByNameTab و SearchByTextTab را برعهده دارد و واسطه این دو فرگمنت با اکتیویتی MainActivity می‌باشد. این کلاس در اکتیویتی MainActivity ساخته شد و مورد استفاده قرار گرفت. کد این کلاس بصورت زیر می‌باشد.

```
package com.goly.golydocumentfinder;

import ...

public class TabAdapter extends FragmentPagerAdapter {

    public TabAdapter(@NonNull FragmentManager fm) { super(fm); }

    @NonNull
    @Override
    public Fragment getItem(int position) {
        switch(position){
            case 0:
                return new SearchByNameTab();
            case 1:
                return new SearchByTextTab();
            default: return null;
        }
    }

    @Override
    public int getCount() { return 2; }

    @Nullable
    @Override
    public CharSequence getPageTitle(int position) {
        switch (position){
            case 0:
                return "Search by name";
            case 1:
                return "Search in all files";
            default:
                return "Goly Document Finder";
        }
    }
}
```

کد ۳-۱۷: AdapterTab Class

SearchByNameStartActivity - ۲-۵-۳

ساختار کد - ۱-۲-۵-۳

اکتیویته `SearchByNameStartActivity` هنگامی اجرا می‌شود که تمامی داده‌های ورودی از صفحه `SearchByNameTab` دریافت شده باشند. حال باید با این داده‌ها شروع به جستجو بر اساس نام اسناد کرد. کد این بخش از برنامه بصورت زیر طراحی شده است.

ابتدا متغیرهای لازم تعریف می‌شوند. ۵ متغیر اول (`name`, `path`, `pdf`, `word`, `txt`) متغیرهایی هستند که از فرگمنت `SearchByNameTab` دریافت شده‌اند و در تابع `onCreate` باید مقداردهی شوند. متغیر `nameArray` همان `name` است با این تفاوت که براساس فاصله‌هایی که در متغیر `name` وجود دارد تبدیل به چند کلمه شده‌اند. متغیر اصلی این کلاس، `showFilesRecyclerView` می‌باشد. این متغیر وظیفه دارد داده‌ها را بصورت `RecyclerView` و بصورت بهینه به کاربر نمایش دهد. دو کلاس `ViewHolder` و `RecyclerViewAdapter` نیز برای همین مدل نمایش به برنامه اضافه شده‌اند. سه متغیر آخر نیز تنها برای محاسبات و ذخیره موقت در همین کلاس طراحی شده‌اند. پس از آن تابع `fileIsClicked` تعریف می‌شود که این تابع یک `Interface` از کلاس `SearchByName_RecyclerViewAdapter` می‌باشد و به منظور نمایش اطلاعات اسناد نهایی پس از کلیک کردن بر روی آن‌ها تعریف شده است.

```
package com.goly.golydocumentfinder;

import ...

public class SearchByNameStartActivity extends AppCompatActivity implements SearchByName_RecyclerViewAdapter.fileIsClickedInterface {

    private String name;
    private File path;
    private boolean pdf;
    private boolean word;
    private boolean txt;
    private String [] nameArray;
    private RecyclerView showFilesRecyclerView;
    TextView cacheLoad;
    SearchByName_RecyclerViewAdapter adptr;
    static int totalItemsInserted;
    @Override
    public void fileIsClicked(File file) {
        Intent i = new Intent( packageContext SearchByNameStartActivity.this, ShowFileInfoActivity.class);
        i.putExtra( name: "showfile", file.toString());
        startActivity(i);
    }
}
```

کد ۱۸-۳: `SearchByNameStartActivity` Class بخش اول

سپس در تابع `onCreate` متغیرها با استفاده از `Intent` مقداردهی می‌شوند. تمامی این داده‌ها از فرگمنت قبلی (`SearchByNameTab`) دریافت شده است. بقیه متغیرها نیز بصورت کامل تعریف می‌شوند و سپس کلاس `startSearchingForFilesOnly` که در همین کلاس تعریف شده است بصورت یک نخ پس‌زمینه اجرا می‌شود. (execute)

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_search_by_name_start);

    setTitle("Searching By Name...");
    getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.parseColor("colorString: "#374CBF"))));

    Bundle extras = getIntent().getExtras();
    name = extras.getString( key: "filename").toLowerCase();
    path = new File(extras.getString( key: "path"));
    pdf = extras.getBoolean( key: "pdf");
    word = extras.getBoolean( key: "word");
    txt = extras.getBoolean( key: "txt");

    cacheLoad = findViewById(R.id.cache2);
    showFilesRecyclerView = findViewById(R.id.showTimeByName_RV);

    showFilesRecyclerView.setAdapter(new SearchByName_RecyclerViewAdapter( flick: this));
    showFilesRecyclerView.setLayoutManager(new LinearLayoutManager( context: this));

    nameArray = name.split( regex: " ");
    totalItemsInserted=0;
    new startSearchingForFilesOnly().execute();
}

```

کد ۱۹-۳: SearchByNameStartActivity Class بخش دوم

حال کلاس StartSearchingForFilesOnly تعریف می‌شود. این کلاس دارای ۵ تابع می‌باشد. چهار تابع اول که شامل onPostExecute و doInBackground و onProgressUpdate و onPreExecute می‌شوند برگرفته از کلاس AsyncTask هستند و باید تعریف شوند. تابع onPreExecute قبل از اجرای این کلاس اجرا می‌شود. تابع doInBackground دقیقاً همان کدی است که باید اجرا شود. و تابع onProgressUpdate وظیفه بروزرسانی وضعیت به کاربر در حین اجرای کار را دارد که با تابع publishProgress صدا زده می‌شود. و در آخر تابع onPostExecute بعد از اجرای کد اجرا می‌شود. آخرین تابع این کلاس همان تابع سرچ ما می‌باشد (SearchForFilesOnly) که وظیفه دارد جستجوی اسناد برحسب نام آن‌ها را انجام دهد.

```

private class startSearchingForFilesOnly extends AsyncTask<Void,Integer,Void> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        cacheLoad.setText("Searching...");
    }

    @Override
    protected Void doInBackground(Void... voids) {
        searchForFilesOnly(path,nameArray,word,pdf,txt);
        publishProgress(...values: 2);
        return null;
    }

    protected void onProgressUpdate(Integer... i){
        int mycase = i[0];
        switch (mycase) {
            case 1:
                showFilesRecyclerView.getAdapter().notifyItemInserted( position: i[1]-1);
                break;
            case 2:
                cacheLoad.setText("Search is complete. Found " + showFilesRecyclerView.getAdapter().getItemCount() + " Files.");
                break;
        }
    }

    @Override
    protected void onPostExecute(Void unused) { super.onPostExecute(unused); }

    public void searchForFilesOnly
        (File path, String [] name, boolean isWord, boolean isPDF, boolean isTxt) {...}
}
}

```

کد ۲۰-۳: SearchByNameStartActivity Class بخش سوم

SearchByName_RecycleAdapter -۲-۲-۵-۳

وظیفه این کلاس، مدیریت بر روی RecyclerView در اکتیویته SearchByNameStartActivity می باشد. کد این کلاس بصورت زیر است.

این کلاس تنها شامل متغیر myFiles است که همان اسنادی هستند که مطابق با داده های کاربر یافت شده اند. یک اینترفیس نیز برای قابلیت کلیک کردن بر روی هر سند تعریف شده است.

```

package com.goly.golydocumentfinder;

import ...

public class SearchByName_RecycleAdapter extends RecyclerView.Adapter <SearchByName_ViewHolder> {
    public ArrayList<File> myFiles;
    private fileIsClickedInterface fici;

    interface fileIsClickedInterface {
        void fileIsClicked(File file);
    }

    public SearchByName_RecycleAdapter(fileIsClickedInterface fici){
        this.fici=fici;
        myFiles = new ArrayList<>();
    }

    @NonNull
    @Override
    public SearchByName_ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        LayoutInflater LI = LayoutInflater.from(parent.getContext());
        View view = LI.inflate(R.layout.search_by_name_viewholder, parent, attachToRoot: false);
        return new SearchByName_ViewHolder(view);
    }
}

```

کد ۲۱-۳: SearchByName_RecycleAdapter بخش اول

سپس تابع onBindViewHolder برای هر سند باید اجرا شود و مشخصات آن را به ViewHolder اعمال کند. تمامی اطلاعات هر سند در این بخش بر روی هر ViewHolder اعمال می‌شود. پس از آن تابع getItemCount وظیفه دارد مقدار کل اسناد را به RecyclerView اعلام کند. در آخر تابع addFile وظیفه دارد اسناد جدید را به myFiles اضافه کند.

```
@Override
public void onBindViewHolder(@NonNull SearchByName_ViewHolder holder, int position) {
    //File Info
    if(myFiles.get(position).getName().length() > 25)
        holder.getName().setText(myFiles.get(position).getName().substring(0,25)+"...");
    else holder.getName().setText(myFiles.get(position).getName());
    if(myFiles.get(position).getPath().length() > 25)
        holder.getPath().setText(myFiles.get(position).getPath().substring(0,25)+"...");
    else holder.getPath().setText(myFiles.get(position).getPath());
    SimpleDateFormat sdf = new SimpleDateFormat( pattern: "MM/dd/yyyy HH:mm:ss");
    holder.getDate().setText(sdf.format(myFiles.get(position).lastModified()));
    holder.getSize().setText(Long.toString( myFiles.get(position).length()/1000)+"KB");
    //image

    if(myFiles.get(position).getName().toLowerCase().endsWith(".pdf"))
        holder.getImage().setImageResource(R.drawable.pdf);
    else if(myFiles.get(position).getName().toLowerCase().endsWith(".txt"))
        holder.getImage().setImageResource(R.drawable.txt);
    else holder.getImage().setImageResource(R.drawable.word);

    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) { fici.fileIsClicked(myFiles.get(position)); }
    });
}

@Override
public int getItemCount() { return myFiles.size(); }
public void addFile(File file){
    myFiles.add(file);
    //notifyItemInserted(myFiles.size()-1);
}
}
```

کد ۲۲-۳: SearchByName_RecyclerViewAdapter بخش دوم

SearchByName_ViewHolder -۳-۲-۵-۳

این کلاس یک کلاس ساده از تمام داده‌هایی که یک سند ما باید نمایش دهد را شامل می‌شود. کد آن بصورت زیر نوشته می‌شود.


```

package com.goly.golydocumentfinder;

import ...

public class SearchByName_ViewHolder extends RecyclerView.ViewHolder {
    private TextView name;
    private TextView path;
    private TextView date;
    private TextView size;
    private ImageView image;
    public SearchByName_ViewHolder (View view){
        super(view);
        name = view.findViewById(R.id.searchByName_FileName_TW);
        path = view.findViewById(R.id.searchByName_Path_TW);
        date = view.findViewById(R.id.searchByName_Date_TW);
        size = view.findViewById(R.id.searchByName_Size_TW);
        image = view.findViewById(R.id.searchByName_File_ImageView);
    }

    public TextView get_Name() { return name; }

    public TextView get_Size() { return size; }

    public TextView get_Path() { return path; }

    public TextView get_Date() { return date; }

    public ImageView get_Image() { return image; }
}

```

کد ۲۳-۳: SearchByName_ViewHolder

۳-۵-۳ SearchByTextStartActivity

۱-۳-۵-۳ ساختار کد

اکتیویته SearchByTextStartActivity هنگامی اجرا می‌شود که تمامی داده‌های ورودی از صفحه SearchByTextTab دریافت شده باشند. حال باید با این داده‌ها شروع به جستجو بر اساس پرس‌وجوی کاربر کرد. کد این بخش از برنامه بصورت زیر طراحی شده است.

همانند اکتیویته SearchByNameStart ابتدا متغیرهای لازم تعریف می‌شوند. در این بخش علاوه بر ۵ متغیر قبلی، یک پرس‌وجو^۱ نیز از کاربر دریافت می‌شود و در تابع onCreate مقداردهی می‌شوند. بسیاری از توابع به‌کار رفته در این کلاس همانند کلاس SearchByNameStart می‌باشد. متغیرهای nameArray

^۱ Query

و queryArray در اصل همان متغیرهای name و query می‌باشند که براساس فاصله بین کلمات بصورت آرایه تبدیل شده‌اند. علاوه بر متغیرهایی که در گذشته استفاده شده‌اند، این اکتیویتی دارای متغیر PB از کلاس ProgressBar نیز می‌باشد. از آنجایی که عمل جستجو بر اساس پرس‌وجو کمی ممکن است طول بکشد، بهتر از با استفاده از یک ProgressBar وضعیت جست‌وجو در لحظه به کاربر نمایش داده شود. متغیر myData از کلاس WordSearchData که در گذشته درمورد آن صحبت شده است به منظور ذخیره اسناد با امتیاز آن اسناد در کنارشان استفاده می‌شود.

```
package com.goly.golydocumentfinder;

import ...

public class SearchByTextStartActivity extends AppCompatActivity implements SearchByText_RecyclerViewAdapter.fileIsClickedInterface {
    private WordSearchData myData;
    private ArrayList <File> myFiles;
    private RecyclerView RV;
    private ProgressBar PB;
    private TextView t;

    private String name;
    private String query;
    private File path;
    private String [] nameArray;
    private String [] queryArray;
    boolean pdf;
    boolean word;
    boolean txt;

    @Override
    public void fileIsClicked(File file) {
        Intent i = new Intent( packageContext: SearchByTextStartActivity.this, ShowFileInfoActivity.class);
        i.putExtra( name: "showfile", file.toString());
        startActivity(i);
    }
}
```

کد ۲۴-۳: SearchByTextActivityClass بخش اول

سپس تابع onCreate تعریف می‌شود. تمامی متغیرها مقداردهی می‌شوند و در انتها یک متغیر از کلاس StartLongSearching ساخته شده و بلافاصله بصورت چنددخی^۱ بر روی نخ‌های پس‌زمینه اجرا^۲ می‌شود.

^۱ MultiThread

^۲ Execute

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_search_by_text_start);
    setTitle("Searching By Text...");
    getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.parseColor( colorString: "#BF3737")));
    t = findViewById(R.id.SBT_Start_Title);
    RV = findViewById(R.id.showTimeByText_RV);
    PB = findViewById(R.id.SBT_Start_progressBar);

    Bundle extras = getIntent().getExtras();
    name = extras.getString( key: "filename").toLowerCase();
    query = extras.getString( key: "query");
    path = new File(extras.getString( key: "path"));
    pdf = extras.getBoolean( key: "pdf");
    word= extras.getBoolean( key: "word");
    txt = extras.getBoolean( key: "txt");

    nameArray = name.split( regex: " ");
    Log.i( tag: "Name Array",Integer.toString(nameArray.length));
    queryArray = query.split( regex: " ");
    Log.i( tag: "Query Array",Integer.toString(queryArray.length));
    t.setText("Searching...");

    myData = new WordSearchData();
    new startLongSearching().execute();

    //OLD META:
    /*...*/
}

```

کد ۲۵-۳: SearchByTextActivityClass بخش دوم

تابع loadString به منظور تبدیل هر سند به یک نوشته طراحی شده‌است. برای بررسی هر سند، این متد برای آن سند اجرا می‌شود.

```

protected String loadString(File file) throws IOException, Exception{
    String text;
    String name = file.getName().toLowerCase();
    //PDF
    if(name.endsWith(".pdf")){
        return DocumentReaderUtil.Companion.readPdfFileContent(file, context: this);
    }
    //MICROSOFT DOCUMENT
    else if (name.endsWith(".doc") || name.endsWith(".docx")) {
        return DocumentReaderUtil.Companion.readWordDocFile(file, context: this);
    }
    //TEXT
    else if (name.endsWith(".txt") ){
        Scanner sc = new Scanner(file);
        sc.useDelimiter("\\Z");
        return sc.next().toString();
    }

    else throw new Exception("This file is not a PDF or Microsoft word or Text.");
}

```

کد ۲۶-۳: SearchByTextActivityClass بخش سوم

تابع `searchForFilesOnly` که در بخش SBN نیز استفاده شده است، در ابتدای این برنامه برای یافتن اسناد دوباره باید اجرا شود. تابع `setMyTitleText` تنها نوشته اولین `TextView` در چیدمان `activity_search_by_text_start.xml` را تغییر می‌دهد.

```
protected ArrayList<File> searchForFilesOnly
    (File path, String [] name, boolean isWord, boolean isPDF, boolean isTxt){...}

protected void setMyTitleText(String x) { t.setText(x); }
```

کد ۲۷-۳: `SearchByTextActivityClass` بخش چهارم

کلاس `startLongSearching` که داخل کلاس `searchByTextStartActivity` تعریف شده است، تمامی وظایف جستجو در نخ پس‌زمینه را بر عهده دارد. تابع `doInBackground` در گذشته توضیح داده شده است. تابع `onProgressUpdate` به منظور نمایش وضعیت جستجو به کاربر طراحی شده است. و در آخر تابع `onPostExecute` که پس از جستجو اجرا می‌شود، اسناد یافت شده را به ترتیب به کاربر نمایش می‌دهد.

```
private class startLongSearching extends AsyncTask<Void,Integer,Void>{
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        setMyTitleText("Searching in files...");
    }

    @Override
    protected Void doInBackground(Void... voids) {...}

    protected void onProgressUpdate(Integer... i){
        switch(i[1]){
            case 0: //Update progress
                PB.setProgress(i[0] + 1);
                if(myFiles.get(i[0]).getName().length()<25)
                    setMyTitleText("Searching in files: " + (i[0] + 1) + "/" + myFiles.size() +
                        "\n" + myFiles.get(i[0]).getName());
                else setMyTitleText("Searching in files: " + (i[0] + 1) + "/" + myFiles.size() +
                    "\n" + myFiles.get(i[0]).getName().substring(0,24));
                break;
            case 1: //Sorting by Score:
                setMyTitleText("Sorting by Score...");
                break;
            case 2: //Search is finished.
                setMyTitleText("Search is complete. Found "+myData.path.size()+" Files.");
                break;
            case -1: //ERROR:
                setMyTitleText("Error.");
                break;
            case -2:
                FancyToast.makeText( context SearchByTextStartActivity.this, message: "Access denied: "
                    +myFiles.get(i[0]).getName(), Toast.LENGTH_SHORT,FancyToast.INFO, androidicon: false);
                break;
        }
    }
    @Override
    protected void onPostExecute(Void unused) {
        super.onPostExecute(unused);
        //RV:
        RV.setAdapter(new SearchByText_RecyclerAdapter(myData, fict SearchByTextStartActivity.this));
        RV.setLayoutManager(new LinearLayoutManager( context SearchByTextStartActivity.this));
    }
}
```

کد ۲۸-۳: `SearchByTextActivityClass` بخش پنجم

در انتها تابع SortMyData، اسناد یافت شده را بر اساس امتیاز آن‌ها به ترتیب می‌کند. الگوریتم آن بدین صورت است از ابتدا تا انتهای اسناد را بصورت یک حلقه طی می‌کند و در هر بار اجرا شدن، سندی که بیشترین امتیاز دارد را جایگزین آن می‌کند. سختی این الگوریتم $O(\log(x))$ می‌باشد.

```
protected void SortMyData(){
    for(int i = 0 ; i<myData.path.size() - 1 ; i++){          //-1 is because we don't need to sort the last remained file.
        int topScore = myData.score.get(myData.path.size()-1);
        int topScoreMark = myData.path.size()-1;
        for(int j = myData.path.size()-2; j>=i ; j--){          //-2 is because we searched in last file already.
            if(myData.score.get(j) > topScore){                //we found a new high score;
                topScore = myData.score.get(j);
                topScoreMark = j;
            }
        }
        //Swap topScore with i:
        File cache = myData.path.get(topScoreMark);
        int cacheScore = myData.score.get(topScoreMark);

        myData.path.set(topScoreMark,myData.path.get(i));
        myData.score.set(topScoreMark,myData.score.get(i));

        myData.path.set(i,cache);
        myData.score.set(i,cacheScore);
    }
}
```

کد ۲۹-۳: SearchByTextActivityClass بخش ششم

SearchByText_RecyclerView - ۲-۳-۵-۳

وظیفه این کلاس، مدیریت بر روی RecyclerView در اکتیویتی SearchByTextStartActivity می‌باشد. کد این کلاس دقیقاً شبیه به کلاس SearchByName_RecyclerView عمل می‌کند. تنها تفاوت‌های این کلاس با کلاس SearchByName_RecyclerView این است که جنس متغیر myFiles از WordSearchData می‌باشد و هنگام نمایش به کاربر، باید امتیاز اسناد نیز به کاربر نمایش داده شوند.

```
package com.goly.golydocumentfinder;
import ...

public class SearchByText_RecyclerView extends RecyclerView.Adapter <SearchByText_ViewHolder> {
    WordSearchData myFiles;

    interface fileIsClickedInterface {
        void fileIsClicked(File file);
    }
    fileIsClickedInterface fici;
    public SearchByText_RecyclerView(WordSearchData myFiles, fileIsClickedInterface fici){
        this.myFiles = myFiles;
        this.fici = fici;
    }

    @NonNull
    @Override
    public SearchByText_ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        LayoutInflater inflater = LayoutInflater.from(parent.getContext());
        View view = inflater.inflate(R.layout.search_by_text_viewholder, parent, attachToRoot: false);
        return new SearchByText_ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull SearchByText_ViewHolder holder, int position) {
        //File Info
        if(myFiles.path.get(position).getName().length() > 20)
            holder.getName().setText(myFiles.path.get(position).getName().substring(0,20)+"...");
        else holder.getName().setText(myFiles.path.get(position).getName());
        if(myFiles.path.get(position).getPath().length() > 20)
            holder.getPath().setText(myFiles.path.get(position).getPath().substring(0,20)+"...");
        else holder.getPath().setText(myFiles.path.get(position).getPath());
        SimpleDateFormat sdf = new SimpleDateFormat( pattern: "MM/dd/yyyy HH:mm:ss");
        holder.getDate().setText(sdf.format(myFiles.path.get(position).lastModified()));
        holder.getSize().setText(Long.toString( myFiles.path.get(position).length()/1000)+"KB");

        //image
        if(myFiles.path.get(position).getName().toLowerCase().endsWith(".pdf"))
            holder.getImage().setImageResource(R.drawable.pdf);
        else if(myFiles.path.get(position).getName().toLowerCase().endsWith(".txt"))
            holder.getImage().setImageResource(R.drawable.txt);
        else holder.getImage().setImageResource(R.drawable.word);

        //SCORE
        holder.get_Score().setText(Integer.toString( myFiles.score.get(position) ));
        //ClickListener for each view:
        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { fici.fileIsClicked(myFiles.path.get(position)); }
        });
    }

    @Override
    public int getItemCount() { return myFiles.path.size(); }
}
```

کد ۳-۳۰: SearchByText_RecyclerView

SearchByText_ViewHolder -۳-۳-۵-۳

این کلاس یک کلاس ساده از تمام داده‌هایی که یک سند ما باید نمایش دهد را شامل می‌شود. تفاوت آن با SearchByName_ViewHolder این است که یک متغیر اضافه‌تر دارد که نام آن Score می‌باشد و به منظور نمایش امتیاز هر سند به کاربر اضافه شده است. کد آن بصورت زیر نوشته می‌شود.

```
package com.goly.golydocumentfinder;

import ...

public class SearchByText_ViewHolder extends RecyclerView.ViewHolder {
    private TextView name;
    private TextView path;
    private TextView date;
    private ImageView image;
    private TextView score;
    private TextView size;

    public SearchByText_ViewHolder(View view){
        super(view);
        name = view.findViewById(R.id.searchByText_FileName_TW);
        path = view.findViewById(R.id.searchByText_Path_TW);
        date = view.findViewById(R.id.searchByText_Date_TW);
        image = view.findViewById(R.id.searchByText_File_ImageView);
        score = view.findViewById(R.id.searchByText_Score_TW);
        size = view.findViewById(R.id.searchByText_Size_TW);
    }

    public TextView get_Name() { return name; }

    public TextView get_Path() { return path; }

    public TextView get_Date() { return date; }

    public ImageView get_Image() { return image; }

    public TextView get_Score() { return score; }

    public TextView get_Size() { return size; }
}
```

کد ۳۱-۳: SearchByText_ViewHolder

۳-۵-۴ - ShowFileInfoActivity Class

این اکتیویتی هنگامی صدا زده می‌شود که کاربر بر روی یکی از اسناد یافت شده در اکتیویتی‌های SearchByNameStartActivity یا SearchByTextStartActivity کلیک کند. هدف این اکتیویتی نمایش اطلاعات آن سند از جمله نام سند، حجم سند، آدرس مکان سند و ... می‌باشد. کد این کلاس بصورت زیر می‌باشد.

ابتدا تمامی متغیرها ساخته شده و تعریف می‌شوند. همانطور که مشاهده می‌کنید اکثر متغیرها بصورت TextView برای نمایش نوشته هستند. یک تصویر برای نمایش نوع سند (PDF یا Word یا Text) و یک دکمه برای باز کردن آن سند نیز تعریف شده است. یک متغیر به نام file نیز که باید از اکتیویتی قبلی به این اکتیویتی پاس داده شده باشد نیز ساخته و تعریف می‌شود.

```
package com.goly.golydocumentfinder;

import ...

public class ShowFileInfoActivity extends AppCompatActivity {
    private TextView NameTW;
    private TextView FileTypeTW;
    private TextView DirectoryPathTW;
    private TextView FileSizeTW;
    private TextView ModifiedDateTW;
    private Button openFileButton;
    private ImageView image;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_show_file_info);
        Bundle extras = getIntent().getExtras();
        File file = new File(extras.getString( key: "showfile"));
        setTitle("File Info");
        getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.parseColor( colorString: "#DFD04A")));
        NameTW = findViewById(R.id.ShowFileInfo_nameTW);
        FileTypeTW = findViewById(R.id.ShowFileInfo_Type);
        DirectoryPathTW = findViewById(R.id.ShowFileInfo_Path);
        FileSizeTW = findViewById(R.id.ShowFileInfo_Size);
        ModifiedDateTW = findViewById(R.id.ShowFileInfo_ModifiedDate);
        openFileButton = findViewById(R.id.ShowFileInfo_openFile);
        image = findViewById(R.id.ShowFileInfo_image);
    }
}
```

کد ۳-۳۲: ShowFileInfoActivity Class بخش اول

پس از آن، شروع به پر کردن مشخصات سند برای نمایش به کاربر بصورت زیر انجام می‌شود.


```

NameTW.setText(file.getName());
String[] x = file.getName().split( regex: "\\.";
String format = x[x.length-1].toLowerCase();
switch(format){
case "pdf":FileTypeTW.setText("PDF File");break;
case "txt":FileTypeTW.setText("Text File");break;
case "doc":FileTypeTW.setText("Microsoft Word File");break;
case "docx":FileTypeTW.setText("Microsoft Word File");break;
default:FileTypeTW.setText(format+" File");break;}
DirectoryPathTW.setText(file.getAbsolutePath().getParent());
FileSizeTW.setText(file.length()/1024 + "KB");

SimpleDateFormat sdf = new SimpleDateFormat( pattern: "MM/dd/yyyy HH:mm:ss");
ModifiedDateTW.setText(sdf.format(file.lastModified()));

if(file.getName().toLowerCase().endsWith(".pdf"))
    image.setImageResource(R.drawable.pdf);
else if(file.getName().toLowerCase().endsWith(".txt"))
    image.setImageResource(R.drawable.txt);
else image.setImageResource(R.drawable.word);

```

کد ۳-۳۳: ShowFileInfoActivity Class بخش دوم

سپس برای دکمه یک `onClickListener` تعریف می‌کنیم. ممکن است این دکمه در برخی از دستگاه‌ها اجازه دسترسی به اپلیکیشن‌های دیگر را نداشته باشد. اگر این دسترسی را نداشته باشد به کاربر اخطار می‌دهد که باز کردن این سند ممکن نیست و بهتر است با استفاده از اپلیکشن مناسب جهت نمایش آن سند به آدرس موردنظر برود و سند خود را مشاهده کند.

```

openFileButton.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        try{
            if (file.exists()) //Checking for the file is exist or not
            {
                Uri myUri = FileProvider.getUriForFile( context: ShowFileInfoActivity.this,
                    authority: getApplicationContext().getPackageName() + ".provider", file);
                Intent objIntent = new Intent(Intent.ACTION_VIEW);
                switch(format)
                {
                    //MIMES:
                    case "pdf":objIntent.setDataAndType(myUri, type: "application/pdf");break;
                    case "txt":objIntent.setDataAndType(myUri, type: "text/plain");break;
                    case "doc":objIntent.setDataAndType(myUri, type: "application/msword");break;
                    case "docx":objIntent.setDataAndType(myUri, type: "application/vnd.openxmlformats-officedocument.wordprocessingml.document");break;
                    default:break;
                }
                objIntent.setDataAndType(Uri.fromFile(file), type: "application/pdf");
                objIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                objIntent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
                startActivity(objIntent);
            }
        } catch(Exception e){
            FancyToast.makeText( context: ShowFileInfoActivity.this, message: "Your Android version does not support opening " +
                "Documents in Goly Document Finder app. Please open your file in your file explorer.",
                Toast.LENGTH_LONG, FancyToast.ERROR, androidIcon: false).show();
        }
    }
});
}
}

```

کد ۳-۳۴: ShowFileInfoActivity Class بخش سوم

AskPermissionActivity Class - ۳-۵-۵

هدف این اکتیویتی، دریافت اجازه کاربر برای خواندن اسناد در حافظه دستگاه می‌باشد. تا زمانی که کاربر اجازه دسترسی ندهد برنامه از این صفحه خارج نمی‌شود. این کلاس تنها یک دکمه دارد که هنگام کلیک بر روی آن، از کاربر درخواست اجازه خواندن حافظه را می‌طلبد.

کد این کلاس بصورت زیر طراحی شده است.

```
package com.goly.golydocumentfinder;

import ...

public class AskPermissionActivity extends AppCompatActivity implements View.OnClickListener{
    Button askPermissionButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ask_permission);
        askPermissionButton = findViewById(R.id.askPermissionButton);
        askPermissionButton.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        if (Build.VERSION.SDK_INT >= 23 && //android marshmello. if it goes under 23 we don't need permission
            ActivityCompat.checkSelfPermission( context: AskPermissionActivity.this,
                Manifest.permission.READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED)
            requestPermissions(new String[] {Manifest.permission.READ_EXTERNAL_STORAGE,
                Manifest.permission.WRITE_EXTERNAL_STORAGE}, requestCode: 1234);
        else {
            FancyToast.makeText( context: AskPermissionActivity.this, message: "Permissions Granted.",
                Toast.LENGTH_SHORT, FancyToast.INFO, androidIcon: false).show();
            Intent x = new Intent( packageContext: this,MainActivity.class);
            startActivity(x);
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode,
        @NonNull String[] permissions,
        @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        switch (requestCode){
            case 1234:
                if(grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED
                    && grantResults[1] == PackageManager.PERMISSION_GRANTED) {
                    {
                        FancyToast.makeText( context: this, message: "Permissions granted.",
                            Toast.LENGTH_SHORT, FancyToast.INFO, androidIcon: false).show();
                        Intent x = new Intent( packageContext: this,MainActivity.class);
                        startActivity(x);
                    }
                }else FancyToast.makeText( context: this, message: "Permissions denied.",
                    Toast.LENGTH_SHORT, FancyToast.CONFUSING, androidIcon: false).show();
                break;
        }
    }
}
```

کد ۳-۳۵: AskPermissionActivity Class

۶-۳- طراحی بخش UI

طراحی ظاهری برنامه و ساخت محیط کاربری بهتر، یکی از مهم‌ترین ویژگی‌های طراحی یک اپلیکیشن می‌باشد. این طراحی از جنبه‌های مختلفی مورد بررسی قرار می‌گیرد؛ از جمله رنگ‌بندی مناسب، چیدمان مناسب، انیمیشن‌های مناسب با اپلیکیشن و ...

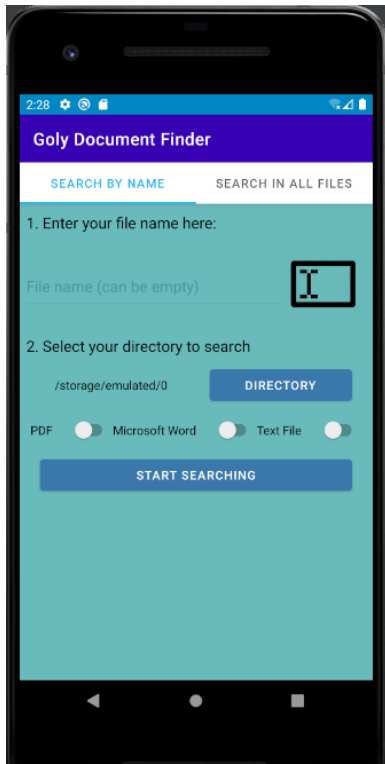
طراحی بخش UI این برنامه به چهار دسته تقسیم می‌شود. دسته اصلی به چیدمان یا Layouts مربوط می‌شود که بخش اصلی طراحی تمامی اکتیویتی‌ها را شامل می‌شود. پس از آن، تمامی تصاویر و یا آیکون اپلیکیشن در دو بخش drawable و mipmap قرار می‌گیرند. در آخر، متغیرهای ثابت در بخش Values قرار می‌گیرند که به عنوان مثال، تم و رنگ‌بندی برنامه را شامل می‌شود.

۱-۶-۳ Layouts

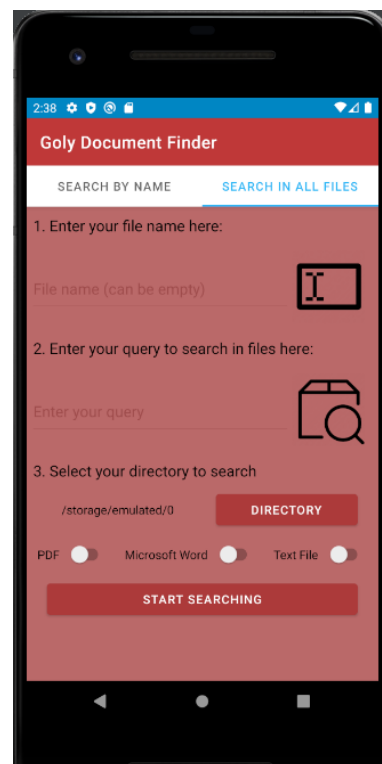
چیدمان‌ها یا Layouts شامل تمامی طراحی‌های محیط کاربری از جمله متن‌ها، عکس‌ها، فیلدها و ... می‌شود. انواع چیدمان‌های این برنامه عبارت‌اند از:

activity_main.xml - ۱-۱-۶-۳

اولین چیدمانی که به کاربر نمایش داده می‌شود، چیدمان activity_main.xml می‌باشد. این چیدمان، شامل یک appBarLayout و viewPager می‌باشد که باعث می‌شود این اکتیویتی بتواند چند فرگمنت را کنار یک‌دیگر بصورت صفحه‌ای نمایش دهد.



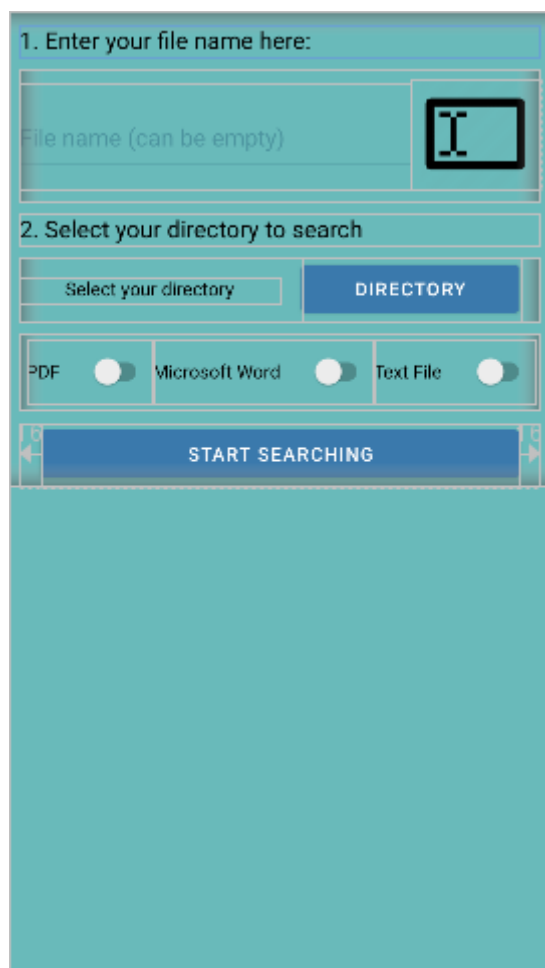
شکل ۵-۳: activity_main.xml SBN



شکل ۶-۳: activity_main.xml SBT

۳-۶-۲- fragment_search_by_name_tab.xml

فرگمنت اول که صفحه اول main_activity.xml را شامل می‌شود، فرگمنت جستجو بر اساس نام می‌باشد. این فرگمنت دارای یک عدد EditText و سه عدد Switch و دو عدد Button است که با کلیک کاربر همگی قابل تغییراند. طراحی ظاهری این فرگمنت بصورت زیر می‌باشد.



شکل ۷-۳: fragment_search_by_name_tab.xml

۳-۶-۳- fragment_search_by_text_tab.xml

فرگمنت دوم که صفحه دوم main_activity.xml را شامل می‌شود، فرگمنت جستجو بر اساس پرس‌وجو می‌باشد. این فرگمنت دارای دو عدد EditText و سه عدد Switch و دو عدد Button است که با کلیک کاربر همگی قابل تغییراند. طراحی ظاهری این فرگمنت بصورت زیر می‌باشد.

1. Enter your file name here:

File name (can be empty)

2. Enter your query to search in files here:

Enter your query

3. Select your directory to search

Select your directory

DIRECTORY

PDF ☐ Microsoft Word ☐ Text File ☐

START SEARCHING

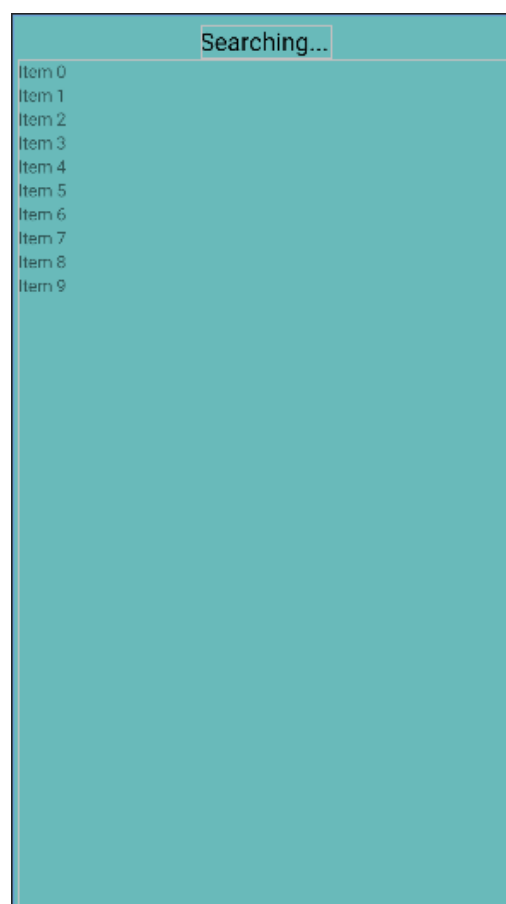
شکل ۸-۳: fragment_search_by_text_tab.xml

my_toolbar.xml ۳-۶-۱-۴-

این چیدمان، یک چیدمان ساده جهت استفاده در activity_main.xml می‌باشد. هدف این چیدمان، کنترل سربرگ‌ها در اکتیویته MainActivity است.

activity_search_by_name_start.xml ۳-۶-۱-۵-

این چیدمان که وابسته به اکتیویته SearchByNameStartActivity است، برای نمایش نتیجه جستجو بر اساس نام بکار گرفته می‌شود. طراحی ظاهری این چیدمان بصورت زیر می‌باشد و نمونه خروجی آن نیز کنار آن قرار دارد.



شکل ۹-۳: activity_search_by_name_start.xml

search_by_name_viewholder.xml -۶-۱-۶-۳

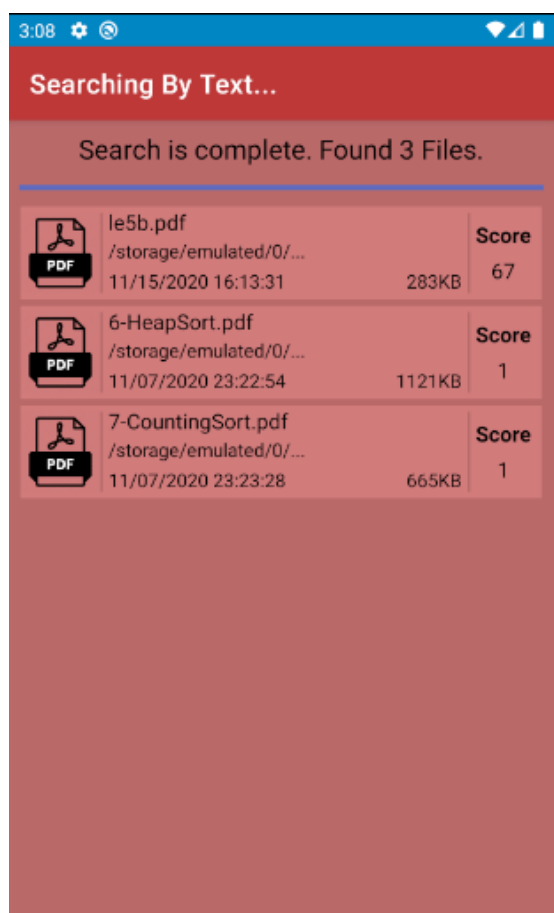
این چیدمان، یک چیدمان برای نمایش هر سند در چیدمان activity_search_by_name_start.xml می‌باشد. طراحی ظاهری این چیدمان بصورت زیر است.

	name	
	path	
	date	size

شکل ۱۰-۳: search_by_name_viewholder

activity_search_by_text_start.xml -۷-۱-۶-۳


این چیدمان که وابسته به اکتیویته SearchByTextStartActivity است، برای نمایش نتیجه جستجو بر اساس پرس‌وجو بکار گرفته می‌شود. طراحی ظاهری و نمونه خروجی این چیدمان بصورت زیر می‌باشد.



شکل ۱۱-۳: activity_search_by_text_start.xml

۸-۱-۶-۳ search_by_text_viewholder.xml

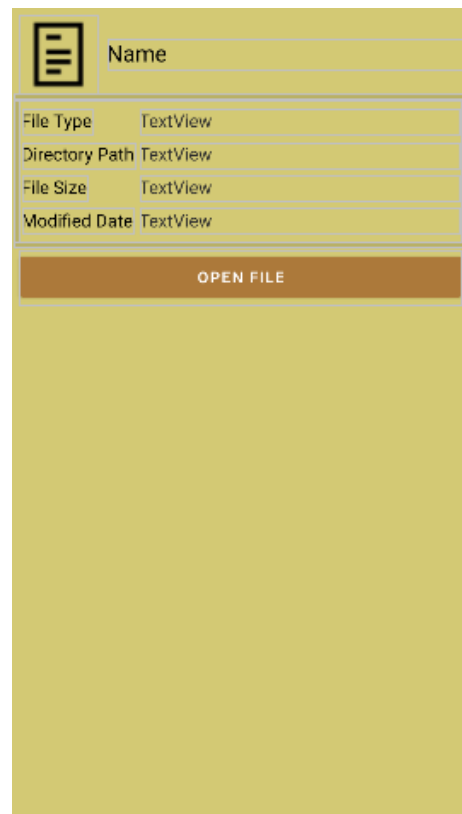
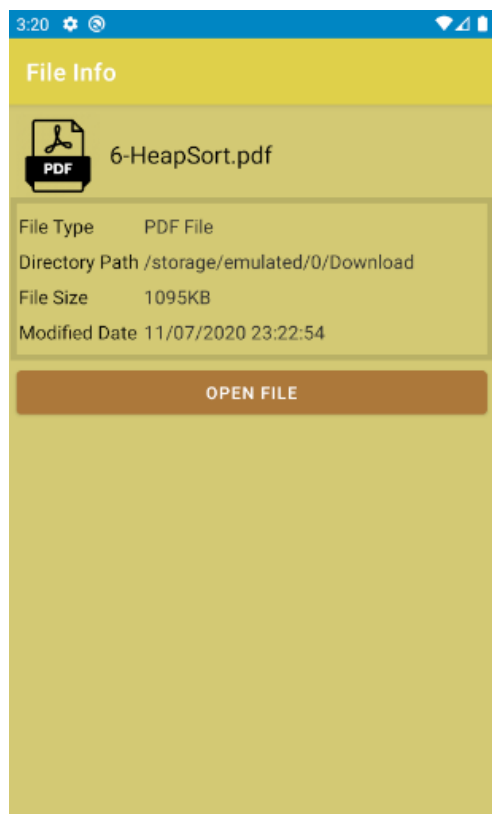
این چیدمان، یک چیدمان برای نمایش هر سند در چیدمان activity_search_by_text_start.xml می‌باشد. طراحی ظاهری این چیدمان بصورت زیر است.

	name		Score
	path		
	date	size	
			-1

شکل ۱۲-۳: search_by_text_viewholder.xml

۹-۱-۶-۳ activity_show_file_info.xml

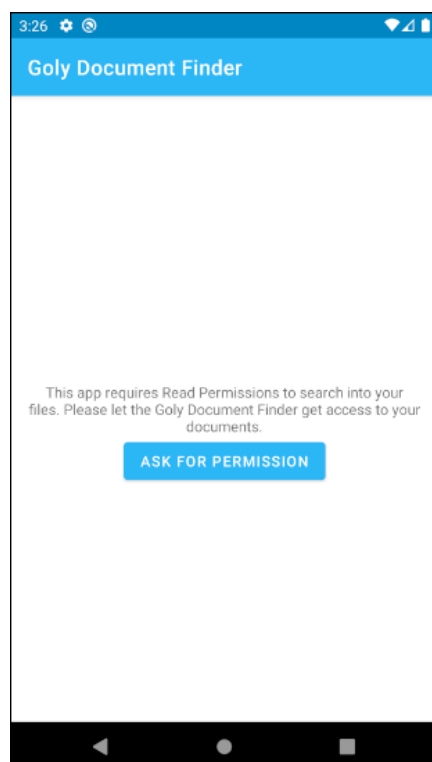
این چیدمان که به اکتیویته ShowFileInfoActivity وابسته است، جهت نمایش مشخصات یک سند که در بخش مشاهده نتایج جستجو بر روی آن کلیک می‌شود طراحی شده است. طراحی ظاهری و نمونه خروجی این چیدمان بصورت زیر می‌باشد.



شکل ۱۳-۳: activity_show_file_info.xml

۱۰-۱-۶-۳ activity_ask_permission.xml

این چیدمان که به اکتیویته `AskPermissionActivity` وابسته است، برای درخواست اجازه دسترسی به حافظه دستگاه طراحی شده است. طراحی ظاهری این چیدمان بصورت زیر می باشد.



شکل ۱۴-۳: activity_ask_permission.xml

Drawable - ۲-۶-۳

این بخش از برنامه، شامل تصاویری می‌شود که به این برنامه اضافه شده‌اند.

Mipmap - ۳-۶-۳

این بخش از برنامه شامل تصاویر اصلی اپلیکیشن از جمله آیکون اپلیکیشن و ... می‌باشد.

Values - ۴-۶-۳

در این بخش، مقدارهای ثابت برنامه قرار می‌گیرند. از جمله رنگ‌ها یا متن‌های ثابت، استایل‌ها و شخصی‌سازی‌های استفاده شده و... .

فصل ۴ - بررسی کارکرد اپلیکیشن و اشکالات رفع شده

۴-۱- مشکلات رفع شده

در زمان طراحی و کدنویسی برنامه، مشکلات و باگ^۱های فراوانی در حین تست برنامه ظاهر شد و در حال حاضر رفع شده است. نمونه‌های این مشکلات عبارت اند از:

- نبود قابلیت جستجوی چند کلمه در بخش SBT و SBN: این مشکل پس از تعریف آرایه برای پرس‌وجو و نام فایل‌ها و تغییری جزئی در توابع جست‌وجو رفع شد.
- به ترتیب نمایش دادن اسناد در بخش SBT پس از محاسبه امتیاز آن‌ها: رفع این مشکل با استفاده از کلاس Async و محاسبات چندمنحی انجام شد.
- نمایش وضعیت سرچ به کاربر در بخش SBT: برطرف شد.
- خطای دریافت آدرس اولیه از کاربر در دستگاه‌های با اندروید بالاتر از ۷: استفاده از try-catch در بخش استفاده از UnicornFilePicker و استفاده از توابع مخصوص اندروید ۷ و بالاتر.
- مشکل در نحوه محاسبه امتیاز برای کلماتی که مانند پرس‌وجو به ترتیب کنار یکدیگراند: بررسی و حل شد.
- جا نشدن چیدمان نمایش حجم اسناد در کلاس ViewHolder: استفاده از روش وزن‌دهی به هر ویو برای نمایش بهتر در دستگاه‌های اندرویدی مختلف.
- خراب شدن ویوها در چیدمان activity_show_file_info.xml به‌دلیل طولانی بودن نام سند: نمایش ۲۵ حرف ابتدایی نام سند به کاربر.
- بسته شدن برنامه در اندرویدهای بالاتر از ۸ در صورت تلاش برای باز کردن اسناد در ShowFileInfoActivity: این مشکل متأسفانه در حال حاضر بخاطر مسائل امنیتی اندروید برای اندرویدهای بالاتر از ۸ وجود دارد ولی بجای بسته‌شدن برنامه، به کاربر هشدار می‌دهد که بجای تلاش برای باز کردن اسناد در برنامه، از راه Explorer خود دستگاه به آدرس مورد نظر برود.

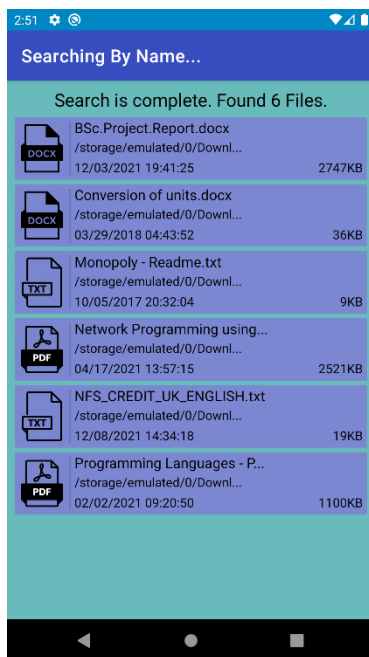
¹ Bug

۲-۴- نمونه خروجی اپلیکیشن

برای تست این برنامه، اسناد اولیه زیر را در نظر می‌گیریم.

BSc.Project.Report.docx	12/3/2021 7:41 PM	Microsoft Word D...	2,684 KB
Conversion of units.docx	3/29/2018 4:43 AM	Microsoft Word D...	36 KB
Monopoly - Readme.txt	10/5/2017 8:32 PM	Text Document	10 KB
Network Programming using Python - English.pdf	4/17/2021 1:57 PM	Chrome HTML Do...	2,463 KB
NFS_CREDIT_UK_ENGLISH.txt	12/8/2021 2:34 PM	Text Document	20 KB
Programming Languages - Persian.pdf	2/2/2021 9:20 AM	Chrome HTML Do...	1,075 KB

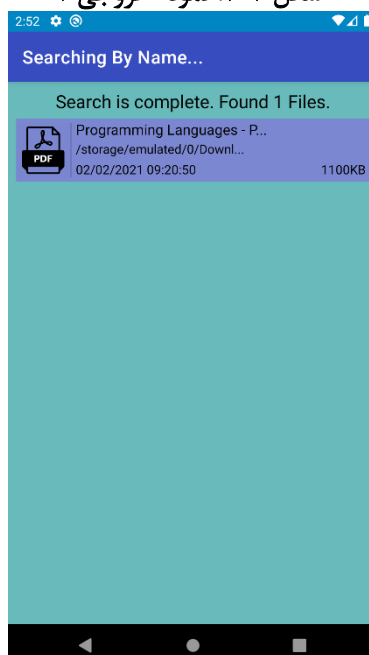
شکل ۱-۴: اسناد اولیه برای بررسی اپلیکیشن



نتایج زیر برای جستجو در دستگاه Nokia 5.1 Plus بدست آمده‌اند.

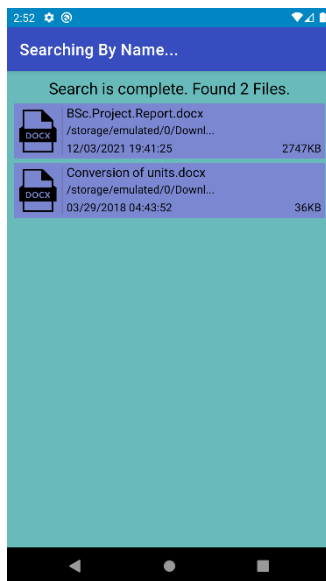
جستجو بر اساس نام - جستجوی تمام اسناد بدون در نظر گرفتن فرمت و نام آن‌ها.
مدت زمان جستجو: کمتر از یک ثانیه

شکل ۲-۴: نمونه خروجی ۱

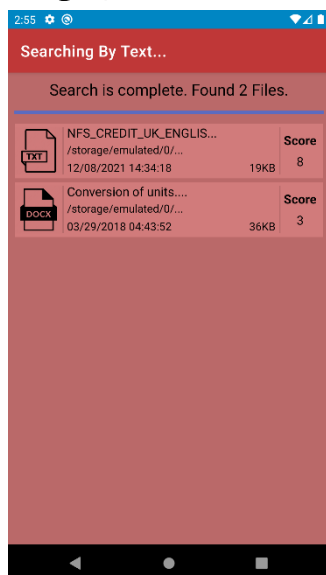


جستجو بر اساس نام - جستجو در تمام فرمت‌ها برای اسناد با نام "rogrami nguage" که با اشتباهات املایی همراه است.
مدت زمان جستجو: کمتر از یک ثانیه

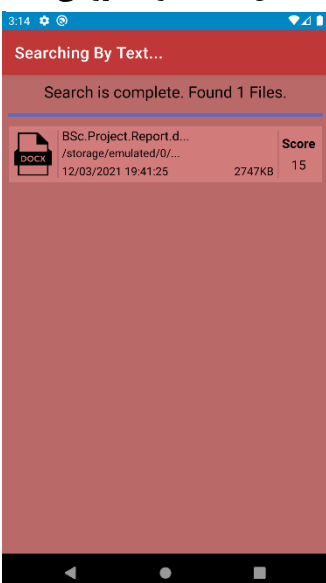
شکل ۳-۴: نمونه خروجی ۲



شکل ۴-۴: نمونه خروجی ۳



شکل ۴-۵: نمونه خروجی ۴



شکل ۴-۷: نمونه خروجی ۵

جستجو بر اساس نام – جستجو در تمام نام‌ها با فرمت خاص
Microsoft Word

مدت زمان جستجو: کمتر از یک ثانیه

جستجو بر اساس پرس‌وجو – جستجو در تمام اسناد با استفاده از
پرس‌وجوی "Rights" در تمامی فرمت‌ها.
مدت زمان جستجو: ۶ ثانیه

جستجو بر اساس پرس‌وجو – جستجو در تمام اسناد با استفاده از
پرس‌وجوی "نحوه" در فرمت Microsoft Word.
مدت زمان جستجو: ۳ ثانیه

مقایسه با نتایج یافت شده با استفاده از نرم‌افزار کامپیوتری Microsoft Word:



شکل ۴-۶: نتیجه جستجو در نرم‌افزار Microsoft Word

فهرست مراجع

Introduction to Information Retrieval, by Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, Cambridge University Press, 2009.

