دانشگاه ارومیه

دانشکده فنی و مهندسی

گروه مهندسی کامپیوتر

پروژه پایانی کارشناسی کامپیوتر گرایش نرم‌افزار

دسکتاپ اپلیکیشن جست و جو با ابزار لوسین

سیاوش حسن‌پور آده

استاد راهنما

دکتر اصغریان

بهمن ماه سال 1396

دانشگاه ارومیه

دانشکده فنی و مهندسی

گروه مهندسی کامپیوتر

پروژه پایانی کارشناسی کامپیوتر گرایش نرم‌افزار

دسکتاپ اپلیکیشن جست و جو با ابزار لوسین

سیاوش حسن‌پور آده

استاد راهنما

دکتر اصغریان

بهمن ماه سال 1396

دانشگاه ارومیه

دانشکده فنی و مهندسی

گروه مهندسی کامپیوتر

دسکتاپ اپلیکیشن جست و جو با ابزار لوسین

دانشجو:

سیاوش حسن‌پور آده

این پروژه به عنوان بخشی از فعالیتهای علمی - پژوهشی مقطع کارشناسی مهندسی **کامپیوتر** گرایش **نرم‌افزار** در تاریخ **1396/11/15** توسط اساتید ذیل مورد پذیرش قرار گرفت.

استاد راهنما: دکتر اصغریان

داور:

مدیر گروه:

دانشگاه ارومیه

دانشکده فنی و مهندسی

تعهد نامه پژوهشی

نظر به اینکه چاپ و انتشار پروژه‌های تحصیلی دانشجویان دانشگاه ارومیه مبین بخشی از فعالیتهای علمی - پژوهشی دانشجو میباشد که با استفاده از اعتبارات دانشگاه انجام میشود، برای آگاهی دانشجو و رعایت حقوق دانشگاه، دانش آموختگان گرامی نسبت به رعایت موارد ذیل متعهد میشوند:

1. قبل از چاپ گزارش پروژه خود، مراتب را بطور کتبی به مدیریت گروه اطلاع و کسب اجازه نمایند.
2. در انتشار نتایج پروژه در قالب مقاله، همایش، اختراع، اکتشاف و سایر موارد ذکر نام دانشگاه ارومیه الزامی است.
3. انتشار نتایج پروژه باید با اطلاع و کسب اجازه از استاد راهنما صورت گیرد.

اینجانب **سیاوش حسن‌پور آده** دانشجوی گرایش **نرم افزار** مقطع **کارشناسی** تعهدات فوق و ضمانت اجرایی آنرا قبول کرده و به آن ملتزم میشوم.

تاریخ و امضا دانشجو

چکیده

این پروژه یک سیستم ایندکسینگ و جست و جو است که با استفاده از کتابخانه lucene طراحی شده است.در این سیستم ادرس بخشی از حافظه اصلی کامپیوتر خود را به اختیار به نرم افزار میدهیم که ایندکس کرده تا بتوانیم جست و جو کنیم . در این سیستم علاوه بر جست و جو می‌توان فایل‌های پیدا شده را از همان محیط جست و جو باز کرده و مشاهده کرد. برای انجام این پروژه ابتدا نیازمندی ها را بررسی کردیم و ابزار های مورد نیاز را مشخص کردیم.

واژه‌های کلیدی:

Lucene ، Index ، جست و جو ، Analyzer ، Reader ، Searcher ، سکتاپ اپلیکیشن

فهرست مطالب

صفحه

فصل اول

مقدمه

پروژه کارشناسی اینجانب، دسکتاپ اپلیکیشن ایندکس و جست و جو است که در آن می‌توان قسمتی از حافظه کامپیوتر خود را ایندکس کرد و در هنگام جست و جو بسیار سریع تر به آنها دسترسی پیدا کرد.

اهمیت این پروژه برای دسترسی سریع به فایل های مورد نظر ما از بین هزاران فایل موجود در سیستم ماست .

این پروژه برای شرکت های که با پروژه‌های بزرگ سروکار دارند و افراد زیادی در توسعه این پروژه‌ها سهیم هستند و هر روزه فایل های زیادی افزوده می‌شود بسیار مفید هست که برای پیدا کردن یک محتوا نیازی نباشد چندین فایل مختلف را برسی کنند . همچنین فرم‌های دیگر این پروژه را می‌توان در سرور ها و در سایت (به عنوان مثال سایت ویکیپدیا) و در گوشیها و به طور کلی می‌توان از آن در توسعه هر نوع برنامه‌ای که می‌خواهید قابلیت جستجو بر روی داده های متنی را داشته باشد، استفاده کرد.

در فصل های بعد به تحلیل و طراحی و نمودارهای مربوطه به آنها، نحوه پیاده سازی سیستم، راهنمای کاربری و استفاده از سیستم و همچنین نتیجه گیری کلی خواهیم پرداخت.

۱-۱ انگیزه طراحی نرم افزار

سیستم جست و جو ویندوز تا جایی که من میدانم سیستم نسبتا ضعیفی هست و فقط قادر است نام فایل ها و پوشه هایی که با عبارت جست و جو صدق می‌کنند رو نمایش دهد و اگر سرویس Index ویندوز فعال شود سرعت سیستم بسیار پایین می آید. این ایده برای سرعت بخشیدن به این جست و جو برای گروه های کوچک و بزرگ برنامه نویس ویا افرادی که مدام با فایل های زیادی سروکار دارند و حتی برای مصارف کاربران معمولی نیز طراحی شده است. این پروژه پتانسیل زیادی برای گسترش دارد.

۱-۲ شرح نیاز و تحلیل نیازمندی ها

از انجایی که سیستم جست و جوی ویندوز کند بوده و بدون در نظر گرفتن محتوا و فقط با نام پوشه و فایل جست و جو می کنند باعث می‌شود به عنوان مثال گروه های برنامه نویسی که بر روی یک پروژه کار می‌کنند و این پروژه دارای ۲۰ الی ۳۰ فایل ویا کلاس جداگانه است ، برای این که به دنبال یک کلاس یا تابع یا یک بخش از کدی که خود و یا همکارشان نوشته است، بگردد باید فایل های زیادی را باز کرده و جستو جو کند ولی با این سیستم به راحتی و با صرف زمان کمی به مطلب مورد نیاز خود برسد. همچنین برای کاربران معمولی که میخواهند محتوایی را از بین چند ترابایت داده در حافظه ی کامپیوترشان را جست و جو کنند اگر از طریق خود ویندوز جست و جو کنند زمان خیلی زیادی صرف می‌شود تا نتیجه بگیرد ولی با استفاده از این سیستم به راحتی و خیلی سریع به محتوایی که می‌خواهد می‌رسد.

۱-۳ کارکردها و کاربردهای نرم‌افزار

توسط این اپلیکیشن شرکت‌ها ، گروه های توسعه نرم‌افزاری و حتی فروشندگان محصولات فرهنگی و کاربران معمولی و ... می‌توانند به راحتی و خیلی سریع به محتوایی که می‌خواهند می‌رسند.

۱-۴ معرفی نرم افزارهای مشابه

از وجود اپلیکیشن هایی شبیه اپلیکیشنی که ما طراحی کرده ایم اطلاعی نداریم ولی اپلیکیشن و وب سایت هایی هستند که تا حدودی کاری مشابه سیستم ما میکنند یعنی در آنها از lucene برای این ایندکس جست و جو استفاده میکنند .

برای مثال سایت هایی مانند twitter از lucene برای real time search استفاده می‌کنند و کمپانی هایی مانند apple و ibm از این ابزار قدرتمند استفاده می‌کنند. به طور کلی اپلیکیشن ها و وب اپلیکیشن هایی را که از lucene استفاده میکنند را میتوان در سایت https://wiki.apache.org/ مشاهده کرد.

فصل دوم

تحلیل و طراحی

۲-۱ معرفی Apache Lucene

Apache Lucene یک پروژه متن‌باز به زبان جاوا است که امکان افزودن قابلیت جستجو به برنامه های کاربردی را با مکانیزمی کارا و آسان فراهم می‌آورد . به زبانی دیگر Lucene مجموعه‌ای از کتابخانه‌های کاربردی و مفید است که می‌توان از آن در توسعه هر نوع برنامه‌ای که می‌خواهید قابلیت جستجو داشته باشد، استفاده کرد. این کتابخانه به زبان جاوا (Java) نوشته شده و سپس به زبان های Delphi, Perl, C#, C++, Python, Ruby, و PHP پیاده سازی (پورت) شده است.

۲-۲ Lucene چگونه کار میکند:

Lucene جستجوی سریع خود را با استفاده از ایندکس‌ها به نتیجه می‌رساند و به جای استفاده از ایندکس‌های کلاسیک که در آن هر سند شامل لیست کاملی از واژه‌هایی است که در آن وجود دارند از ایندکس‌های وارونه استفاده می‌کند. بدین گونه که برای هر واژه لیستی از سندها را فراهم می‌آورد که آن واژه در آن‌ها وجود دارد. همچنین مکان یا مکان‌هایی که هر واژه در سند تکرار شده است را نیز در لیست اعمال می‌کند. برای نمونه در شکل زیر واژه lucene در سند اول و در مکان ششم آن وجود دارد، همچنین در سند دوم و مکان چهارم ان .

```
Lucene    -> { (1,6), (2,4) }
full      -> { (1,10),(2,9) }
going     -> { (1,0) }
index     -> { (3,3) }
search    -> { (1,11), (2,10)}
```

شکل 2-1 مثال چگونگی کارکرد lucene

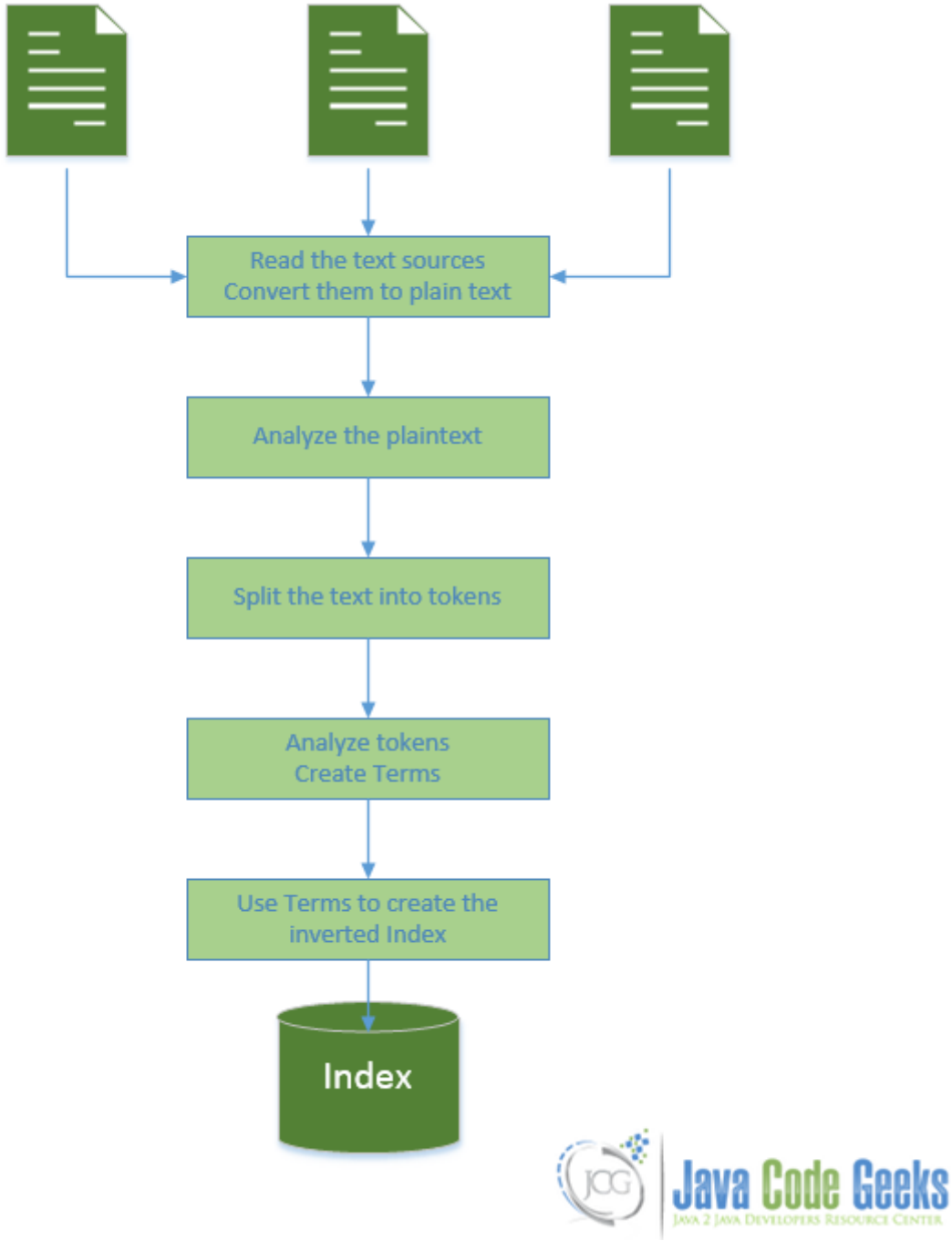
۲-۳ مراحل استفاده از کتابخانه Lucene:

در ادامه مراحل مختلف برای آماده‌سازی و انجام جستجو را بررسی کرده.

۲-۳-۱ مرحله اول: index کردن اطلاعات:

Lucene قبل از انجام عمل جستجو کارهایی را انجام می‌دهد که یکی از آن‌ها ایندکس کردن است.جریان فرایند ایندکس کردن را در تصویر زیر مشاهده می‌کنید.

همان‌گونه که در شکل ۲,۲ مشاهده می‌کنید سندها ورودی این جریان هستند. اگر سندی دارید با فرمت خاصی مثلاً یک PDF، باید آن را با قابلیت‌هایی که در زبان برنامه نویسی مورد استفاده وجود دارد به متنی واضح و آشکار تبدیل کنید تا ادامه کارها بدرستی انجام شود. سپس این متن آنالیز می‌شود و به کلماتی جدا از هم تجزیه می‌شود. در این مرحله آنالیزهای مختلفی می‌تواند انجام شود که بسته به نیازی که برنامه شما دارد می‌توانید این آنالیزها را سفارشی کنید. پس از این مرحله ایندکس‌های وارون برای هر کلمه تولید می‌شود. اکنون این ایندکس‌ها قابل جستجو هستند و می‌توان باز هم بسته به نیاز، درخواست‌هایی با فرمت‌های مختلف روی اندکس‌ها اعمال کرد.



شکل 2-2 مراحل کلی ایندکس کردن lucene

اجزای اصلی مرحله ایندکس اطلاعات به شرح زیر است:

❖ دایرکتوری‌ها

ایندکس‌های تولید شده توسط Lucene می‌توانند در مکانی روی سیستم فایل یا برای عمل‌کرد سریع‌تر در حافظه اصلی یا روی یک پایگاه داده ذخیره شوند. برای پیاده‌سازی هر کدام از این گزینه‌ها کلاس‌هایی وجود دارد که همه آن‌ها از کلاس انتزاعی Directory ارث‌بری می‌کنند. برای سادگی و درک بهتر از همان ذخیره کردن ایندکس‌ها روی سیستم فایل استفاده می‌کنیم. استفاده از سایر گزینه‌ها در عمل تفاوت چندانی با هم ندارند. Lucene هر آنچه را که برای ذخیره ایندکس‌ها ضروری باشد در یک دایرکتوری ذخیره می‌کند. برای کار کردن با دایرکتوری مد نظر می‌توانید از کلاس FSDirectory استفاده کنید و به عنوان ورودی به آن یک مسیر مطلق از سیستم فایل خودم می‌دهیم. (برای کار کردن با حافظه می‌توانید از کلاس RAMDirectory استفاده کنید.)

❖ سندها

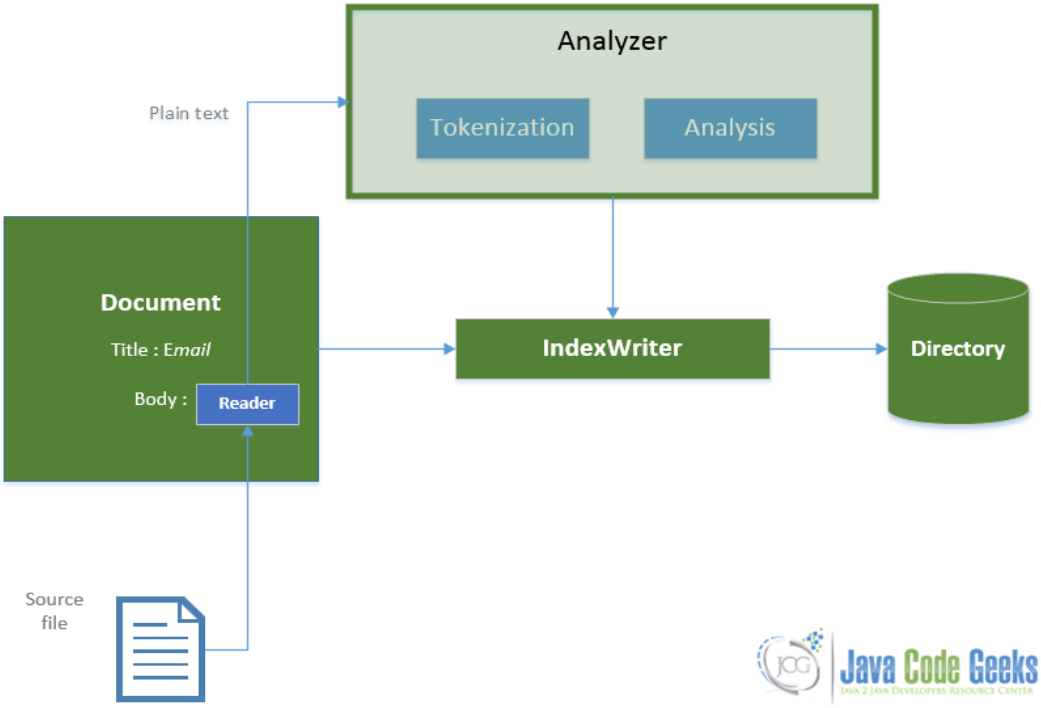
اسنادی که می‌خواهید آن‌ها را قابل جستجو کنید می‌توانند هر نوع فایل متنی مانند اسناد Word، فایل‌های PDF یا هر نوع سند متنی دیگر باشند. برای هر سندی که می‌خواهید آن را ایندکس کنید باید یک شی از کلاس Document ایجاد کنید. همچنین بعد از اینکه پرس‌وجویی را روی ایندکس‌ها انجام دادید نتایج به صورت لیستی از سندها به شما نمایش داده می‌شوند.

❖ فیلد ها

به هر سند می‌توان مجموعه‌ای از فیلدها اضافه کرد که هر کدام می‌توانند اطلاعاتی را درباره سند ارائه دهند. برای مثال یک فیلد می‌تواند عنوان، شناسه سند، توصیفی درباره‌ی سند یا محتوای آن باشد. هر فیلد می‌تواند شامل سه مشخصه باشد: نام، نوع و مقدار. نام و مقدار که نیازی به توضیح ندارند اما نوع فیلد نمایانگر رفتار سند می‌باشد. برای مثال می‌توانید نوع فیلد را به‌گونه‌ای تنظیم کنید که روی ذخیره‌شدن، ایندکس‌شدن یا تجزیه شدن اجزای یک مقدار کنترل داشته باشید.

❖ انالیزر

آنالیزور یکی از مهمترین اجزای فرایند ایندکس‌گذاری و جستجو است. آنالیزور مسئول گرفتن متن خام ورودی و تبدیل آن به واژه‌های قابل جستجو است. آنالیزور داخل خود از یک Tokenizer بهره می‌برد که ورودی یاد شده را به مجموعه‌ای از واژه‌ها می‌شکند. آنالیزور علاوه بر توکن‌کردن ورودی کارهای دیگری نیز انجام می‌دهد. برای مثال می‌تواند قبل از توکن‌کردن یک پیش‌پردازش روی متن انجام دهد و الگوهایی مشخص از متن، مثل تگ‌های HTML را حذف کند. یا اینکه بعد از توکن‌کردن عملیاتی مانند ریشه‌یابی واژه‌ها (Stemming)، حذف stop wordها و ... را انجام دهد. بسته به این امکاناتی که آنالیزور انجام می‌دهد کلاس‌های مختلفی وجود دارد. یکی از این کلاس‌ها StandardAnalizer است که stop wordها را حذف می‌کند و واژه‌ها را به حروف کوچک تبدیل می‌کند و ریشه‌یابی نیز را انجام می‌دهد. به صورت دقیق مراحل فوق در شکل ۲-۳ نشان داده شده است.



شکل ۲-۲ مراحل دقیق انجام ایندکس

۲-۳-۲ مرحله دوم: جست و جو اطلاعات:

در این مرحله بر روی ایندکس های ساخته شده جست و جو انجام میدهیم. مراحل جست و جو به شرح زیر است.

- **QueryBuilder و Query**:

بعد از اینکه ایندکس‌ها ساخته شد نوبت به انجام جستجو روی داده‌های ایندکس‌شده است. نکته مهمی که در این‌جا باید مورد توجه قرار گیرد این است که در پیاده‌سازی عملیات جستجو باید از همان نوع آنالیزوری استفاده کنید که در مرحله ایندکس‌کردن اسناد استفاده شده بود. کلاس Query یک کلاس انتزاعی است و برای ساخت نمونه‌ای از آن ابتدا باید نمونه‌ای از کلاس QueryBuilder ایجاد کرد و با استفاده از آن، یکی از انواع زیرکلاس‌های Query را به شیء‌ای از کلاس Query مقید کرد. یکی از زیرکلاس‌هایQuery، کلاس BooleanQuery است که با استفاده از آن می‌توان جستجوی بولی انجام داد. کلاس‌های دیگری مانند WildcardQuery و PhraseQuery و ... هم وجود دارند که متناسب با نامشان جستجو را انجام می‌دهند.

- **IndexReader** :

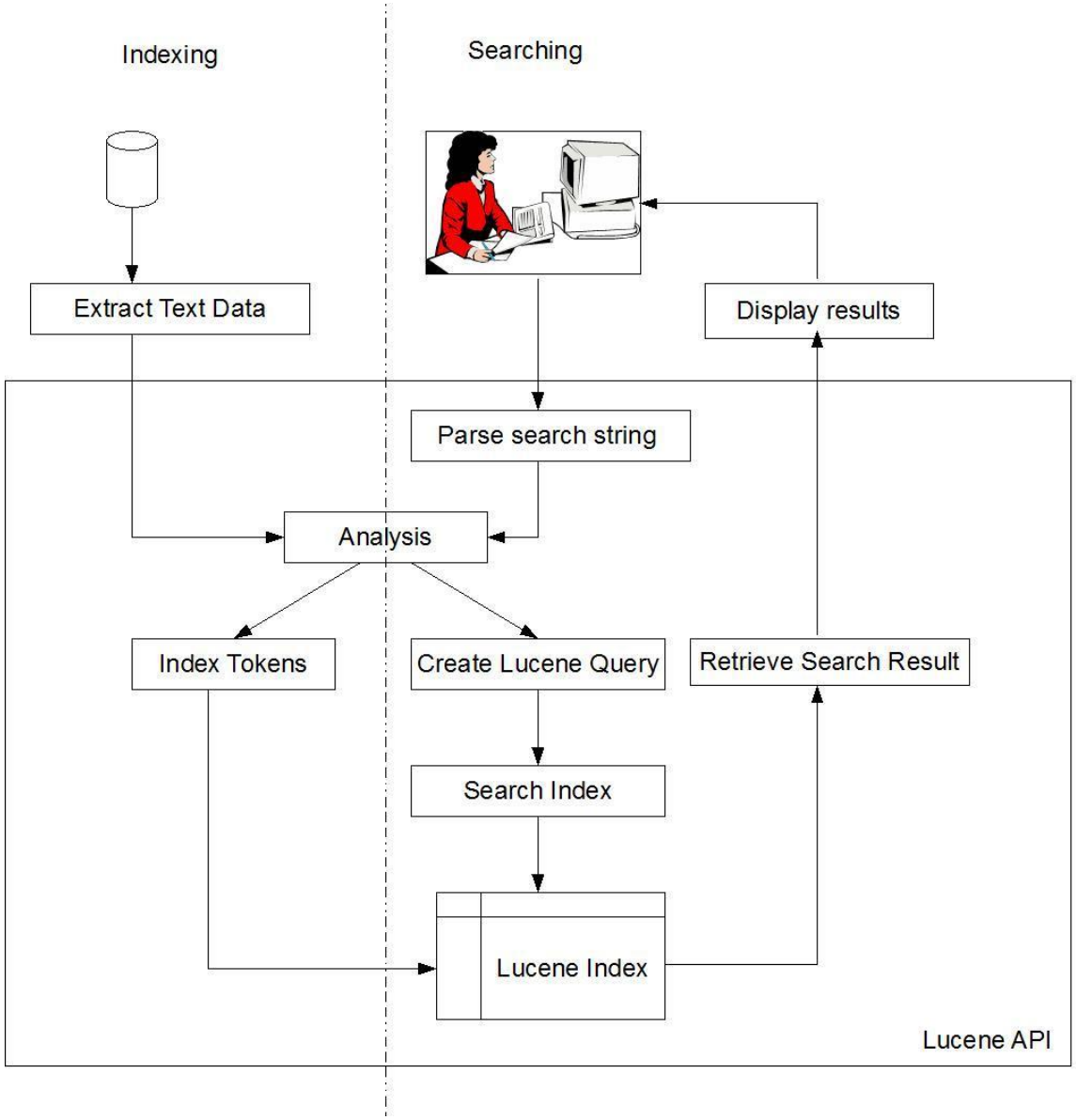
برای اینکه بخواهیم جستجویی روی ایندکس‌ها انجام دهیم ابتدا باید بتوانیم به آن‌ها دسترسی داشته باشیم. کلاس انتزاعی IndexReader این کار را انجام می‌دهد.

- **IndexSearcher** :

با استفاده از کلاسIndexSearcher که نمونه‌ای از کلاس IndexReader را به‌عنوان ورودی سازنده‌اش می‌پذیرد می‌توان جستجو را با استفاده از پرس‌وجویی که قبلاً ساختیم انجام داد.

این متد دو آرگومان ورودی می‌گیرد اولی پرس‌وجوی ماست و دومی حد بالای نتایج بازگشتی را مشخص می‌کند. متد در نهایت نمونه‌ای از کلاس TopDocs را برمی‌گرداند که شامل اسنادی است که پاسخ پرس‌وجوی ما هستند. TopDocs فیلدی بنام []ScoreDoc دارد که خود کلاسی است شامل دو فیلد doc و score، اولی شناسه سند بازیابی شده و دومی رتبه سند می‌باشد.

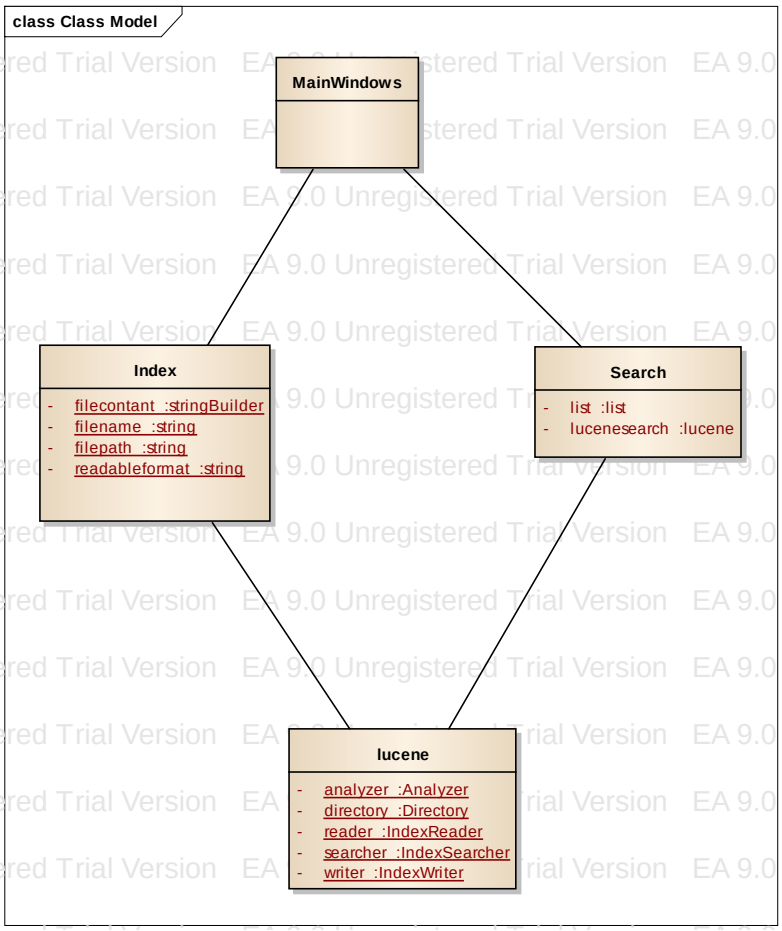
در شکل ۲-۴ مراحل گفته شده بطور دقیق نشان داده شده است.



شکل ۲-۴ مراحل دقیق جست و جو

۲-۴ نمودار کلاس‌ها

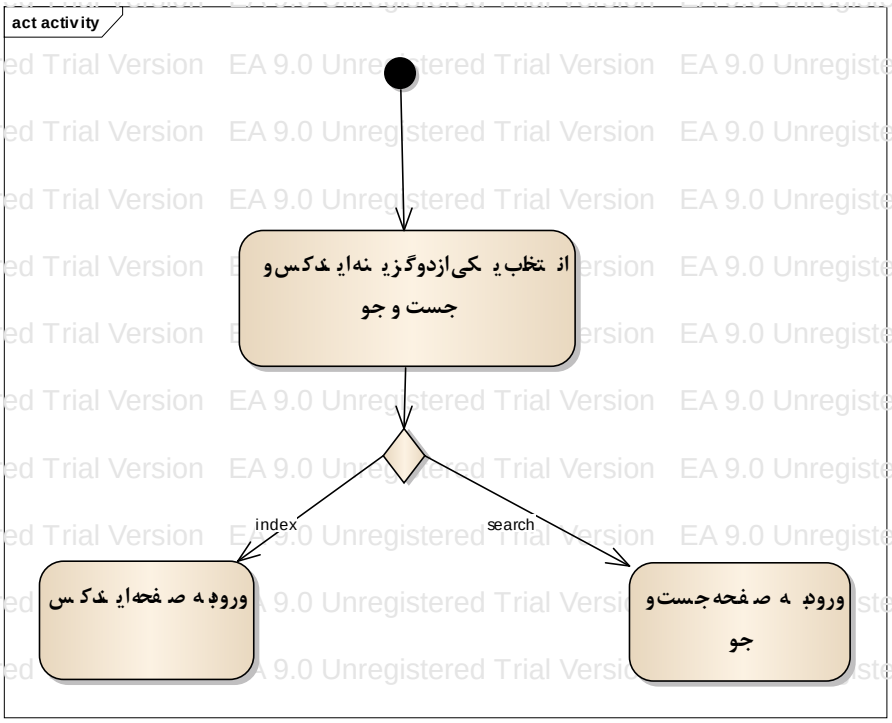
نمودار کلاس این سیستم شامل تمامی کلاس ها و همچنین روابط بین آن ها می باشند. سیستم جاری شامل ۴ کلاس است. در شکل۲-۵ نمودار کلاس ها رسم شده است.



شکل ۲-۵ نمودار کلاس‌ها

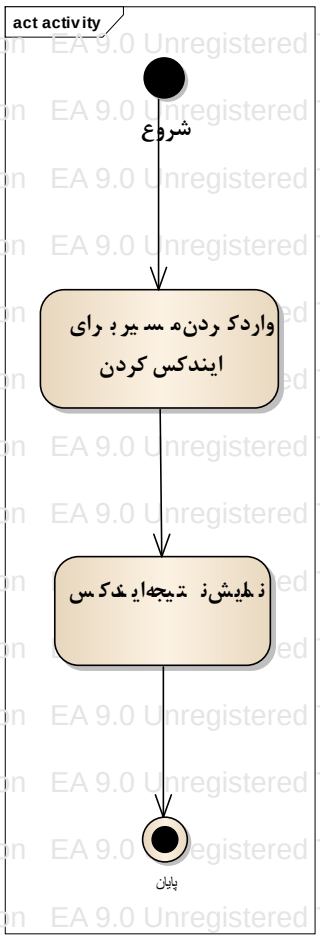
۲-۵ نمودار فعالیت

شکل ۲-۶ نمودار فعالیت ورود به سیستم را نشان میدهد،کاربر بعد ورود به سیستم و انتخاب یکی از گزینه‌های index ویا search وارد فرم مربوطه می‌شود.



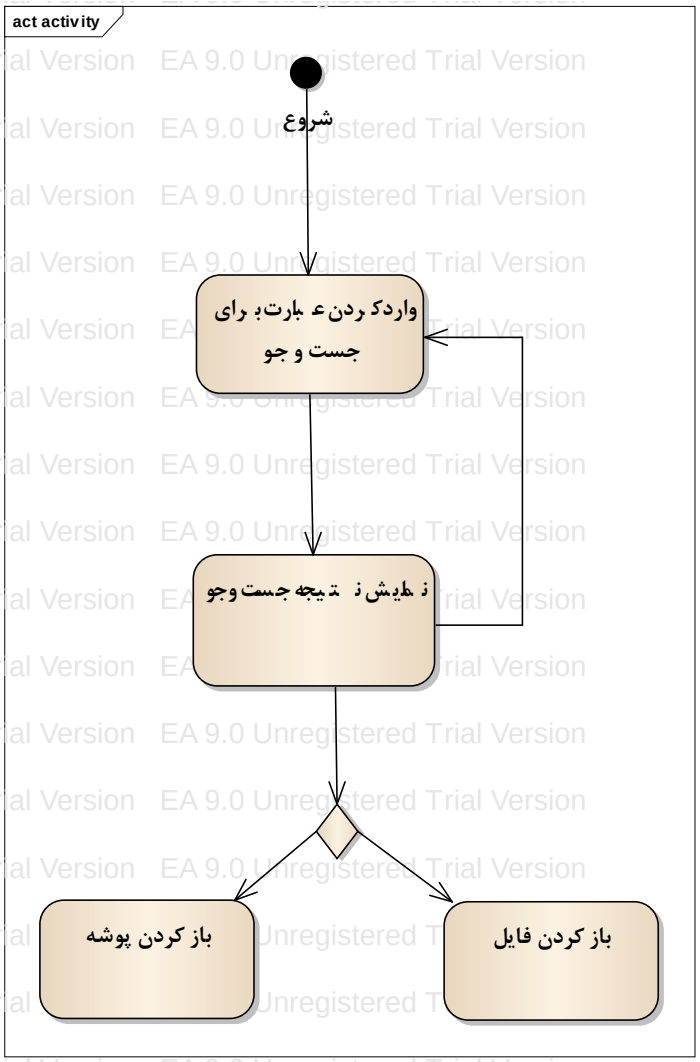
شکل ۲-۶ نمودار فعالیت زیر سیستم mainWindows

شکل ۲-۷ نمودار فعالیت زیر سیستم index را نمایش می‌دهد. کاربر بعد از وارد کردم مسیر سیستم شروع به ایندکس کردن می‌کند



شکل ۲-۷ نمودار فعالیت زیر سیستم index

شکل ۲-۸ نمودار فعالیت زیر سیستم search را نشان می‌دهد. کاربر بعد از با وارد کردن کلمه مورد نظر برای جست و جو نتیجه حاصل از جست و جو نمایش داده می‌شود و کاربر می‌تواند با انتخاب یکی و خود فایل و یا پوشه فایل را باز کند.



شکل ۲-۸ نمودار فعالیت زیر سیستم search

۲-۶ طراحی فرم ها و واسط کاربری:

طراحی فرم ها و واسط کاربری توسط ابزار wpf در C# نوشته شده است که view های برنامه به شکل جدول زیر است.

عملکرد **view**	نام **view**
برای نمایش انتخاب گزینه های index و search	L.I.S
برای وارد کردن مسیر برای index شدن	index
برای وارد کردن query و نمایش حاصل جست و جو	Search

جدول 2-1 Viewهای مربوط به پروژه

فصل سوم

پیاده سازی و تست

۳-۱ معرفی محیط پیاده سازی

زبان برنامه نویسی این سیستم C# و IDE مورد استفاده Visual Studio 2017 است، لازمه استفاده از این IDE سیستم عامل ویندوز است

زبانی که با ان این پروژه را نوشتم C# است. رابطه کاربری مورد استفاده WPF است

۳-۲ پیاده سازی برخی از روال‌های مهم سیستم

الگوریتم های مهمی که در قسمت back-end استفاده شده است در این قسمت توضیح داده می‌شود.

۳-۲-۱ روال lucene

لوسین شامل دو به بخش اصلی Lucene_index و Lucene_search است.

Lucene_index متدی است که محتوا و نام و مسیر فایل را گرفته و در داخل کلاس Document که هر یک از ورودی ها به عنوان یک فیلد از Document هست اضافه می‌کند. هر فیلد را می‌توان با گزینه های متفاوت تنظیم کرد.

گزینه‌ها عبارت اند از:

Store یعنی ایا می‌خواهید ذخیره شود این فیلد یا نه .

Index یعنی ایا می‌خواهید این فیلد را ایندکس شود یا نه.

بعد از این کار، Document حاصل را به Writer داده که ایندکس کند .

```
1 reference | siavash, 2 days ago | 1 author, 2 changes
public bool lucene_index(string path, string name, StringBuilder content)
{
    try
    {
        var doc = new Document();
        doc.Add(new Field("path", path, Field.Store.YES, Field.Index.NO));
        doc.Add(new Field("name", name.ToLower(), Field.Store.YES, Field.Index.ANALYZED));
        doc.Add(new Field("content", content.ToString(), Field.Store.YES, Field.Index.ANALYZED));
        writer.AddDocument(doc);
        return true;
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message.ToString());
        return false;
    }
}
```

شکل۳-۱

Lucene_search متدی است query را از مشتری گرفته آنرا توسط متد parse در کلاس MultiFieldQueryParser تحلیل کرده و جست و جو می کند.

MultiFieldQueryParser کلاسی هست که به ما این اجازه را میدهد که بر روی چندین فیلد همزمان query را اجرا بکنیم.

TopDocs کلاسی هست که این اجازه را میدهد که چند داکیومنت اول را از بین تمامی داکیومنت‌ها نتیجه برداریم.

```
public List<Tuple<string, string>> lucene_search(string query)
{
    try
    {
        MultiFieldQueryParser queryParser = new MultiFieldQueryParser
            (Version.LUCENE_30, new string[] { "name", "content" }, analyzer);
        TopDocs resultDocs = searcher.Search(queryParser.Parse(query), 100);
        var hits = resultDocs.ScoreDocs;

        List<Tuple<string, string>> s = new List<Tuple<string, string>>();
        foreach (var hit in hits)
        {
            var documentfromsearch = searcher.Doc(hit.Doc);
            s.Add(Tuple.Create(documentfromsearch.Get("path"), documentfromsearch.Get("name")));
        }
        searcher.Dispose();
        return s;
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message.ToString());
        return null;
    }
}
```

شکل ۳-۲

۳-۲-۲ روال index

بعد از این که کاربر دکمه‌ی index را زد ابتدا بررسی میکنیم آیا مسیری وارد شده است یا نه و آیا مسیر وارد شده وجود دارد یا نه . اگر شراط گفته شده درست بود وارد بخش خواندن فایل و دادن نام ، محتوا و ادرس آن فایل به متد Lucene_Index در کلاس lucene می‌شود (شکل ۳-۳).

```
index.indexStart();
index_btn.IsEnabled = false;
bool result = false;
string path = browes_tbx.Text;

try
{
    Queue<string> queue = new Queue<string>();
    queue.Enqueue(path);
    while (queue.Count > 0)
    {
        path = queue.Dequeue();
        try
        {
            foreach (string subDir in Directory.GetDirectories(path))
            {
                if (subDir != @"E:\$RECYCLE.BIN" && subDir != @"E:\System Volume Information" && subDir != @"E:\.Trash-1000")
                    queue.Enqueue(subDir);
            }
        }
        catch (Exception ex) { }
        string[] files = null;
        try
        {
            files = Directory.GetFiles(path);
        }
        catch (Exception ex) { }
```

شکل ۳-۳

در بخش ابتدا لوسین را استارت میکنیم بعد تمامی مسیر های فایل و موجود را در لیست files قرار می دهیم. بعد فرمت های این فایل را بررسی می کنیم اگر فایل متنی بود، فایل را باز می کنیم و محتوا و مسیر و نام فایل را به عنوان ورودی به متد lucene_index که در بخش ۱،۲،۳ توضیح داده شده است، میدهیم تا index کند در غیر این صورت اگر متنی نبود فقط مسیر و نام فایل را به عنوان ورودی تابع lucene_index می‌دهیم . در نهایت نتیجه را به متد updateresulttextbox داده تا در خروجی چاپ کند (شکل ۳-۴).

```
if (files != null)
{
    foreach (string file in files)
    {
        try
        {
            fileContent.Clear();
            filename = string.Empty;
            filepath = string.Empty;
            if (readableFormatsList.Contains(Path.GetExtension(file)) && file.Length < (100) * (1024 * 1024))
            {
                string fx = Path.GetExtension(file);
                if (fx == ".doc") { wordReader(file); }
                else if (fx == ".docx") { wordReader(file); }
                else if (fx == ".pdf") { pdfReader(file); }
                else { txtReader(file); }
            }
            else
            {
                fileContent.Append("");
            }
            filename = Path.GetFileName(file);
            filepath = file;
            await Task.Run(() =>
            {
                result = index.lucene_index(filepath, filename, fileContent);
                updateResultTextBox(filename);
            });
        }
```

شکل ۳-۴

طریقه خواندن محتوای فایل در فرمت های مختلف متفاوت هست و باید از کتابخانه های مناسب برای خواندن این فرمت ها استفاده شود.

```
if (readableFormats.Any(Path.GetExtension(file).Contains))
{
    string fx = Path.GetExtension(file);
    if (fx == ".doc") { wordReader(file); }
    else if (fx == ".docx") { wordReader(file); }
    else if (fx == ".pdf") { pdfReader(file); }
    else { txtReader(file); }
}
```

شکل ۳-۴

برای فرمت doc و docx از کتابخانه Code7248.word_reader استفاده شده است. که در آن از کلاس textExtractor برای استخراج محتوا استفاده می‌شود.

برای فرمت pdf از کتابخانه iTextSharp استفاده شده است. که در آن از کلاس pdfReader برای استخراج محتوا استفاده می‌شود.

```
2 references | siavash, 4 days ago | 1 author, 1 change
public void wordReader(string path)
{
    TextExtractor extractor = new TextExtractor(path);
    fileContent.Append(extractor.ExtractText());
}

1 reference | siavash, 4 days ago | 1 author, 1 change
public void pdfReader(string path)
{
    PdfReader reader = new PdfReader(path);
    for (int page = 1; page <= reader.NumberOfPages; page++)
    {
        fileContent.Append(PdfTextExtractor.GetTextFromPage(reader, page));
    }
    reader.Close();
}
```

شکل ۳-۵

برای فرمت های دیگر از قبیل سورس کدها فایل txt و xml از کلاس موجود در کتابخانه system.io به نام streamReader استفاده می‌کنیم.

```
1 reference | siavash, 1 day ago | 1 author, 4 changes
public void txtReader(string path)
{
    try
    {
        using (StreamReader reader = new StreamReader(path))
        {
            fileContent.Append(reader.ReadToEnd());
        }
    }
    catch (Exception ex)
    {
        fileContent.Append("");
    }
}
```

شکل ۳-۶

۳-۲-۳ روال Search

کاربر بعد از وارد کردن query و کلیک بر روی دکمه search ، یک شی از کلاس lucene میسازیم. و query ای را که کاربر وارد کرده به ورودی تابع lucene_search می‌دهیم و نتیجه در داخل یک لیست می‌ریزیم و ان را به کار بر در صفحه نمایش میدهیم.

```
private void search_btn_Click(object sender, RoutedEventArgs e)
{
    List<Tuple<string, string>> list;
    List<listitem> l = new List<listitem>();
    lucene lucenesearch = new lucene();
    lucenesearch.searchStart();
    list = lucenesearch.lucene_search(search_tbx.Text.ToLower());
    if (list != null)
    {
        foreach (var item in list)
        {
            listitem li = new listitem();
            li.name = item.Item2;
            li.path = item.Item1;
            l.Add(li);
        }

        result_ltv.ItemsSource = l;
        lucenesearch.searchClose();
    }
```

شکل ۳-۷

۳-۳ نحوه تست سیستم

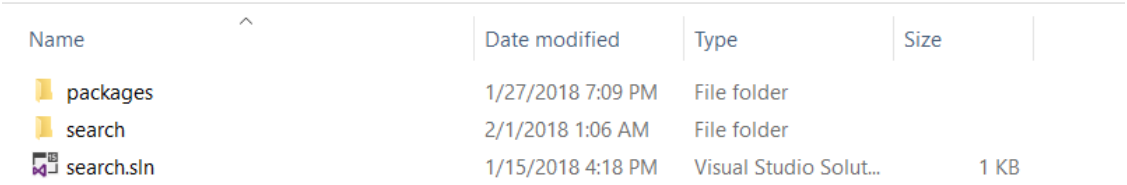
تست سیستم به صورت دستی است، به این صورت که بعد از اتمام هر قسمت با پر کردن اطلاعات به انواع روش هایی که احتمال داده شد کاربر وارد کند، اطلاعات را وارد کرده و پاسخ سیستم را مشاهده کرده و در صورت وجود ایراد آن را برطرف کردیم.

فصل چهارم
راهنمای کاربری

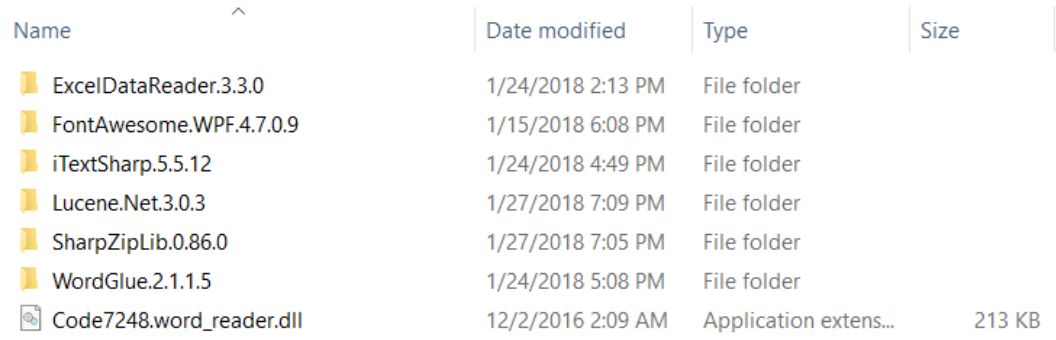
۴-۱ ساختار کدهای متن

در پوشه اصلی کدهای متن پروژه تعدادی پوشه و فایل وجود دارد. پوشه‌ای که با نام packages شروع می‌شود کتابخانه‌هایی هستند که ما از طریق Nuget در پروژه نصب کرده ایم (شکل ۴-۱)، با باز کردن فایل packeges که درون پوشه اصلی قرار دارد کل کتابخانه هایی که از طریق Nuget نصب کرده‌ایم دیده می‌شود(شکل ۴-۲).

با باز کردن فایل search.sln کل پروژه در محیط Visual Studio باز خواهد شد و از آنجا می توان به تمامی کدها دسترسی پیدا کرد.



شکل ۴-۱

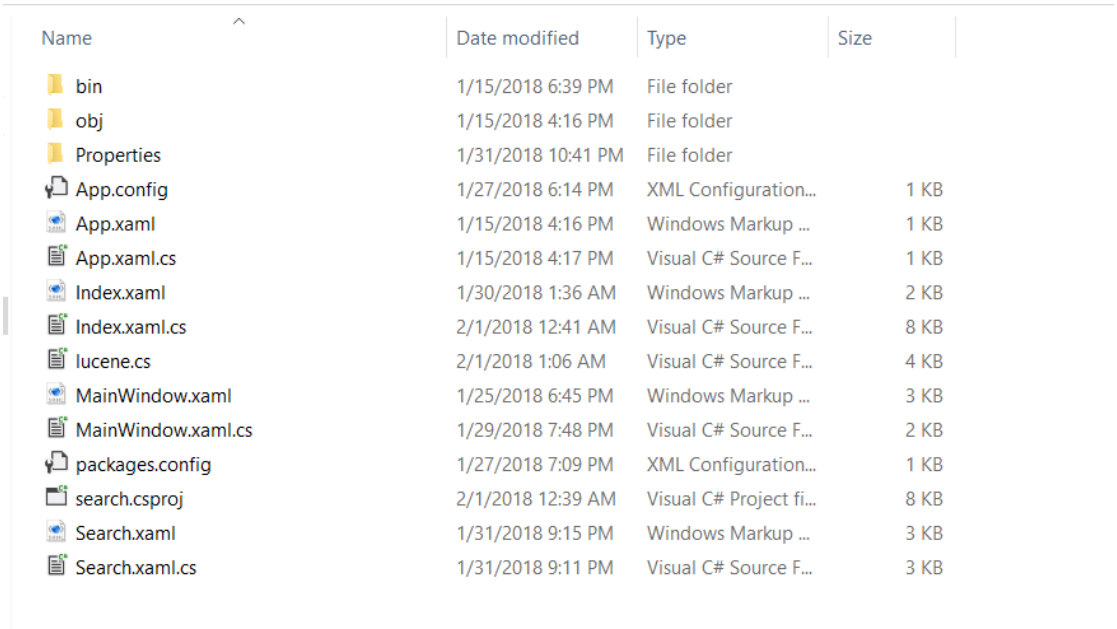


شکل ۴-۲

فرم ها و واسط های کاربری و کلاس ها در پوشه ی search (شکل ۴-۳) قرار دارد.

در پوشه search فایل هایی با فرمت xaml فرم ها و رابطه کاربری پروژه هستند. همچنین فایل هایی با فرمت xaml.cs. کلاس های کد های C# این فرم‌ها هستند.

فایل با فرمت cs خالی کلاس پروژه هست که توست خود ما ساخته شده است.



Name	Date modified	Type	Size
bin	1/15/2018 6:39 PM	File folder	
obj	1/15/2018 4:16 PM	File folder	
Properties	1/31/2018 10:41 PM	File folder	
App.config	1/27/2018 6:14 PM	XML Configuration...	1 KB
App.xaml	1/15/2018 4:16 PM	Windows Markup ...	1 KB
App.xaml.cs	1/15/2018 4:17 PM	Visual C# Source F...	1 KB
Index.xaml	1/30/2018 1:36 AM	Windows Markup ...	2 KB
Index.xaml.cs	2/1/2018 12:41 AM	Visual C# Source F...	8 KB
lucene.cs	2/1/2018 1:06 AM	Visual C# Source F...	4 KB
MainWindow.xaml	1/25/2018 6:45 PM	Windows Markup ...	3 KB
MainWindow.xaml.cs	1/29/2018 7:48 PM	Visual C# Source F...	2 KB
packages.config	1/27/2018 7:09 PM	XML Configuration...	1 KB
search.csproj	2/1/2018 12:39 AM	Visual C# Project fi...	8 KB
Search.xaml	1/31/2018 9:15 PM	Windows Markup ...	3 KB
Search.xaml.cs	1/31/2018 9:11 PM	Visual C# Source F...	3 KB

شکل ۴-۳

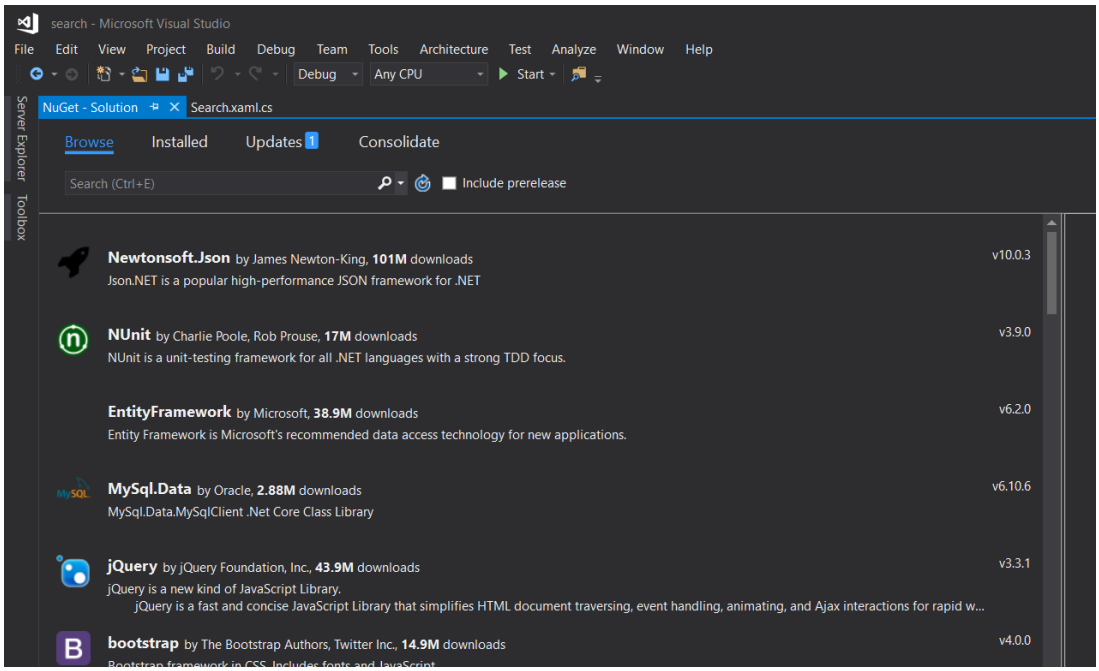
در پوشه search ، پوشه‌ای به نام Properties قرار دارد که همه ی config ها و تنضیمات پروژه در آن قرار دارد (شکل ۴-۴). محتویات این پوشه که به صورت پیش فرض، خود C# آن ها را ایجاد و کنترل میکند، مگر این که نیاز به تنظیمات خاصی باشد. ما نباید چیزی را حذف یا اضافه کنیم و یا محتویات آن ها را تغییر بدهیم در صورت نیاز از داخل خود visual studio تغیر میدهیم.

Name	Date modified	Type	Size
app.manifest	1/31/2018 10:41 PM	MANIFEST File	4 KB
AssemblyInfo.cs	1/15/2018 4:17 PM	Visual C# Source F...	3 KB
Resources.Designer.cs	1/15/2018 4:17 PM	Visual C# Source F...	3 KB
Resources.resx	1/15/2018 4:16 PM	Microsoft .NET Ma...	6 KB
Settings.Designer.cs	1/15/2018 4:17 PM	Visual C# Source F...	2 KB
Settings.settings	1/15/2018 4:16 PM	Settings-Designer ...	1 KB

شکل ۴-۴

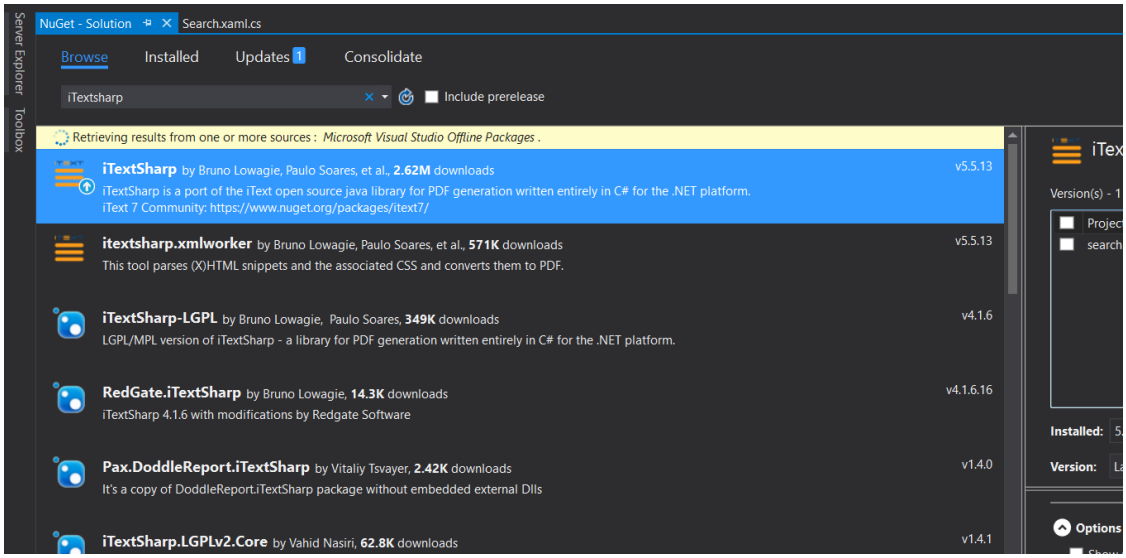
۴-۲ نحوه کامپایل

برای کامپایل ابتدا فایل search.sln را باز کرده سپس از تب tools، nugget package manager را زده سپس گزینه manage nugget packeges for solution... را انتخاب میکنیم . با باز شدن تب جدید مطابق شکل ۴-۵ با جست و جو و نصب کتاب خانه های زیر در تب Browse برنامه را آماده اجرا میکنیم.



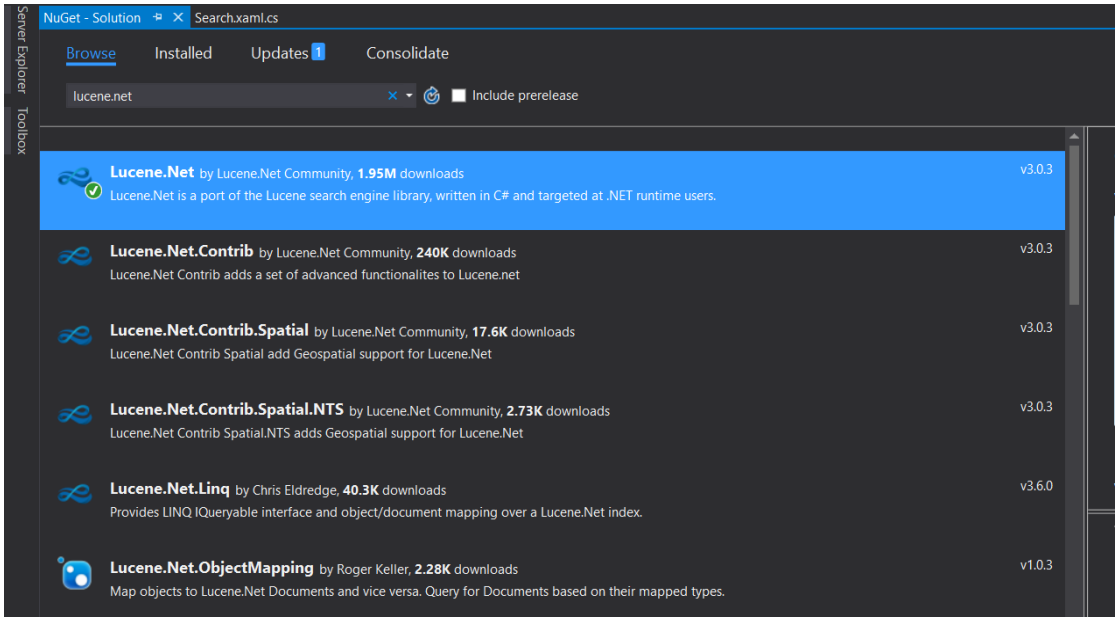
شکل ۴-۵

برای نصب کتاب خانه itextsharp برای خواندن متن های فایل های pdf، در تب Browse ، itextsharp را جست و کرده و نتیجه طبق شکل ۴-۶ می‌شود. با انتخاب itextsharp آن را نصب می‌کنیم.



شکل ۴-۶

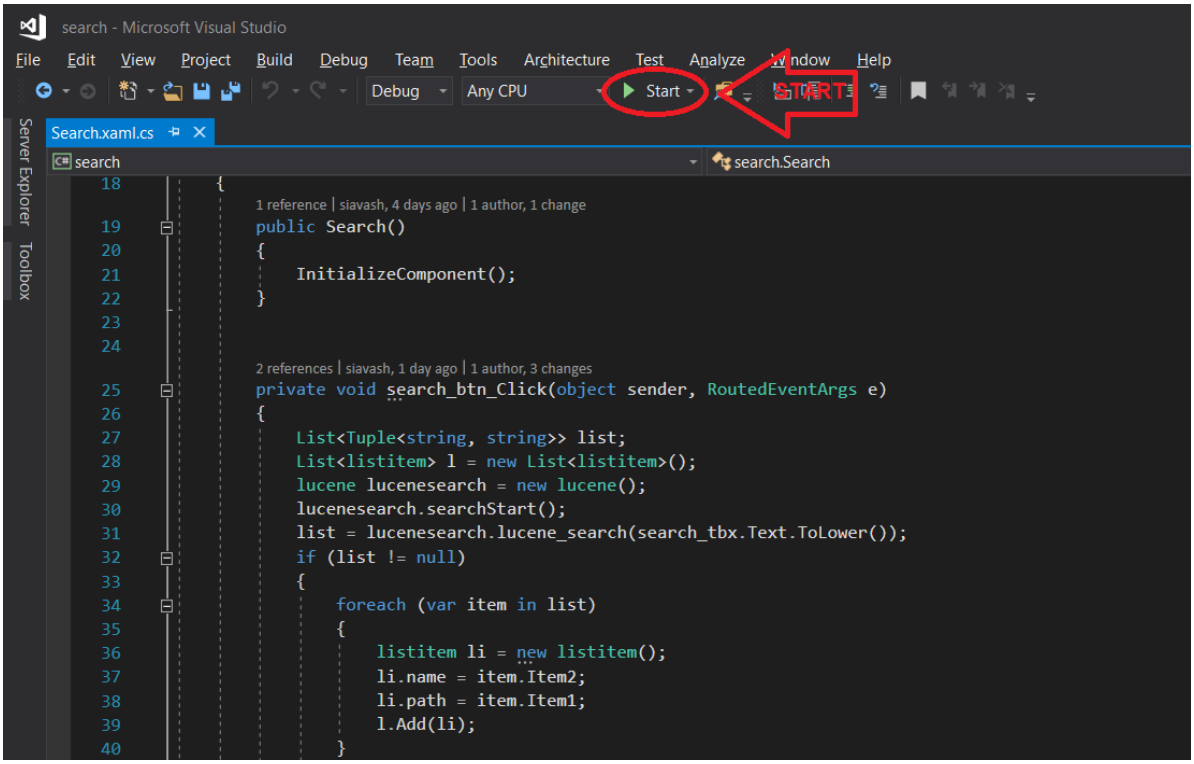
برای نصب lucene در تب Browse جست و جو میکنیم و نصب می‌کنیم مطابق شکل ۴-۷.



شکل ۴-۷

برای نصب کتابخانه های fontawesome.wpf برای ایکن در برنامه استفاده شده است و word_reader. برای خواندن محتویات فایل های word office مطابق کتابخانه های بالا در تب Browse جست و جو کرده و نصب می‌کنیم.

حال دکمه start را مطابق شکل ۴-۸ زده تا برنامه اجرا شود.



شکل ۴-۸ کامپایل برنامه

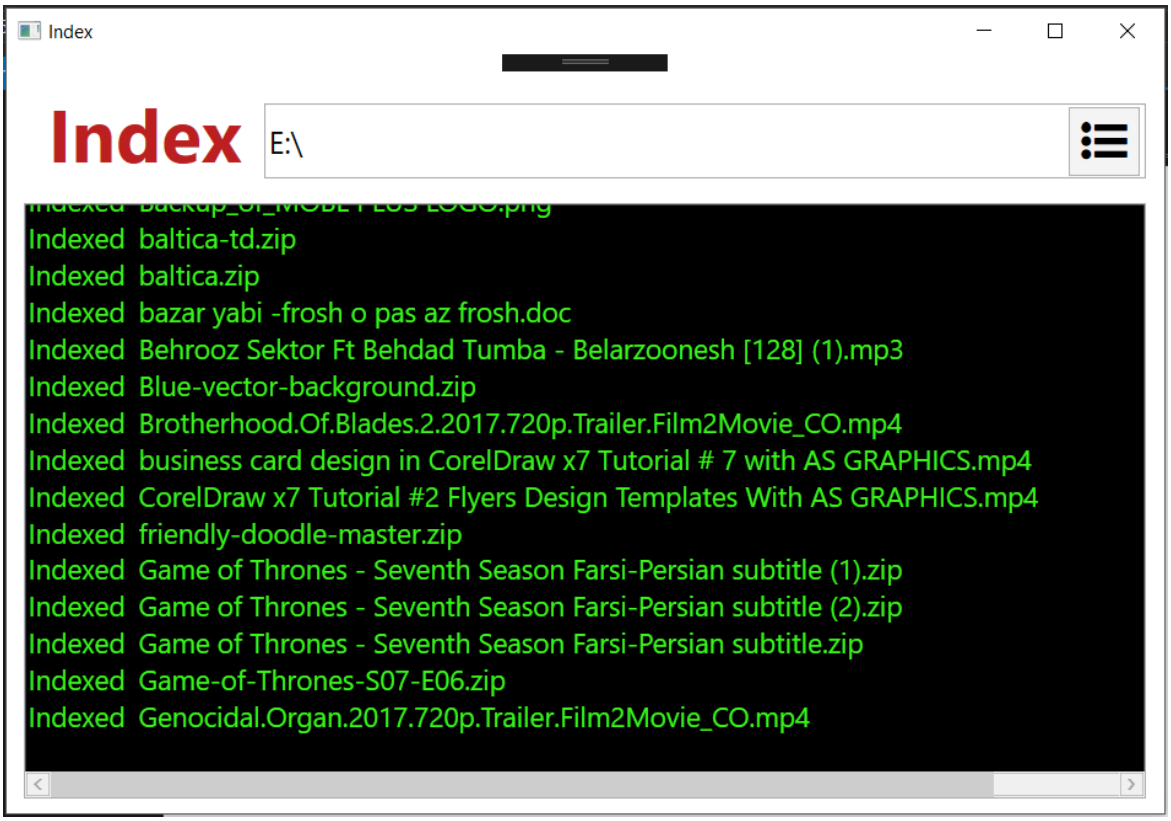
۴-۳ راهنمای استفاده از سیستم

زمانی که وارد برنامه شدیم دو انتخاب خواهیم داشت search و index .

با انتخاب گزینه search وارد صفحه جست و جو می‌شویم و با انتخاب دکمه index وارد صفحه ایندکس(شکل ۴-۹).

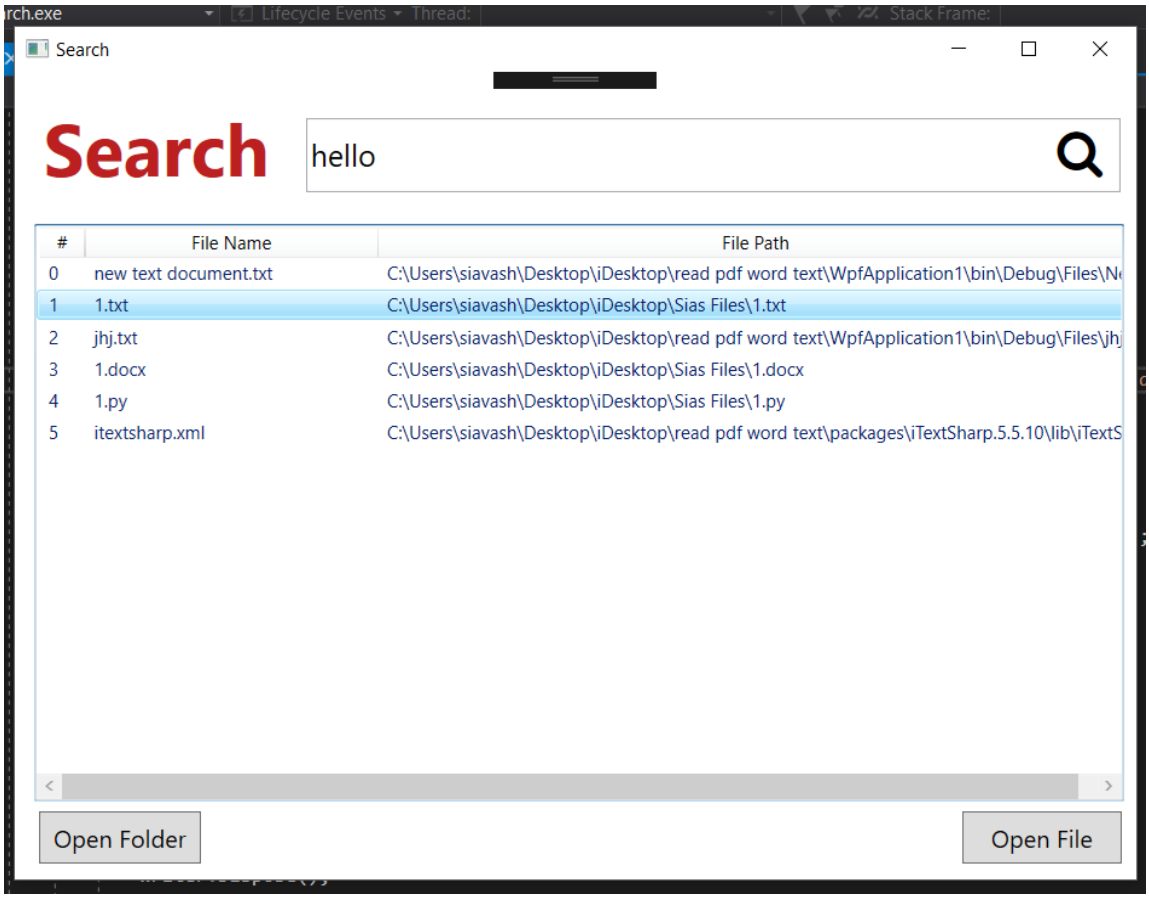
شکل ۴-۹

با انتخاب دکمه index صفحه جدیدی باز می‌شود که ما با وارد کردن مسیری که می‌خواهیم index شود و زدن دکمه ایندکس برنامه شروع به ایندکس شدن می‌کند (شکل ۴-۱۰).



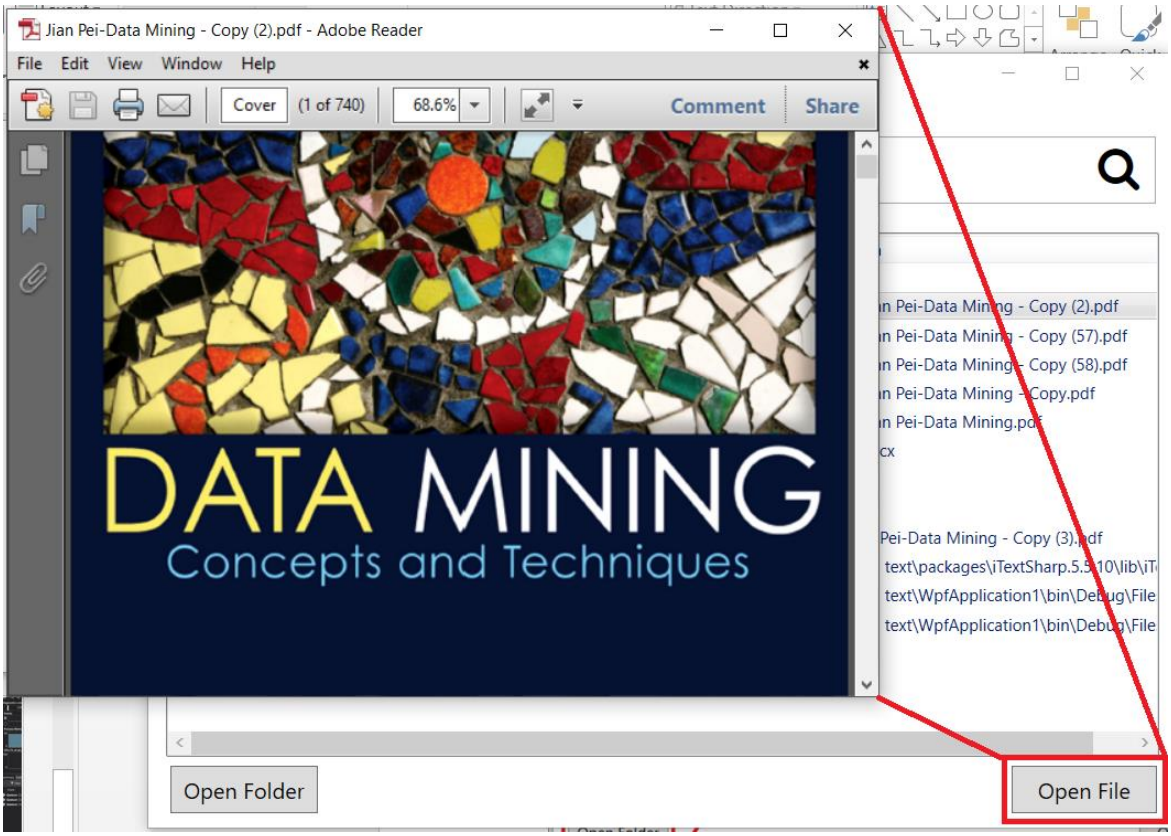
شکل ۴-۱۰

در صفحه اول با انتخاب گزینه search وارد صفحه جست و جو می‌شویم و در قسمت جست و جو query ای که مد نظرمان هست را نوشته و جست و جو می‌کنیم . و نتایج در صورت وجود در زیر به صورت لیست وار نمایش داده می‌شود. با انتخاب بر روی یکی از نتایج و زدن یکی از دکمه های open file و یا open folder به ترتیب فایل را باز کنیم و یا پوشه ای که فایل در آن قرار دارد را باز کنیم (شکل ۴-۱۱).

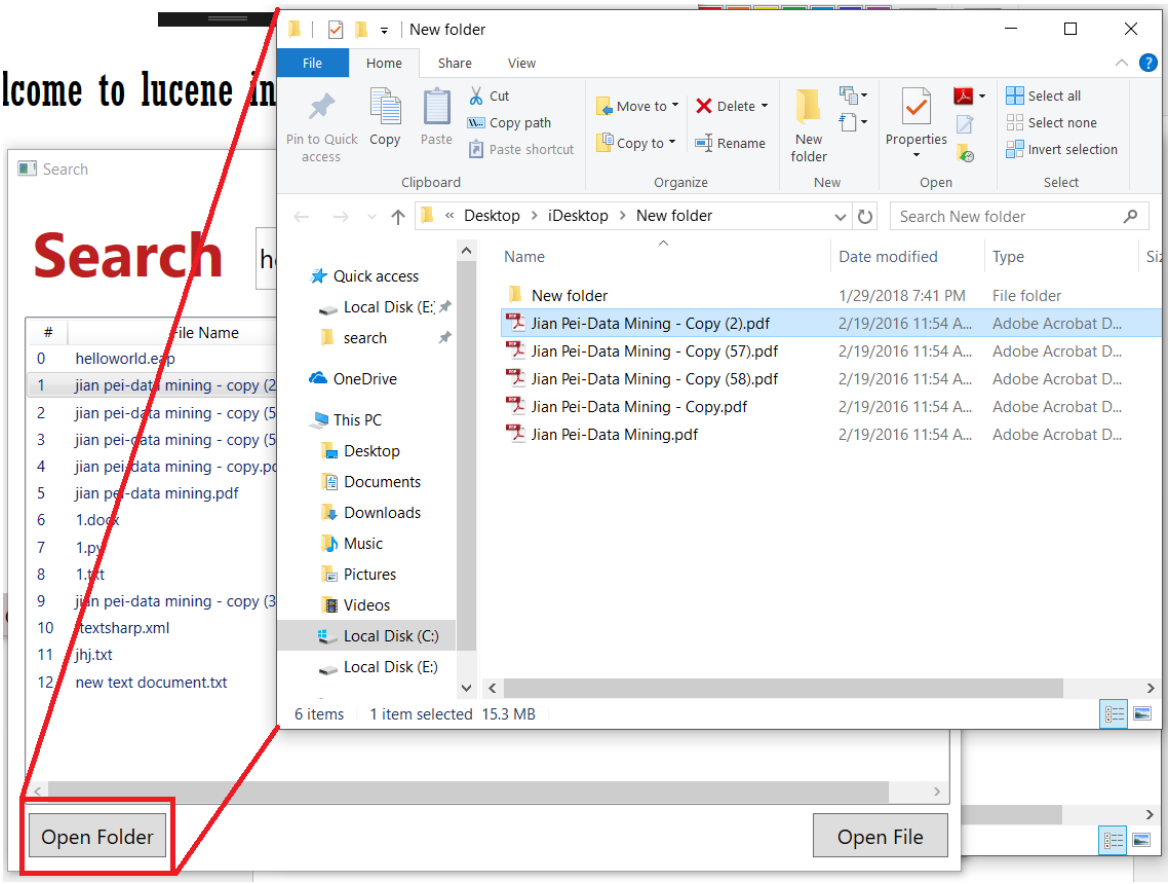


شکل ۴-۱۱

با انتخاب بر روی یکی از نتایج و زدن یکی از دکمه های open file (شکل ۴-۱۲) و یا open folder (شکل ۴-۱۳) به ترتیب فایل را باز می‌کنیم و یا پوشه ای که فایل در آن قرار دارد را باز کنیم .


Jian Pei-Data Mining - Copy (2).pdf - Adobe Reader
File Edit View Window Help
Cover (1 of 740) 68.6% Comment Share
DATA MINING
Concepts and Techniques
Open Folder
Open File

شکل ۴-۱۲


lcome to lucene in
Search
Search
File Name
0 helloworld.cpp
1 jian pei-data mining - copy (2
2 jian pei-data mining - copy (5
3 jian pei-data mining - copy (5
4 jian pei-data mining - copy.pd
5 jian pei-data mining.pdf
6 1.docx
7 1.py
8 1.txt
9 jian pei-data mining - copy (3
10 textsharp.xml
11 jhj.txt
12 new text document.txt
Open Folder
Open File
New folder
File
Home
Share
View
Cut
Copy path
Paste shortcut
Pin to Quick access
Copy
Paste
Move to
Copy to
Delete
Rename
New folder
Properties
Select all
Select none
Invert selection
Clipboard
Organize
New
Open
Select
Desktop › iDesktop › New folder
Search New folder
Quick access
Local Disk (E:)
search
OneDrive
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (E:)
Name
Date modified
Type
New folder 1/29/2018 7:41 PM File folder
Jian Pei-Data Mining - Copy (2).pdf 2/19/2016 11:54 A... Adobe Acrobat D...
Jian Pei-Data Mining - Copy (57).pdf 2/19/2016 11:54 A... Adobe Acrobat D...
Jian Pei-Data Mining - Copy (58).pdf 2/19/2016 11:54 A... Adobe Acrobat D...
Jian Pei-Data Mining - Copy.pdf 2/19/2016 11:54 A... Adobe Acrobat D...
Jian Pei-Data Mining.pdf 2/19/2016 11:54 A... Adobe Acrobat D...
6 items
1 item selected 15.3 MB

شکل ۴-۱۳

فصل پنجم

نتیجه گیری

نتیجه گیری

از آنجایی که پیاده سازی نرم افزار جست و جو استفاده جهانی داشته باشد نیازمند تیمی قوی برای تحلیل و طراحی است و زمان زیادی است، ممکن است به دلیل کمبود تعداد نفرات و کمبود وقت در این پروژه، نتیجه در بعضی جهات کامل نباشد ولی نتیجه کلی بسیار امیدوار کننده بوده و می توان از این پروژه برای پیاده سازی نرم افزار های بسیار کاراتر از این نرم افزار استفاده کرد. در نهایت بیشترین سود از این پروژه برای اینجانب تجربه بود که در این راه به دست آمده که مطمئنا در آینده نزدیک از تجارب کسب شده در این پروژه برای توسعه اپلیکیشن ها و برنامه هایی بهتر استفاده خواهم کرد. یک تجربه مهم نیز که در این پروژه کسب کردم این بود همیشه خیلی زود تر شروع کرده و به توانایی های خود زیاد بسنده نکنیم که باید وقت نسبتا زیادی روی تحلیل سیستم گذاشت تا به بهترین و کامل ترین تحلیل برسیم که منجرب به بهترین طراحی و پیاده سازی می‌شود. تحلیل کامل باعث صرفه جویی در وقت در مراحل بعدی پروژه و بهینه شدن کد ها و کارایی سیستم می‌شود.

منابع و مراجع

[1] وبسایت http://summits.ir

[2] وبسایت http://www.dotnettips.info