

به نام خدا
گزارش کار واحد کارآموزی



دانشگاه اصفهان
دانشکده مهندسی کامپیوتر

کارآموز: محمد حسین هوشمند

گرایش: مهندسی نرم افزار

محل کارآموزی : شرکت مهندسی پیام پرداز

استاد راهنما: دکتر بهروز ترک لادانی

مسئول کارآموزی: سید مجتبی جعفری قمصری

تابستان ۱۴۰۱

فهرست مطالب

۱	۱- معرفی محل و موضوع کارآموزی
۲	۲- فهرست کارهای انجام شده
۳	۳- ارزیابی کارآموزی
۴	۴- گزارش‌های فنی و پیوست‌ها
۵	۴-۱- آشنایی با افزونه‌ها
۱۴	۴-۳- طراحی افزونه رمزنگار با دستکاری بایتی
۱۷	۴-۴- آشنایی با پروتوکل رمزنگاری AES
۱۸	۴-۵- طراحی افزونه‌ی رمزنگار متن
۲۶	۴-۶- امکان بهبود و ارتقای افزونه
۲۷	منابع استفاده شده

۱- معرفی محل و موضوع کارآموزی

شرکت پیام پرداز شرکتهی با سابقه در حوزهی امنیت اطلاعات و ارتباطات می باشد که تاکنون محصولات زیادی در این زمینه تولید کرده است. هسته ی داخلی این شرکت توسط اساتید دانشگاه اصفهان و دانشگاه صنعتی اصفهان در سال ۱۳۷۵ تشکیل شده است.

با توجه به اهمیت مسئله ی امنیت و رمزنگاری در حوزه های نرم افزاری، این شرکت توانسته است محصولات زیادی را با توجه به شرایط کشور بومی سازی و نگهداری کند. غالب مشتریان این شرکت سازمان ها و نهادهای دولتی هستند که به عنوان نمونه می توان به شرکت هایی چون ایرانسل، رایتل و ایران خودرو اشاره کرد.

از جمله محصولاتی که در این شرکت تولید و نگهداری می شود، می توان به محصولاتی مانند نارین، رایمون، کیهان و رایان اشاره کرد. این محصولات به طور مداوم در حال توسعه هستند و روز به روز امکانات جدیدتری به آنها اضافه می شود. نهایتاً این محصولات به بازار معرفی می شوند و سازمان ها یا نهادهای دولتی می توانند آنها را خریداری کنند. محصولی که در این دوره کارآموزی مورد بررسی و توسعه قرار می گیرد، محصول رایان می باشد که وظیفه ی آن حفاظت و کنترل اسناد سازمانی می باشد. نگهداری و حفاظت از اسناد حیاتی خصوصاً اسناد محرمانه، همواره یکی از دغدغه های مهم مدیران بوده است. این اسناد و مدارک، باید تحت پروتکل ها و ساختارهایی رمزنگاری یا حفاظت شوند تا مورد سوءاستفاده قرار نگیرند یا از سازمان مذکور خارج نشوند. به عنوان مثال اسناد و مدارک می توانند از راه های زیادی مانند شبکه های اجتماعی، عکس برداری، ایمیل و... انتشار پیدا کند. لذا سامانه رایان به منظور ارائه خدماتی در زمینه ی حفاظت از اسناد و مدارک، سعی می کند این کار را به صورت نامحسوس از دید کاربر و بدون تغییر قوانین سازمانی شرکت مورد نظر انجام دهد.

به عنوان مثال، اگر اسناد در فایل های حفاظت شده ای تحت عنوان (Hot folder) ذخیره شده و همزمان محصول رایان بر روی سیستم نصب شود، محدودیت های حفاظتی برنامه رایان، بر روی فایل های حفاظتی اعمال می شود. به عنوان مثال گزینه ی Share در فایل های Word غیرفعال می شود. یا زمانی که از صفحه Screenshot گرفته می شود، عکس چیزی جز یک صفحه ی سیاه نمایش نمی دهد. نکته قابل توجه آن است که این محصول از رابط کاربری خاصی استفاده نمی کند و تمام این اتفاقات به طور نامحسوس و پس از نصب نرم افزار اعمال می شود.

مسئله ای که در این دوره ی کارآموزی قرار است مورد بررسی و پیاده سازی قرار بگیرد، مسئله ی افزونه ها یا (Add in) است که در نرم افزارهای بسیاری به شیوه های مختلف به کار گرفته می شود. افزونه ها درواقع برنامه ها

کدهایی جانبی هستند که می‌توانند قابلیت‌های جدیدی را به یک محصول اضافه کنند. در ادامه به توضیحات بیشتری از Add in پرداخته می‌شود.

آدرس محل کارآموزی: اصفهان- خیابان پروین اعتصامی - نبش خیابان عسگریه - پلاک ۳۴۶

وبسایت: www.payampardaz.com

۲- فهرست کارهای انجام شده

در این دوره، مسئله اصلی که مورد توجه و مطالعه قرار گرفت، مسئله‌ی افزونه‌ها برای فایل آفیس خصوصاً فایل‌های Word است. این مسئله از آن جایی اهمیت دارد که تقریباً اکثر فایل‌ها و اسناد، در قالب محصولات این شرکت استفاده می‌شود. از آنجایی که ماهیت و ساختار محصولات آفیس تا حد زیادی به هم نزدیک هستند، لذا تجربیات این دوره را می‌توان به سایر محصولات شرکت مایکروسافت مانند Excel یا PowerPoint تعمیم داد. در ادامه خلاصه‌ای از کارهایی که در این دوره انجام شد ذکر می‌گردد.

آشنایی با افزونه‌ها

در چند هفته‌ی ابتدایی به طور دقیق در رابطه‌ی افزونه‌ها و کارکردشان تحقیق و بررسی صورت گرفت. سپس به طور خاص در نرم‌افزارهای Word ، PowerPoint و Excel این مسئله مورد بررسی قرار گرفت.

شناخت محیط توسعه و انواع پروژه‌های افزونه

در ادامه محیط Visual Studio 2019 بر روی سیستم نصب می‌شود و با حالات مختلف ایجاد یک پروژه‌ی افزونه آشنایی صورت می‌گیرد. لازم به ذکر است که محیط Visual Studio 2015 در حال حاضر از امکانات توسعه‌ی ادین پشتیبانی نمی‌کند.

ایجاد یک افزونه‌ی ساده برای فایل Word

پس از مهیا شدن شرایط ایجاد پروژه، اولین پروژه افزونه برای فایل‌های Word، ایجاد می‌شود. در این پروژه، افزونه با هر بار بسته شدن یک فایل Word، جمله‌ای را به ابتدای فایل متنی اضافه می‌کند

ایجاد یک افزونه‌ی قفل‌کننده‌ی فایل Word

در این پروژه، از طریق دستکاری بایت‌های هدر فایل‌های Word، سعی می‌شود به طور غیر مستقیم، فایل‌ها را قفل کرد.

آشنایی با پروتوکل AES برای رمزنگاری

به منظور رمزکردن فایل‌های متنی Word، لازم به آشنایی با این پروتوکل بود.

ایجاد یک افزونه‌ی رمزنگار متنی فایل Word

در این پروژه، یک رمزنگار متنی به کمک پروتوکل AES ایجاد می‌شود.

تلاش برای بهبود و ارتقای افزونه رمزگار

جهت رمزنگاری حداکثری فایل‌های Word تحقیقاتی صورت گرفت و همچنین مواردی که افزونه‌ی رمزنگار می‌تواند بهبود پیدا کند، مورد ارزیابی قرار گرفت.

۳- ارزیابی کارآموزی

قطعا مهم‌ترین تجربه‌ای که کارآموز در این دوره کسب می‌کند، آشنایی و لمس محیط کار می‌باشد. اینکه در محیط کارآموزی اول از همه بایستی به قوانین سازمانی محل کار احترام گذاشته شود و کارآموزان و کارمندان از اختلال و بی‌نظمی در محیط کار پرهیز نمایند. رعایت اخلاق و رفتار حرفه‌ای در بین کارمندان شرکت نکته‌ی قابل توجهی بود که در رابطه با آن تاکید بسیاری می‌شود. همچنین برخی از مواردی که در درس مهندسی نرم‌افزار بر آن تاکید می‌شود، در محیط کار قابل مشاهده است که استفاده از متودولوژی اسکرام در یکی از گروه‌ها، نمونه‌ای از این موارد است. همچنین کارگاه‌هایی در زمینه‌های کدنویسی حرفه‌ای اهمیت رفتار حرفه‌ای در محل کار تشکیل گشت که برای کارآموز بسیار مفید و جالب بود.

از لحاظ فنی، با توجه به اینکه کارآموز تجربه‌ای در زمینه‌ی برنامه‌نویسی C# نداشته است، توانسته در این دوره به خروجی‌های قابل قبولی در زمینه‌ی افزونه‌نویسی دست پیدا کند. خروجی کار نهایتاً به ساخت یک افزونه‌ی

رمزنگار متنی می‌باشد. هرچند در ادامه تلاش بر این شد که این رمزنگاری در لایه‌های پایین‌تر فایل انجام گردد تا عکس‌ها و جداول و فونت‌ها رمزنگاری شود. در این راستا این پروژه می‌تواند به ایده‌آل نزدیک‌تر شود. همچنین سعی شده است نکات مبتنی بر کدنویسی تمیز در کدها رعایت گردد.

در مجموع این محیط کارآموزی به کسانی که در زمینه‌های امنیتی علاقه‌مند هستند به شدت پیشنهاد می‌شود. همچنین این مکان می‌تواند محیط مناسبی برای افرادی باشد که به دنبال کسب رزومه خوب هستند. فضای آرام و جذاب در کنار کادر محترم از نکات شاخص این محیط است که می‌تواند زمینه‌ی خوبی را برای فعالیت کارآموز فراهم نماید.

۴- گزارش های فنی و پیوست‌ها

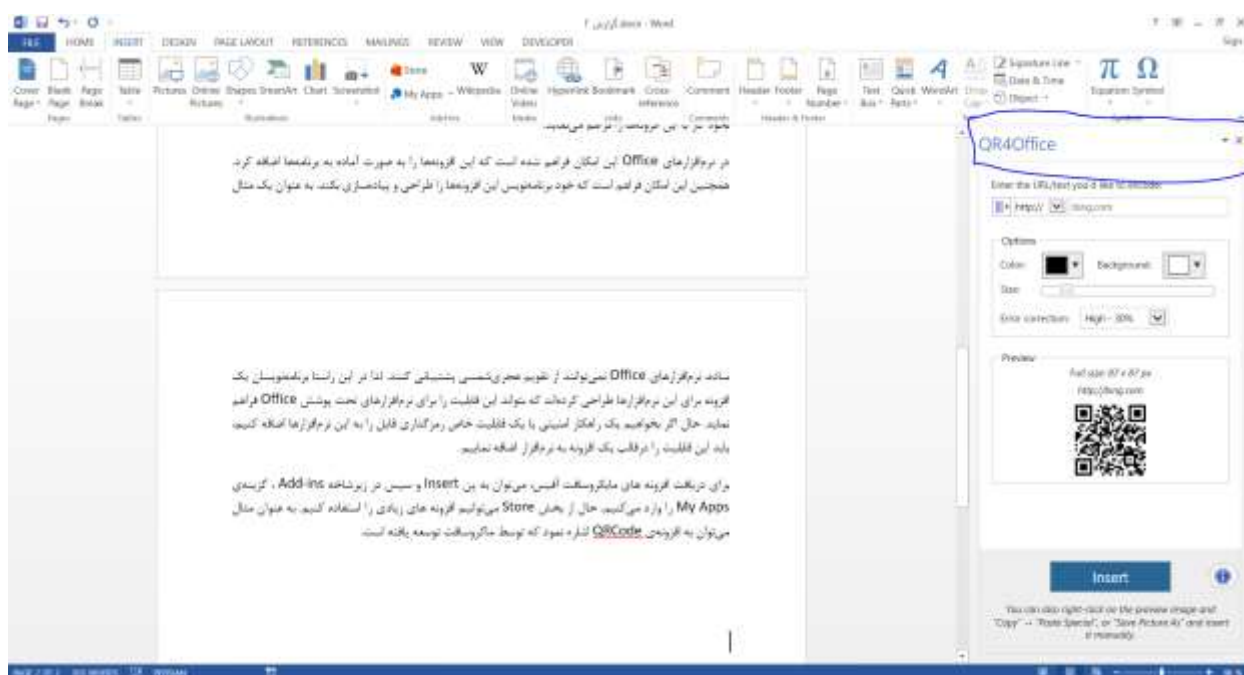
۴-۱- آشنایی با افزونه‌ها

همانطور که پیش‌تر اشاره شد، افزونه‌ها در واقع کدها و برنامه‌هایی جانبی هستند که می‌توانند قابلیت‌های جدیدی را به نرم‌افزار اضافه کند. از آنجایی که بیشتر اسناد و مدارک سازمانی در قالب فایل‌هایی چون word و Excel که تحت پوشش شرکت ماکروسافت هستند نگهداری می‌شوند، حوزه‌ی فعالیت در زمینه‌ی بررسی و تحلیل افزونه‌ها در این نرم‌افزارها می‌باشد. در این راستا مقاله‌های بسیاری در این زمینه نوشته شده است که نمونه‌ی معتبر آن در سایت ماکروسافت وجود دارد که اطلاعات مهمی در زمینه‌ی آموزش و نحوه‌ی کار با این افزونه‌ها را فراهم می‌نماید.

در نرم‌افزارهای Office این امکان فراهم شده است که این افزونه‌ها را به صورت آماده به برنامه‌ها اضافه کرد. همچنین این امکان فراهم است که خود برنامه‌نویس این افزونه‌ها را طراحی و پیاده‌سازی بکند. به عنوان یک مثال ساده، نرم‌افزارهای Office نمی‌توانند از تقویم هجری شمسی پشتیبانی کنند. لذا در این راستا برنامه‌نویسان یک افزونه برای این نرم‌افزارها طراحی کرده‌اند که بتواند این قابلیت را برای نرم‌افزارهای تحت پوشش Office فراهم

نماید. حال اگر هدف برنامه‌نویس، ایجاد یک راهکار امنیتی یا یک قابلیت خاص رمزگذاری فایل برای این نرم‌افزارها باشد، می‌توان این قابلیت را در قالب یک کد افزونه به نرم‌افزار اضافه کرد.

برای دریافت افزونه‌های مایکروسافت آفیس، می‌توان به صفحه‌ی Insert وارد شد و سپس در زیرشاخه Add ins، گزینه‌ی My Apps را انتخاب کرد. حال از بخش Store می‌توان افزونه‌های زیادی را بر روی Office نصب کرد. به عنوان مثال می‌توان به افزونه‌ی QRCode اشاره نمود که توسط مایکروسافت توسعه یافته است. همانطور که در تصویر قابل مشاهده است، این افزونه یک URL را می‌تواند به عنوان ورودی دریافت کند و یک QRCode به عنوان خروجی تحویل دهد. در نتیجه آن را می‌توان به داخل محتوا اضافه کرد. (شکل ۱)



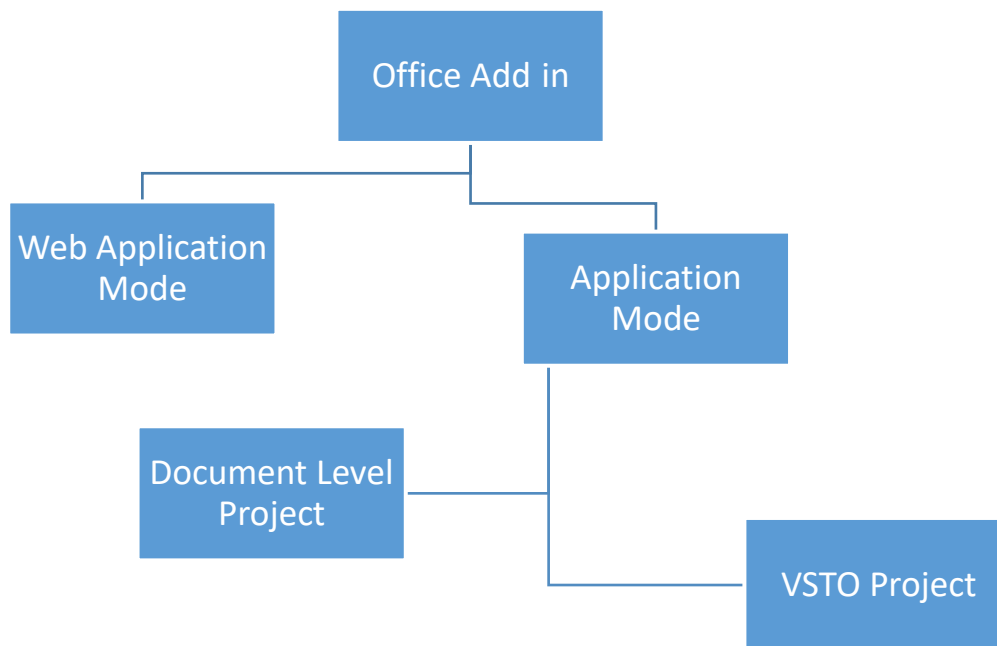
شکل ۱

۴-۲- انواع پروژه‌های Office Add in

برنامه نویسی افزونه برای اکثر برنامه‌های تحت پشتیبانی Office قابل استفاده می‌باشد. برای توسعه‌ی و پیاده‌سازی این افزونه‌ها، مایکروسافت دو راه کلی را پیشنهاد می‌کند:

راه اول توسعه‌ی افزونه در حالت **Web Application Mode** است. در این حالت، افزونه همانند یک صفحه وب بر روی فایل ظاهر می‌شود. بخش **Backend** یا منطقی افزونه با استفاده از کدهای جاوااسکریپت پیاده‌سازی می‌شود و بخش ظاهری یا **Interface** آن با استفاده از **HTML**، **CSS** و ... قابل طراحی است.

اما راه دوم شاید برای افرادی که با زبان **#C** یا **Basic Visual** آشنا هستند راه مناسب‌تر و سریع‌تری به نظر برسد. در این روش، افزونه در حالت **Application Mode** توسعه می‌یابد که بحث اصلی این گزارش است. در این حالت، افزونه بدون نیاز به صفحات وب و با استفاده از ماژول‌های آماده‌ی **.Net Framework** توسعه می‌یابد. مجدداً حالت **Application Mode** را می‌توان به دو زیرحالت **VSTO Project** و **Document Level** **Project** تقسیم نمود. برای درک حالت‌های ایجاد یک پروژه‌ی افزونه، شکل ۲ زیر قابل درک است.



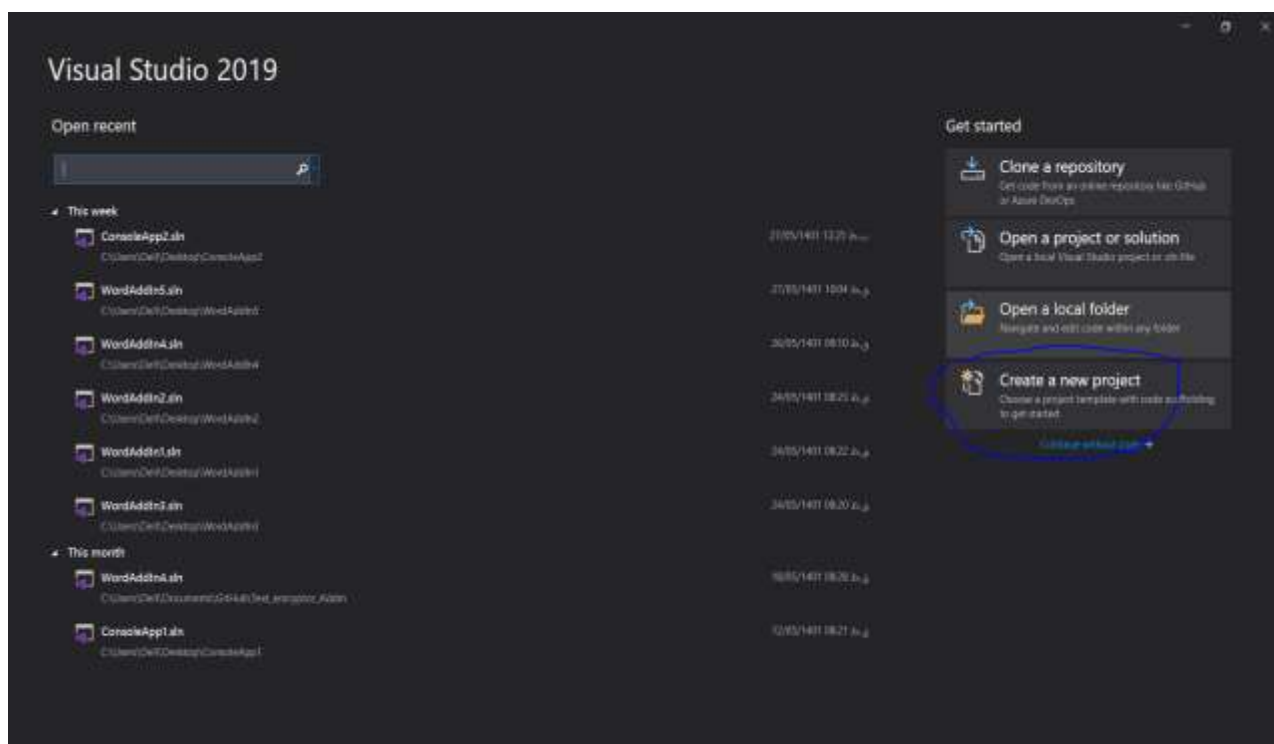
شکل ۲

افزونه‌ای که بر مبنی **VSTO** یا **(Visual Studio Tools for Office)** پیاده‌سازی می‌شود، بر روی تمام فایل‌ها اجرا می‌شود. اما افزونه‌ای که مبتنی بر **Document level** است، بر خلاف حالت قبل، تنها بر روی یک فایل خاص اجرا می‌شود و برای سایر فایل‌ها قابل استفاده نمی‌باشد.

نکته‌ی قابل توجه این است که افزونه‌های مبتنی بر Document level، تنها برای فایل های مبتنی بر Word و Excel پیاده‌سازی می‌شوند. در صورتی که افزونه‌های مبتنی بر VSTO می‌توانند برای تمامی محصولات Office طراحی و استفاده شود.

محیطی که مایکروسافت برای توسعه و کدنویسی افزونه‌ها پیشنهاد می‌کند، محیط Visual Studio می‌باشد. مجدداً تاکید می‌شود که بهتر است نسخه‌ی ۲۰۱۹ به بالای این IDE نصب شود. لازم به ذکر است که این محیط از آن جایی که تحت حمایت مایکروسافت است، می‌تواند از تمامی حالت‌های پروژه‌ی افزونه که در شکل بالا ذکر شد، پشتیبانی نماید.

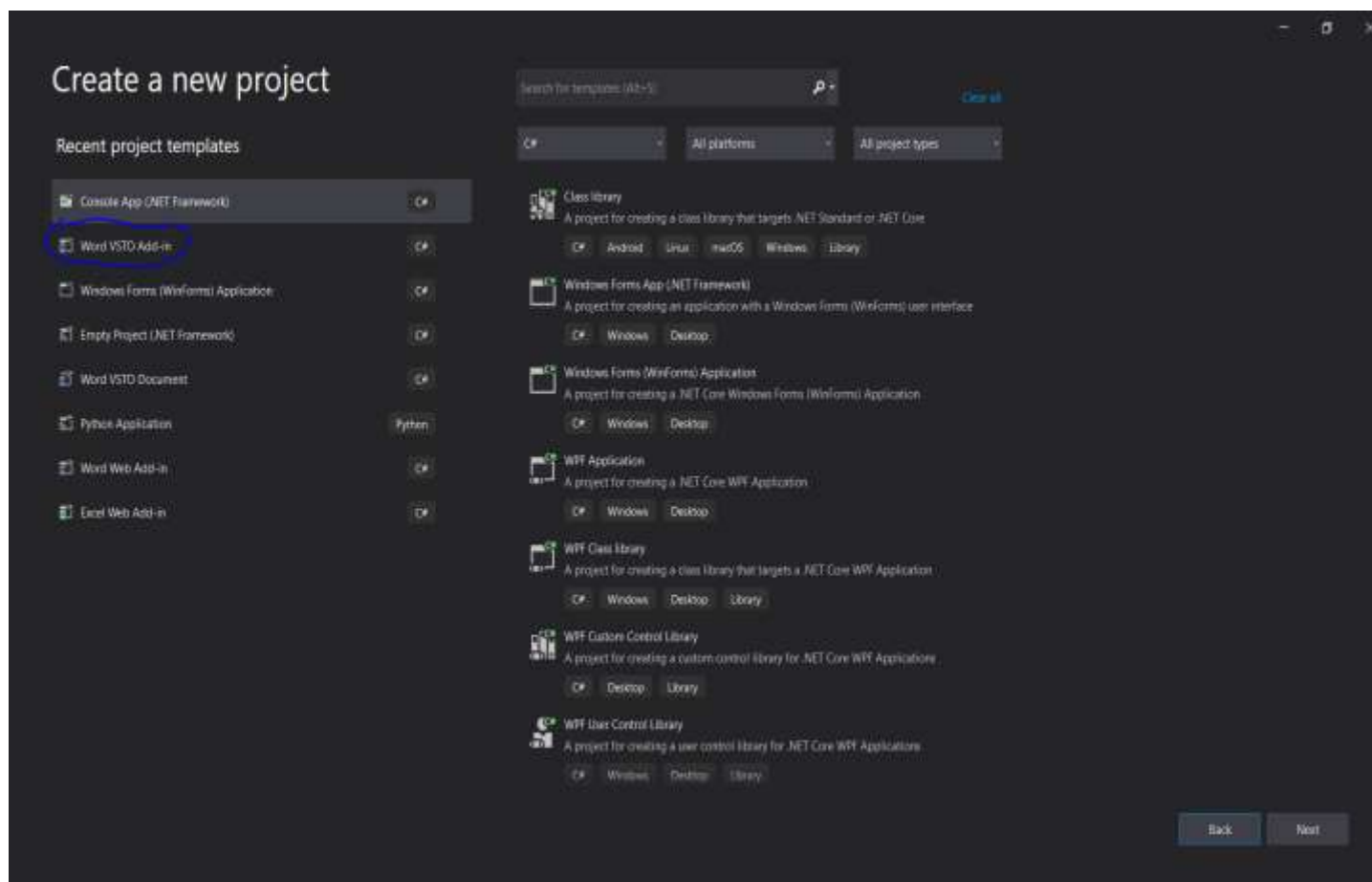
- برای شروع ابتدا وارد محیط Visual Studio شوید و سپس گزینه‌ی Create new project را انتخاب کنید. (شکل ۳)



شکل ۳

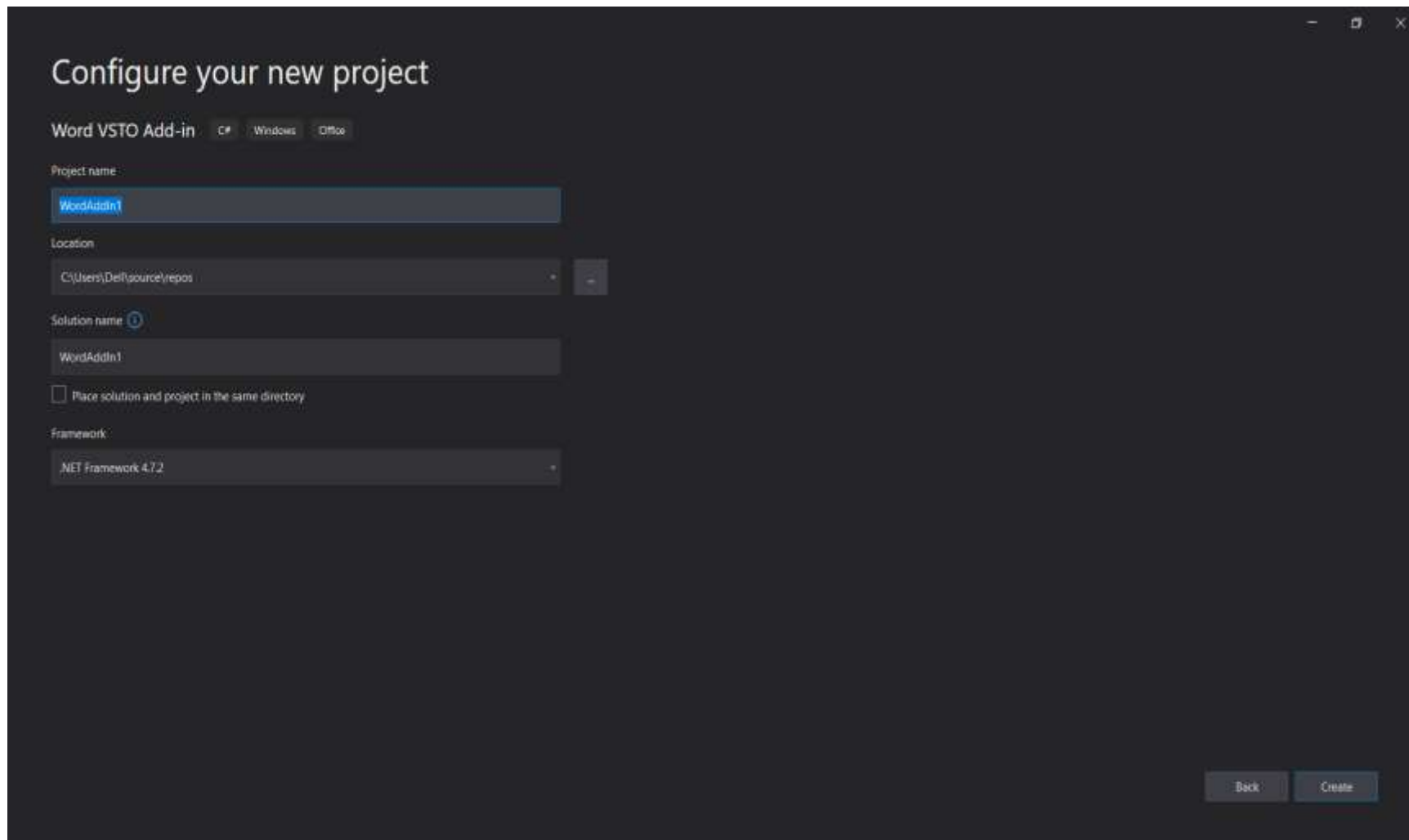
- در ادامه نوع پروژه را انتخاب کنید. قبل از ایجاد پروژه، از نصب بسته‌ی Office/SharePoint و زبان #C بر روی IDE خود، اطمینان حاصل نمایید.

- سپس بر روی گزینه‌ی Word VSTO Add in کلیک کنید. می‌توانید این قالب پروژه را در کادر جست‌وجو پیدا کنید. (شکل ۴)



شکل ۴

- در ادامه مشخصات پروژه‌ی افزونه‌ی word را تعیین کرده و بر روی گزینه‌ی create کلیک کنید. (شکل ۵)



Configure your new project

Word VSTO Add-in C# Windows Office

Project name:

WordAddIn1

Location:

C:\Users\Deil\source\repos

Solution name ⓘ

WordAddIn1

☐ Place solution and project in the same directory

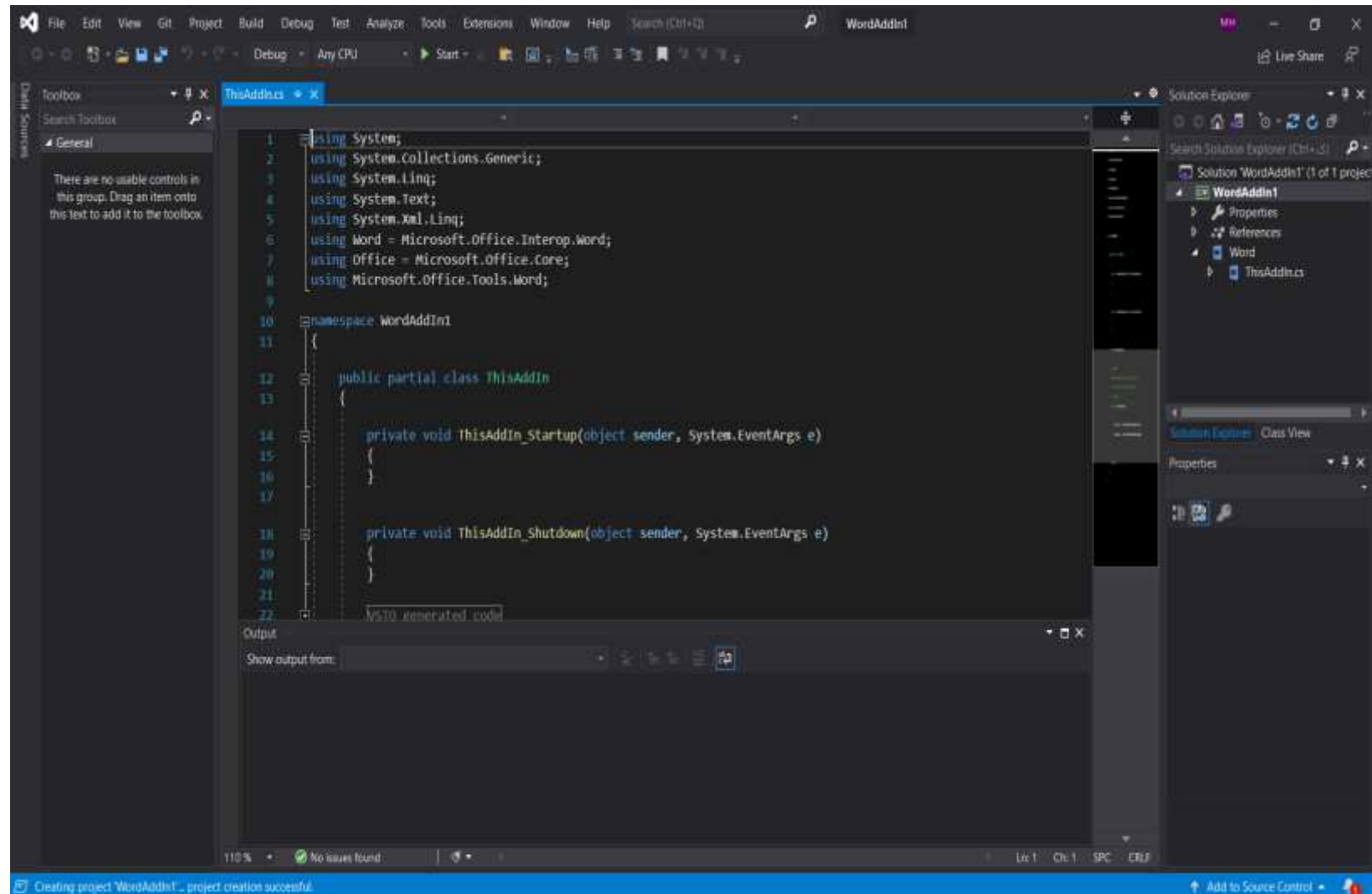
Framework:

.NET Framework 4.7.2

Back Create

شکل ۵

- اکنون صفحه اصلی پروژه نمایش داده می‌شود. (شکل ۶)



شکل ۶

هسته‌ی اصلی پروژه افزونه را کلاس ThisAddIn تشکیل می‌دهد. این کلاس دو Event به نام Startup و Shutdown دارد. موقعیت Startup، لحظه‌ی باز شدن فایل Word (همزمان با فعال شدن افزونه) است و موقعیت Shutdown در لحظه‌ی بسته شدن Add in اتفاق می‌افتد. برای این دو Event، می‌توان در دو تابع ThisAddIn_Startup و ThisAddIn_Shutdown کدنویسی را شروع کرد.

کدی که برای شروع نوشته می‌شود، برنامه‌ایست که قبل از بسته شدن یک فایل Word، یک جمله را به ابتدای فایل اضافه کند. برای این کار، ابتدا یک تابع در کلاس ThisAddIn تعریف می‌شود و نام آن را Add_Text گذاشته می‌شود.

در ادامه داخل تابع Add_text کد زیر را اضافه می‌شود که این کار را انجام می‌دهد. (شکل ۷)

```
15 void Add_Text(Word.Document Doc, ref bool Cancel)
16 {
17     this.Application.ActiveDocument.Paragraphs[1].Range.InsertParagraphBefore();
18     this.Application.ActiveDocument.Paragraphs[1].Range.Text = "This text was added by using code from V
19     this.Application.ActiveDocument.Save();
20 }
21
```

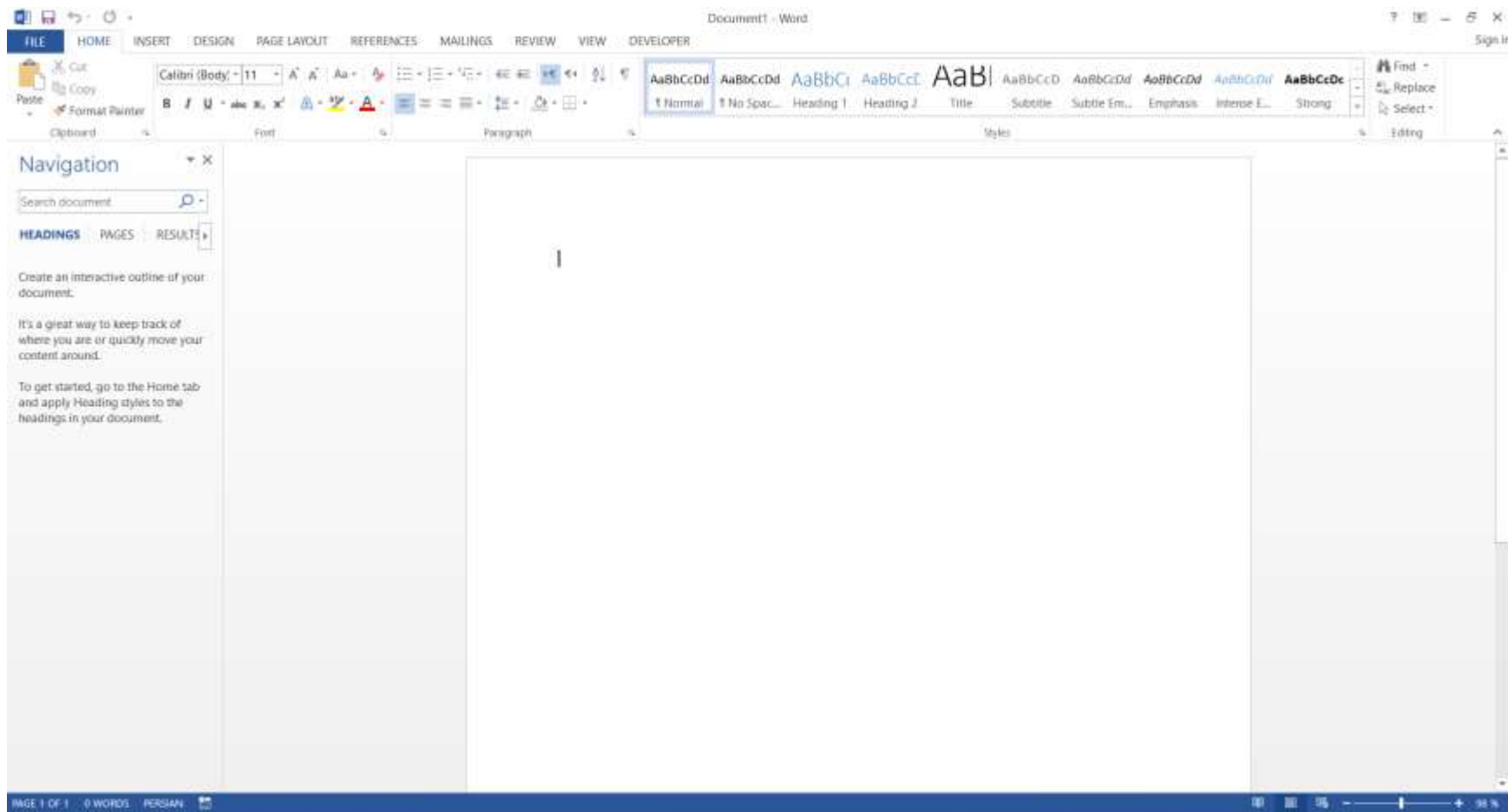
شکل ۷

حال باید زمانی که افزونه شروع به کار می‌کند، Event مربوط به بسته شدن فایل را فعال کرد. به صورتی که وقتی فایل بسته می‌شود، پیش از آن تابع Add_Text صدا زده شود. پس قطعه کد زیر داخل تابع startup نوشته می‌شود. (شکل ۸)

```
25 private void ThisAddIn_Startup(object sender, System.EventArgs e)
26 {
27     this.Application.DocumentBeforeClose +=
28     new Word.ApplicationEvents4_DocumentBeforeCloseEventHandler(Add_Text);
29 }
30
```

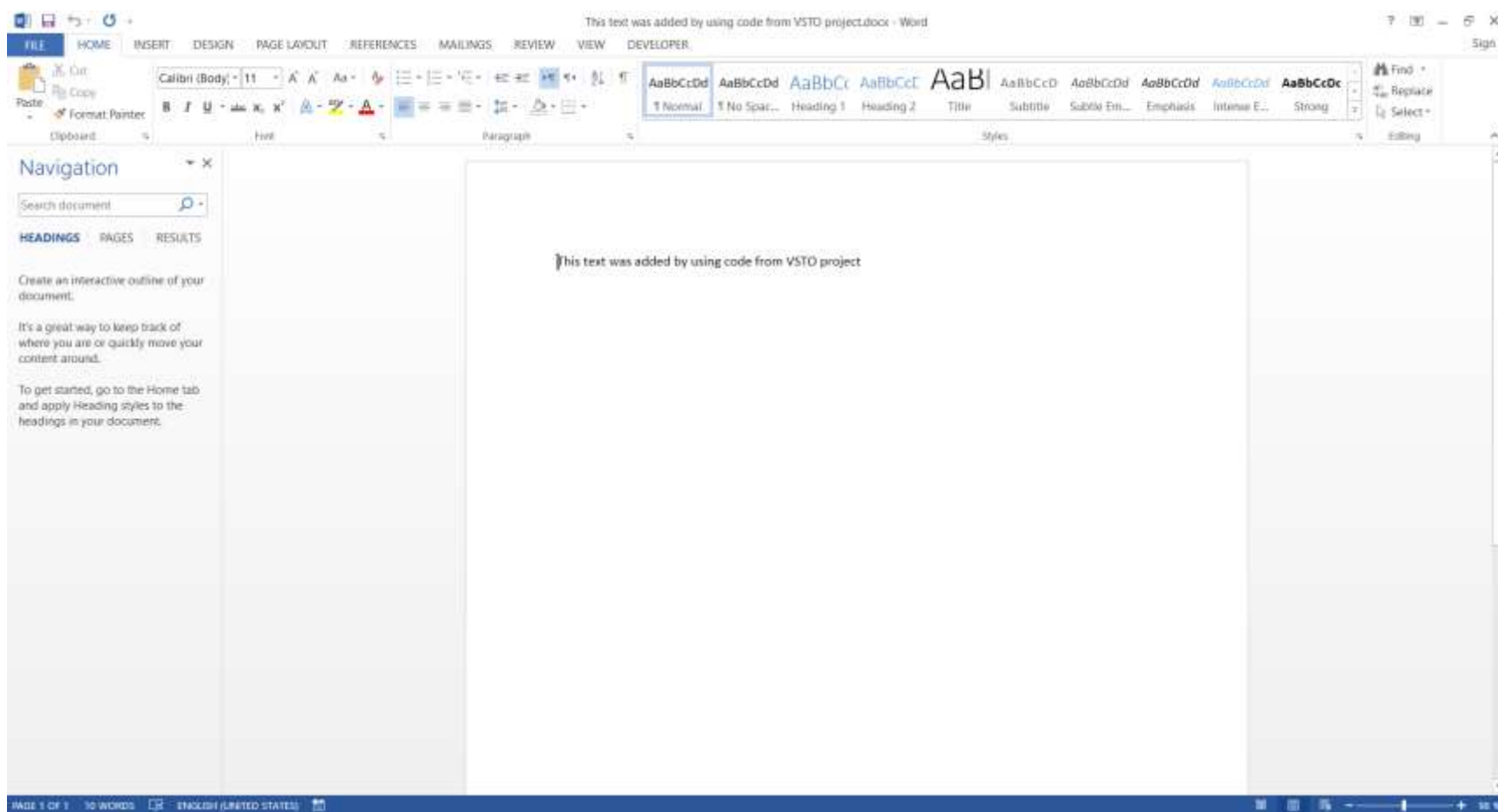
شکل ۸

حال نوبت به آزمایش افزونه می‌رسد. برای این کار از کلید ترکیبی **Ctrl+F5** استفاده می‌شود و همانگونه که در تصویر زیر قابل مشاهده است، یک صفحه خالی **Word** باز می‌گردد. این فایل **Word** را ببندید و مسیر و نامی برای آن در نظر بگیرید. (شکل ۹)



شکل ۹

اگر فایل را باز شود، طبق انتظار همان‌گونه که مشاهده می‌شود، جمله‌ای به فایل افزوده شده است. در واقع افزونه این جمله را پیش از بسته شدن فایل به آن افزوده است. (شکل ۱۰)



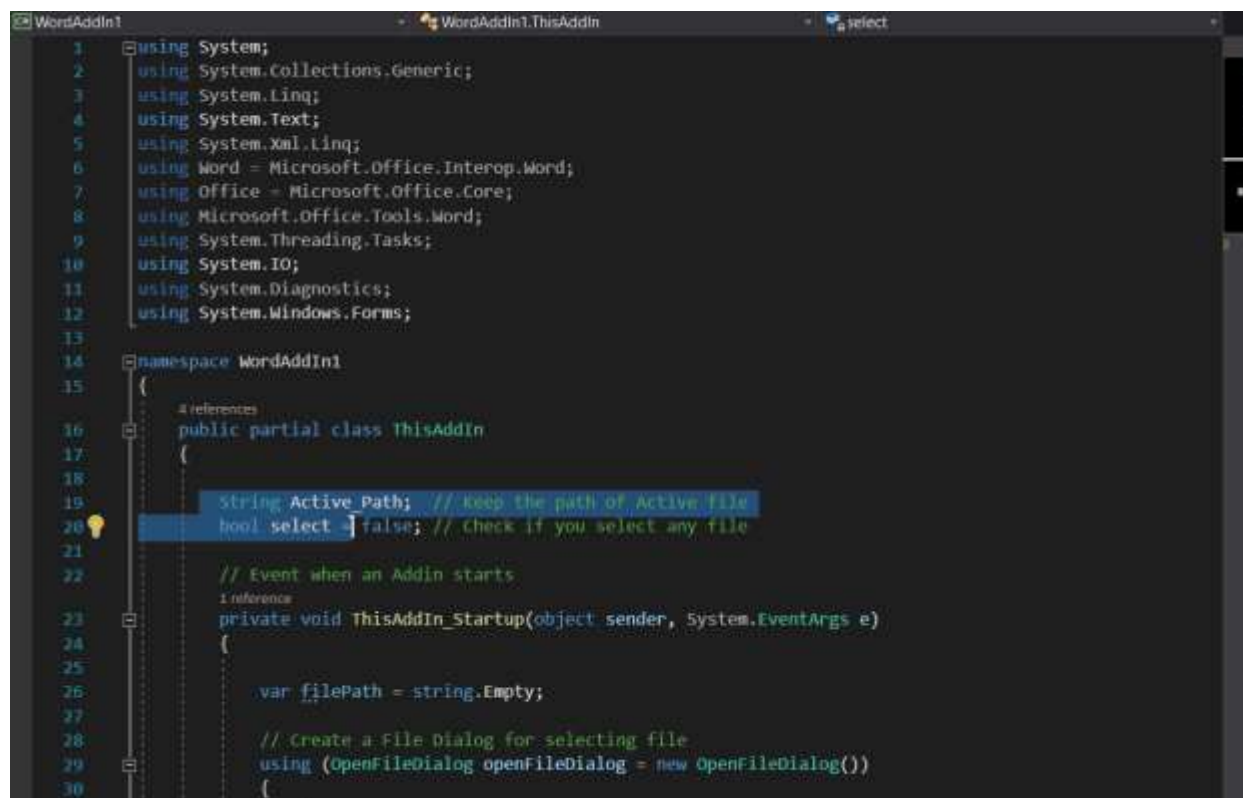
شکل ۱۰

۴-۳- طراحی افزونه رمزنگار با دستکاری بایتی

در مرحله بعد، افزونه‌ی زیر طراحی می‌شود:

افزونه ای که در شروع به کار، یک فایل Word را باز کند و دو عدد ۸۰ و ۷۵ را داخل دو بایت اول آن (در حالت باینری این قسمت از فایل‌های Word به هدر و تنظیمات فایل اختصاص دارد). ذخیره کند. در این صورت فایل Word بدون هیچ مشکلی اجرا خواهد شد. بعد از اتمام کار با فایل و ذخیره آن، فایل بعد از بسته شدن، عدد ۱۳۳ را در دو بایت اول فایل ذخیره کند. در این صورت تنظیمات فایل Word به نوعی از کار می‌افتد و محتویات فایل قابل مشاهده نمی‌باشد. زیرا هدر فایل به صورت نادرست دستکاری شده است.

ابتدا یک متغیر `Active_Path` را برای ذخیره سازی مسیر فایل جاری در نظر گرفته می‌شود. همچنین از متغیر `select` به منظور اطمینان از انتخاب فایل استفاده شده است. ابتدا تابع `ThisAddin_Startup` نوشته می‌شود. در این تابع ابتدا یک `FileDialog` باز می‌گردد تا فایلی جهت مشاهده محتوا انتخاب شود. (شکل ۱۱)



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Xml.Linq;
6 using Word = Microsoft.Office.Interop.Word;
7 using Office = Microsoft.Office.Core;
8 using Microsoft.Office.Tools.Word;
9 using System.Threading.Tasks;
10 using System.IO;
11 using System.Diagnostics;
12 using System.Windows.Forms;
13
14 namespace WordAddin1
15 {
16     public partial class ThisAddin
17     {
18         // references
19         String Active Path; // keep the path of Active file
20         bool select = false; // check if you select any file
21
22         // Event when an Addin starts
23         private void ThisAddin_Startup(object sender, System.EventArgs e)
24         {
25             var filePath = string.Empty;
26
27             // Create a File Dialog for selecting file
28             using (OpenFileDialog openFileDialog = new OpenFileDialog())
29             {
30
```

شکل ۱۱

سپس برای `FileDialog` تنظیمات اولیه‌ای به دلخواه لحاظ می‌شود. در ادامه طبق شکل ۱۲، پس از اینکه فایل انتخاب شد، گزینه‌ی `ok` انتخاب می‌شود.

برای دستکاری فایل جاری، ابتدا فایل را در یک متغیر از نوع `stream` ذخیره می‌گردد. در ادامه محتویات این فایل توسط یک `BinaryReader` داخل یک آرایه‌ی بایتی ذخیره می‌شود.

در ادامه مقادیر بایت اول و دوم فایل باینری به حالت پیش‌فرض `Word` یعنی ۸۰ و ۷۵ منسوب می‌شود.

پس از اعمال تغییرات، آرایه از طریق یک `BinaryWriter` به فایل جاری بازگردانده می‌گردد. حال هدر فایل جاری در تنظیمات صحیح قرار گرفته است.

```
do
{
    openFileDialog.InitialDirectory = "c:\\\\";
    openFileDialog.Filter = "docx files (*.docx)|*.docx";
    openFileDialog.RestoreDirectory = true;
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        filePath = openFileDialog.FileName;
        select = true; // Make sure you have selected a file
        using (var stream = File.Open(filePath, FileMode.Open)) // Open a stream file
        {
            using (var reader = new BinaryReader(stream, Encoding.UTF8, false))
            {
                // Create a BinaryReader to assign binaryfile into a byte array
                byte[] editArray = reader.ReadBytes(Convert.ToInt32(stream.Length));
                if (stream.Length != 0)
                {
                    // Change setting for active opening the file
                    editArray[0] = 80;
                    editArray[1] = 75;

                    // Create a BinaryWriter to assign changes
                    using (var writer = new BinaryWriter(stream, Encoding.UTF8, false))
                    {
                        writer.BaseStream.Position = 0;
                        writer.Write(editArray);
                        stream.Close();
                    }
                }
            }
        }
    }
} // Open the file & Save path
```

شکل ۱۲

در صورتی کاربر هیچ فایلی را انتخاب نکرده باشد و گزینه‌ی `cancel` را بزند، عملیات بالا در یک حلقه‌ی `While` مجدداً تکرار می‌شود.

شکل ۱۳ نمایی از تابع ThisAddIn_Shutdown می‌باشد. هنگامی که فایل Word بسته می‌شود، نسخه‌ی فایل stream آن ساخته می‌شود. سپس از طریق یک Binary Reader، آن stream در حالت بایت به یک آرایه منتقل می‌شود. در ادامه با انتساب مقدار ۱۳۳ (این مقدار دلخواه می‌باشد) به دو بایت اول، هدر Word را به گونه‌ای تغییر می‌کند که فایل Word باز نشود. این تنظیمات از طریق یک BinaryWriter در فایل جاری ثبت می‌شود.

```
64 }
65 // Open the file & Save path
66 this.Application.Documents.Open(filePath);
67 Active_Path = filePath;
68 }
69 } while (select==false); // You have to select a file
70 }
71 }
72 }
73
74 // Event when an Addin shutdown
75 1 reference
76 private void ThisAddIn_Shutdown(object sender, System.EventArgs e)
77 {
78     //Open a stream file
79     using (var stream = File.Open(Active_Path, FileMode.Open))
80     {
81         // Create a BinaryReader to assign binaryfile into a byte array
82         using (var reader = new BinaryReader(stream, Encoding.UTF8, false))
83         {
84             // Change setting for Inactive opening the file
85             byte[] editArray = reader.ReadBytes(Convert.ToInt32(stream.Length));
86
87             editArray[0] = 133;
88             editArray[1] = 133;
89             // Create a BinaryWriter to assign changes
90             using (var writer = new BinaryWriter(stream, Encoding.UTF8, false))
91             {
92                 writer.BaseStream.Position = 0;
93                 writer.Write(editArray);
94             }
95         }
96         stream.Close();
97     }
```

شکل ۱۳

۴-۴- آشنایی با پروتوکل رمزنگاری AES

در این بخش به طور خلاصه توضیحاتی درباره‌ی پروتوکل AES آورده شده است.

این پروتوکل گونه‌ای از پروتکل‌های رمزنگاری است که در دو قالب سخت‌افزاری و نرم‌افزاری قابل پیاده‌سازی می‌باشد. اندازه بلاک ثابت ۱۲۸ بیتی و اندازه کلید ۱۲۸، ۱۹۲ و ۲۵۶ بیتی دارد.

همچنین اندازه‌ی کلیدها مضربی از ۳۲ می‌باشد. برای امنیت بیشتر پروتکل، از یک ماتریس IV یا مقداراولیه استفاده می‌نمایند

به طور کلی الگوریتم از ۴ مرحله تشکیل می‌شود. (شکل ۲۱)



شکل ۱۴

در طول این مراحل، سلسله عملیات‌های ساده‌ی ریاضی انجام می‌شود که موارد زیر هستند. تعداد تکرارهای این عملیات‌های بسته به سطح امنیتی سیستم متغیر می‌باشد:

۱- عملیات sub bytes

۲- عملیات shift rows

۳- عملیات Mix columns

۴- عملیات Add round key

۴-۵- طراحی افزونه‌ی رمزنگار متن

افزونه‌ای که پیشتر برای رمزنگاری یا به نوبه‌ای قفل‌سازی فایل Word نوشته شد، دارای ایرادی بود که مطلوب مسئول کارآموزی قرار نگرفت. در ادامه این موضوع شرح داده می‌شود.

برای رمزنگاری یک فایل ورد، نیاز به مسیر فایل مورد نظر است. این مسیر به دو صورت مستقیم و غیرمستقیم قابل دستیابی است. در روش مستقیم، افزونه خود باید مسیر فایل جاری را تشخیص نماید و اقدام به رمزنگاری آن نماید. در روش غیرمستقیم، این مسیر از طریق یک OpenFileDialog و به کمک کاربر مشخص می‌گردد و در ادامه افزونه فایل را رمز می‌کند.

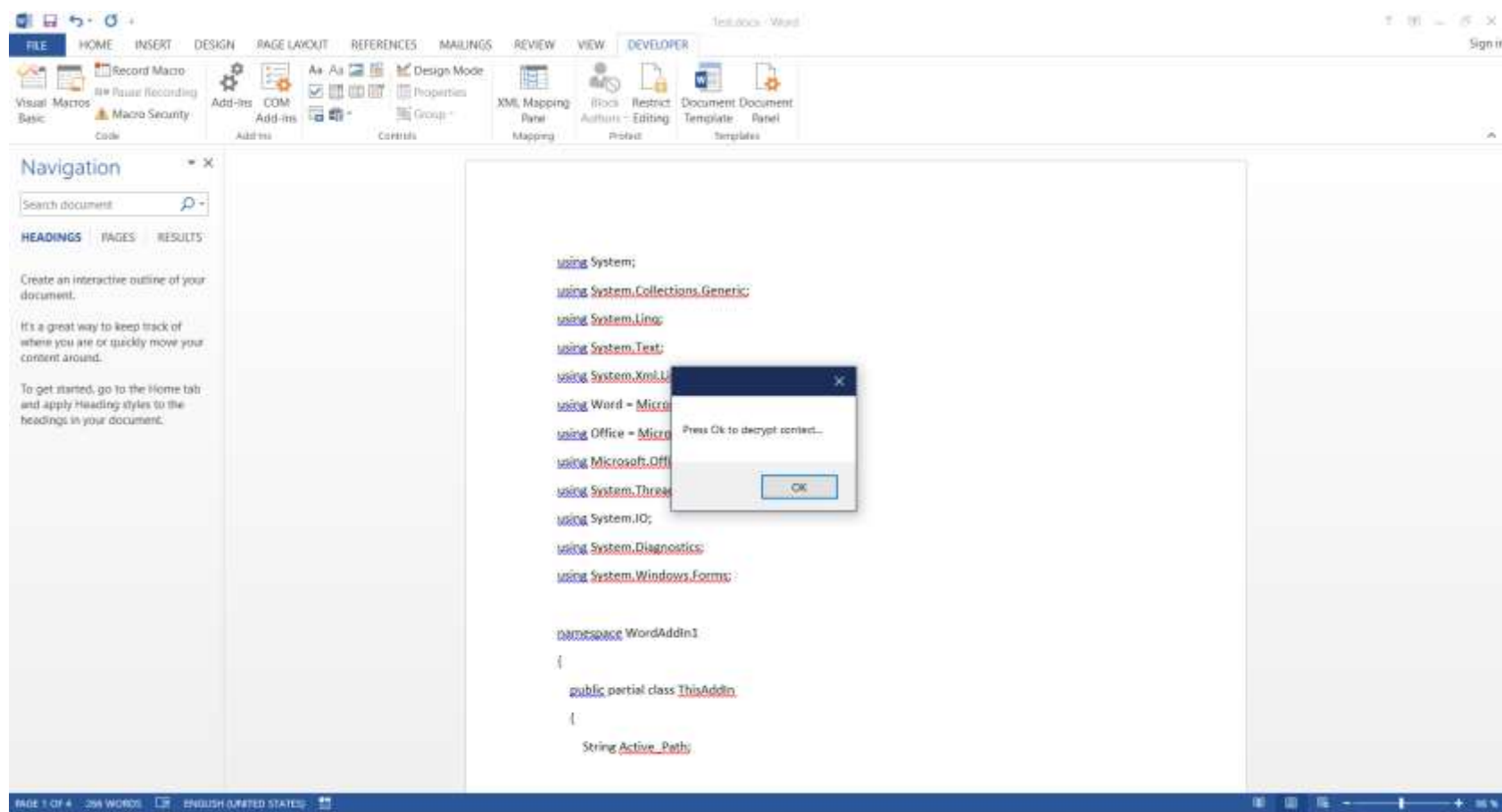
روش مطلوب، روش مستقیم است زیرا دخالت کاربر را به همراه ندارد و فایل جاری توسط افزونه شناسایی می‌شود. اما سوالی که مطرح می‌شود آن است که چرا پیشتر از OpenFileDialog برای مسیریابی فایل استفاده شد؟

همانطور که گفته شد، برای رمزنگاری و رمزگشایی فایل ورد، نیاز به مسیر فایل است. در افزونه‌ای که به روش مستقیم این مسیر را شناسایی می‌کند، زمانی که فایل ورد را باز می‌کند، این مسیر را از یک Property به نام Activefile استخراج می‌کند. اما مشکل اینجاست که این Property زمانی مقدار می‌گیرد که فایل مذکور، معتبر و طبق قالب Docx باشد. لذا چون این فایل پیشتر هدر آن دستکاری شده بود، یک فایل معتبر محسوب نمی‌شد و بنابراین مقدار Activefile برابر Null می‌گشت. لذا به ناچار این مسیر به شیوه‌ی غیرمستقیم از طریق یک OpenFileDialog استخراج می‌شود.

اما همانطور که گفته شد، ایراد این روش در دخالت کاربر در انتخاب مسیر است و ایده‌آل، همان نزدیک شدن به روش مستقیم می‌باشد.

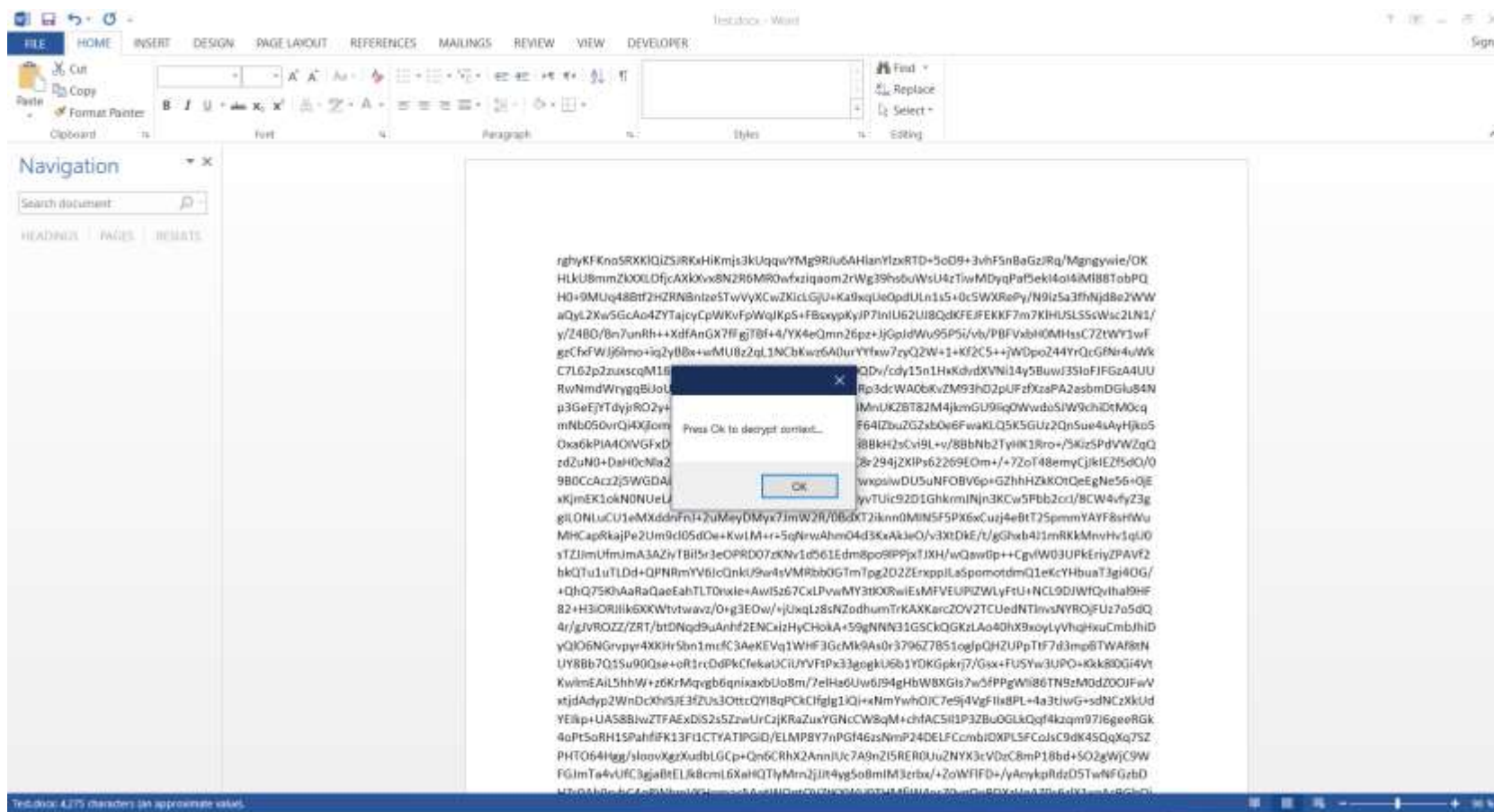
برای این کار طبق مشورت با مسئول کارآموزی، رمزنگاری متن به کمک الگوریتم AES انجام می‌شود. از این طریق دیگر نیازی به OpenFileDialog و مسیریابی غیرمستقیم وجود ندارد. زیرا هدر فایل دستکاری نمی‌شود. برای درک بهتر موضوع ابتدا خروجی کار نمایش داده و در ادامه به شرح کد پرداخته می‌شود.

همانطور که در شکل ۱۵ قابل مشاهده است، فایلی تحت عنوان Test که در آن یک محتوای متنی نوشته شده است باز می‌شود. در ادامه یک Message Box نمایش داده می‌شود که اعلام می‌کند که فایل جاری در حال رمزگشایی است. از آن جایی که این فایل متنی رمزنگاری نشده است، پس تغییری ایجاد نمی‌شود.



شکل ۱۵

در ادامه می‌توانید این فایل ورد را به هر طریقی دست‌کاری کنید. پس از اتمام کار، متن فایل به کمک الگوریتم AES رمزنگاری می‌شود و خروجی در قالب یک رشته‌ی مبتنی بر Base64 در فایل ذخیره می‌شود. حال اگر مجدد فایل Test باز شود، خروجی زیر نمایان می‌شود. در نتیجه اگر سیستمی خارج از سازمان بخواهد از این فایل استفاده کند، چون افزونه‌ی مورد نظر را ندارد با چنین متنی مواجه خواهد شد که غیرقابل فهم است. (شکل ۱۶)



شکل ۱۶

در ادامه مجدد سناریوی قبل تکرار می‌گردد و با کلیک برای روی گزینه‌ی Ok متن رمزگشایی می‌شود.

حال نوبت به توضیح کد می‌رسد:

در شکل ۱۷، تابع رمزنگار مشاهده می‌گردد. این تابع هنگام بستن فایل فراخوانی می‌گردد. در این تابع ابتدا فایل Word جاری در یک متغیر به اسم Doc ذخیره می‌شود. سپس محتوای متنی فایل، داخل رشته‌ی context ذخیره می‌شود. در ادامه از طریق تابع AES_Encryption، رمزنگاری روی متن context صورت می‌گیرد و خروجی به صورت رشته‌ای در قالب Base64 برگرداده می‌شود. این رشته در فایل ثبت و ذخیره می‌گردد.

```
141 1 reference
142 private void Encrypt(Word.Document Doc, ref bool cancel )
143 {
144     String Base64context;
145     Doc = Application.ActiveDocument;
146     String context = Doc.Content.Text;
147     Base64context = AES_Encryption(context, Key, Iv);
148     Doc.Content.Text = Base64context;
149     Doc.Save();
150 }
151
```

شکل ۱۷

در شکل ۱۸، تابع رمزگشا مشاهده می‌شود. این تابع هنگام باز شدن فایل فراخوانی می‌شود. در این تابع، ابتدا یک MessageBox نمایش داده می‌شود که پیام شروع رمزگشایی را به کاربر می‌دهد.

سپس محتوای داکيومنت جاری را در رشته‌ای تحت عنوان Base64context ذخیره می‌کند و بعد از آن، رشته را از حالت Base64، به صورت یک آرایه‌ی بایتی، به تابع AES_Decryption منتقل می‌کند. خروجی این تابع، رشته‌ی قابل خواندنی است که در فایل ذخیره می‌شود.


```

113
114 //Decrypt function for ActiveDocument content text. (from Base64context to readable text using AES)
115 //1- Get Active document
116 //2- Get Base64string from Content.Text
117 //3- Convert Base64String to bytearray
118 //4- Start decryption by AES
119 //5- Save document
1 reference
120 public void Decrypt(Word.Document Doc)
121 {
122     MessageBox.Show("Press Ok to decrypt context...");
123     String context;
124     byte[] context_bytes;
125     String Base64context = Doc.Content.Text;
126     context_bytes = Convert.FromBase64String(Base64context);
127     context = AES_Decryption(context_bytes, Key, Iv);
128     Doc.Content.Text = context;
129     Doc.Save();
130 }
131
132

```

شکل ۱۸

در ادامه به شرح تابع AES_Encryption در شکل ۱۹ پرداخته می‌شود. همان‌طور که در شکل زیر قابل مشاهده است، تابع AES_Encryption وظیفه‌ی اجرای الگوریتم رمزنگاری AES را بر عهده می‌گیرد. این تابع ابتدا رشته‌ی قابل خواندن مورد نظر را دریافت می‌کند و آن را از لحاظ پوچ نبودن بررسی می‌کند.

در ادامه یک شی AES (به نام aesalg) ساخته می‌شود که شرایط را برای پیاده سازی الگوریتم مهیا می‌کند. پس از آن محتوای key و Iv (مخفف Initialize value) برای این شی مقداردهی می‌شود. در ادامه از شی aesalg یک شی encryptor ساخته می‌شود و مقادیر key و Iv به آن منتقل می‌شود.

حال یک شی MemoryStream خالی ایجاد می‌شود و به همراه encryptor، محتوا رمز نگاری می‌شود و به صورت یک آرایه‌ی بایتی در داخل آرایه‌ی Encrypted ذخیره می‌گردد.

حال این آرایه در قالب یک رشته‌ی مبتنی بر Base64، به عنوان خروجی تابع برگردانده می‌شود.

(توضیحات اضافه‌تر به صورت کامنت افزوده شده است)

```
1 reference
39 static String AES_Encryption(string plainText, byte[] Key, byte[] IV)
40 {
41     byte[] encrypted;
42     String Base64String;
43     // Check arguments.
44     Check_plainText(plainText);
45     // Create an Aes object
46     // with the specified key and IV.
47     using (Aes aesAlg = Aes.Create())
48     {
49         aesAlg.Key = Key;
50         aesAlg.IV = IV;
51
52         // Create an encryptor to perform the stream transform.
53         ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);
54         // Create the streams used for encryption.
55         using (MemoryStream msEncrypt = new MemoryStream())
56         {
57             using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))
58             {
59                 using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
60                 {
61                     //Write all data to the stream.
62                     swEncrypt.Write(plainText);
63                 }
64                 encrypted = msEncrypt.ToArray();
65                 Base64String = Convert.ToBase64String(encrypted);
66             }
67         }
68     }
69     // Return the encrypted string in base64 type
70     return Base64String;
71 }
```

شکل ۱۹

همان‌طور که در شکل ۲۰ قابل مشاهده است، تابع `AES_Decryption` وظیفه‌ی اجرای الگوریتم رمزگشایی `AES` را بر عهده می‌گیرد. این تابع ابتدا آرایه‌ی رمز شده را دریافت می‌کند و آن را از لحاظ پوچ نبودن بررسی می‌کند. در ادامه مشابه اتفاقی که در تابع `AES_Encryption` اتفاق می‌افتد، در این تابع رخ می‌دهد. با این تفاوت که از شی `aesAlg` یک شی `encryptor` ساخته می‌شود و مقادیر `key` و `IV` به آن منتقل می‌شود. نهایتاً تابع، یک رشته‌ی خوانا را بازمی‌گرداند. (توضیحات اضافه‌تر به صورت کامنت افزوده شده است)

```

1 reference
76 static string AES_Decryption(byte[] cipherText, byte[] Key, byte[] IV)
77 {
78     // Check arguments.
79     Check_cipherText(cipherText);
80
81     // Declare the string used to hold
82     // the decrypted text.
83     string plaintext = null;
84
85     // Create an Aes object
86     // with the specified key and IV.
87     using (Aes aesAlg = Aes.Create())
88     {
89         aesAlg.Key = Key;
90         aesAlg.IV = IV;
91         // Create a decryptor to perform the stream transform.
92         ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);
93
94         // Create the streams used for decryption.
95         using (MemoryStream msDecrypt = new MemoryStream(cipherText))
96         {
97             using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
98             {
99                 using (StreamReader srDecrypt = new StreamReader(csDecrypt))
100                 {
101                     // Read the decrypted bytes from the decrypting stream
102                     // and place them in a string.
103                     plaintext = srDecrypt.ReadToEnd();
104                 }
105             }
106         }
107     }
108     return plaintext;
109 }

```

شکل ۲۰

در شکل ۲۱ دو آرایه‌ی key و iv قابل مشاهده هستند. همچنین توابع Check_plainText و Check_cipherText به منظور بررسی ورودی‌های رشته استفاده می‌شوند.

```
15 namespace WordAddIn4
16 {
17     4 references
18     public partial class ThisAddIn
19     {
20         // Initialize value for AES protocol
21         byte[] Iv = { 54, 23, 72, 254, 1, 36, 193, 153, 27, 83, 13, 154, 64, 3, 201, 85 };
22         byte[] Key = { 12, 64, 42, 46, 235, 222, 125, 5, 34, 164, 42, 2, 200, 64, 13, 96, 65,
23             153, 176, 2, 63, 7, 24, 199, 59, 14, 106, 34, 132, 55, 222, 70 };
24
25
26
27         1 reference
28         static void Check_plainText(string plainText)
29         {
30             if (plainText == null || plainText.Length <= 0)
31                 throw new ArgumentNullException("plainText");
32
33         1 reference
34         static void Check_cipherText(byte[] cipherText)
35         {
36             if (cipherText == null || cipherText.Length <= 0)
37                 throw new ArgumentNullException("cipherText");
38         }
39     }
40 }
```

شکل ۲۱

۴-۶- امکان بهبود و ارتقای افزونه

پس از ایجاد افزونه‌ی رمزنگار متنی، نتایج تست‌هایی که به دست آمد نشان می‌داد که افزونه امکان حفظ خصوصیتی مانند فونت یا بولد بودن یا نبودن (در برخی موارد خاص) را ندارد. همچنین رمزنگار امکان رمزکردن اشیایی مانند عکس و جداول یا فیلم را نمی‌دهد. همچنین تلاش برای دستیابی به عکس‌ها و فیلم‌ها از طریق افزونه به نتیجه‌ای منجر نشد.

لذا پس از بررسی‌ها، این نتیجه حاصل شد که در هفته‌های باقی‌مانده، مجدداً بر روی رمزنگاری در لایه‌ی سخت افزاری تحقیق شود. با این تفاوت که این‌دفعه فایل‌های XML مرتبط با سند ورد، دستکاری بایستی شوند. این کار به این صورت است که زمانی که کاربر فایل ورد را می‌بندد، تنظیمات غلط بر روی فایل‌های XML مرتبط اعمال شود (در این حالت محتوای بصری حفظ می‌شود). و زمانی که فایل ورد باز می‌شود، این تنظیمات به حالت طبیعی بازگردد. به این ترتیب اگر کاربری افزونه‌ی مورد نظر را نداشته باشد، زمانی که فایل ورد را باز می‌کند، با فایلی مواجه می‌شود که تنظیمات درستی ندارد و نمی‌تواند محتوا را مشاهده کند.

اما مجدداً سناریوی بالا با شکست مواجه شد زیرا مجدداً افزونه نتوانست آدرس فایل را شناسایی کند. با این وجود توسعه و بهبود این افزونه به طوری که بتواند امکان رمزنگاری حداکثری فایل ورد را داشته باشد، می‌تواند موضوع مناسبی برای علاقه‌مندان به این حوزه باشد.

منابع استفاده شده

مقالات و منابع پروتکل AES

1-

<https://fa.wikipedia.org/wiki/%D8%A7%D8%B3%D8%AA%D8%A7%D9%86%D8%AF%D8%A7%D8%B1%D8%AF%D8%B1%D9%85%D8%B2%D9%86%DA%AF%D8%A7%D8%B1%DB%8C%D9%BE%DB%8C%D8%B4%D8%B1%D9%81%D8%AA%D9%87>

2-

<https://www.aparat.com/v/M3PI4/%D8%A7%D9%84%DA%AF%D9%88%D8%B1%DB%8C%D8%AA%D9%85%D8%B1%D9%85%D8%B2%D9%86%DA%AF%D8%A7%D8%B1%DB%8C%AES%D8%A8%D9%87%D8%B2%D8%A8%D8%A7%D9%86%D8%B3%D8%A7%D8%AF%D9%87>

مقالات آشنایی و برنامه‌نویسی افزونه

3-

<https://docs.microsoft.com/en-us/office/dev/add-ins/quickstarts/excel-quickstart-jquery?tabs=yeomangenerator>

4-

<https://docs.microsoft.com/en-us/office/dev/add-ins/quickstarts/word-quickstart?tabs=yeomangenerator>

5-

<https://docs.microsoft.com/en-us/office/dev/add-ins/quickstarts/word-quickstart?tabs=yeomangenerator>

6-

<https://docs.microsoft.com/en-us/office/dev/add-ins/overview/office-add-ins>

7-

<https://docs.microsoft.com/en-us/visualstudio/vsto/word-solutions?view=vs-2022>

8-

<https://docs.microsoft.com/en-us/visualstudio/vsto/walkthrough-creating-your-first-document-level-customization-for-word?view=vs-2022&tabs=csharp>

9-

<https://docs.microsoft.com/en-us/visualstudio/vsto/walkthrough-creating-your-first-vsto-add-in-for-word?view=vs-2022&tabs=csharp>

10-

<https://docs.microsoft.com/en-us/visualstudio/vsto/office-development-samples-and-walkthroughs?view=vs-2022>

11-

<https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.aes?view=net-6.0>

12-

<https://docs.microsoft.com/en-us/visualstudio/vsto/how-to-programmatically-protect-documents-and-parts-of-documents?view=vs-2022&tabs=csharp>

13-

<https://docs.microsoft.com/en-us/office/dev/add-ins/overview/learning-path-beginner>

14-

<https://docs.microsoft.com/en-us/dotnet/api/system.io.binarywriter?view=net-6.0>

15-

<https://docs.microsoft.com/en-us/dotnet/api/system.io.binaryreader?view=net-6.0>

16-

[https://docs.microsoft.com/en-us/dotnet/api/system.io.binaryreader.readbytes?view=net-6.0#system-io-binaryreader-readbytes\(system-int32\)](https://docs.microsoft.com/en-us/dotnet/api/system.io.binaryreader.readbytes?view=net-6.0#system-io-binaryreader-readbytes(system-int32))

17-

<https://www.c-sharpcorner.com/UploadFile/mahesh/openfiledialog-in-C-Sharp>