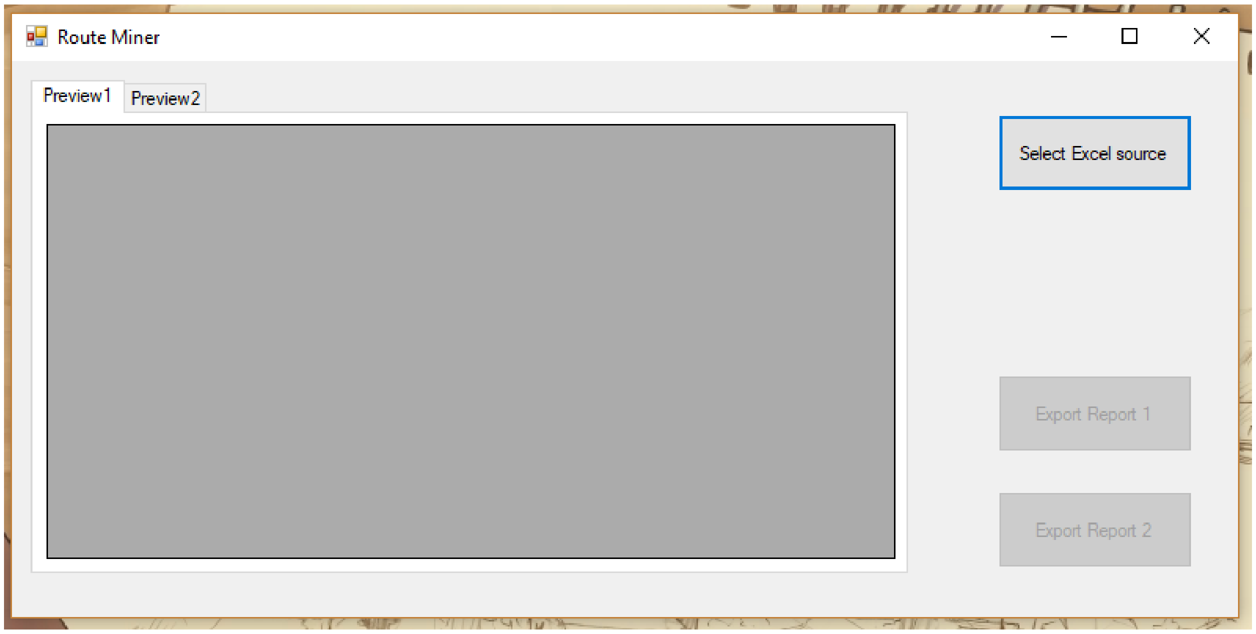
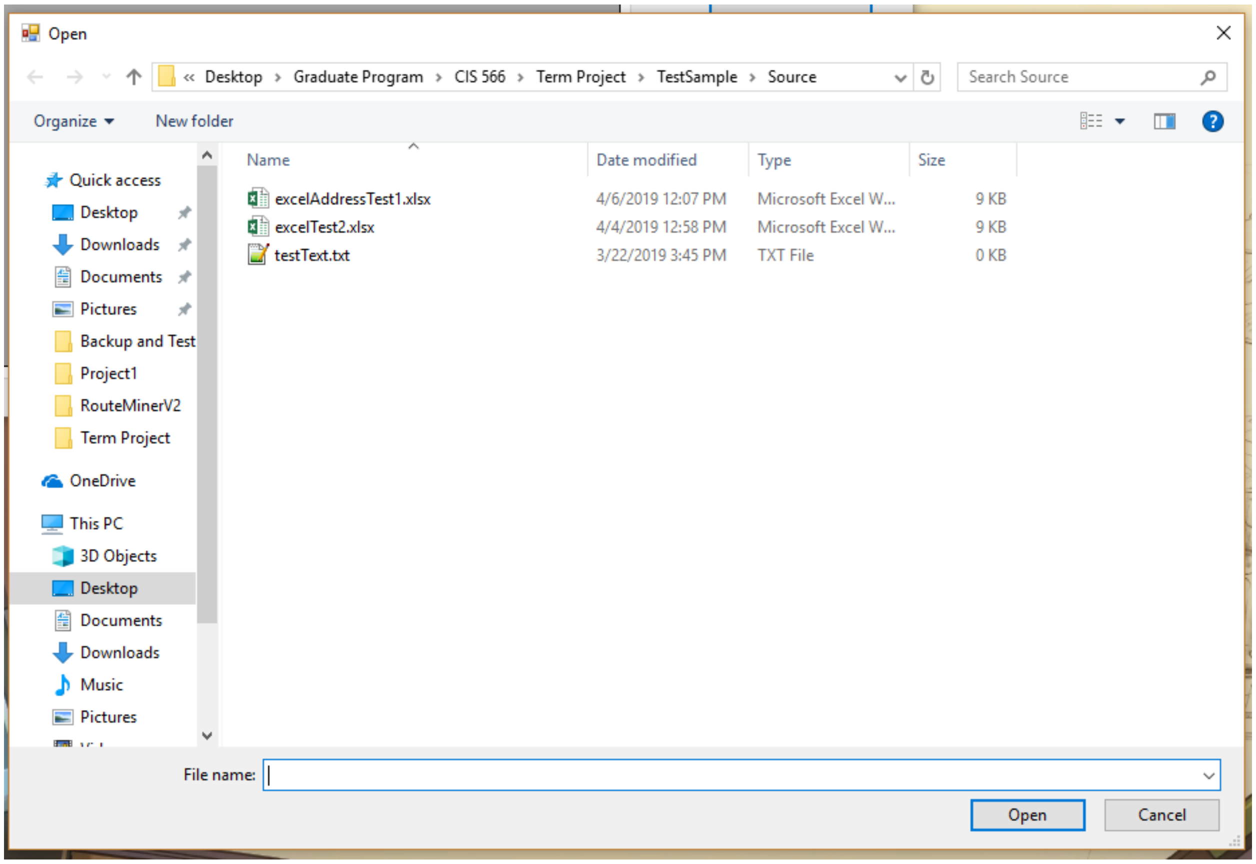
Initial View:



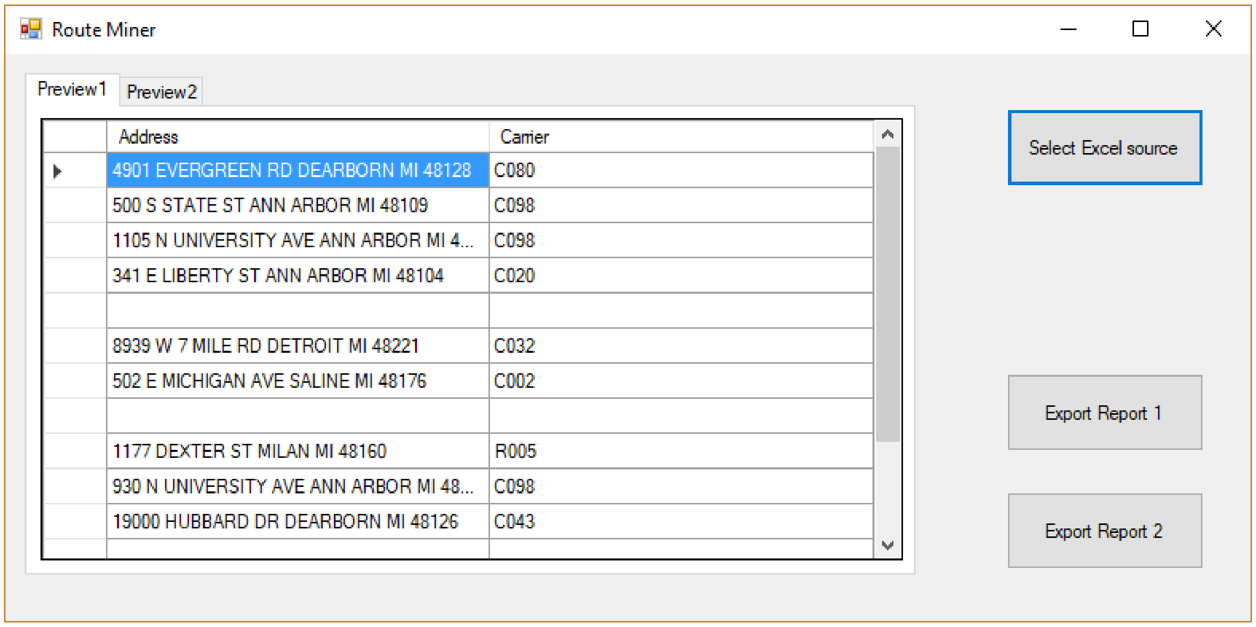
Since there's nothing to display or export, the previews are blanks, while export button are disabled.



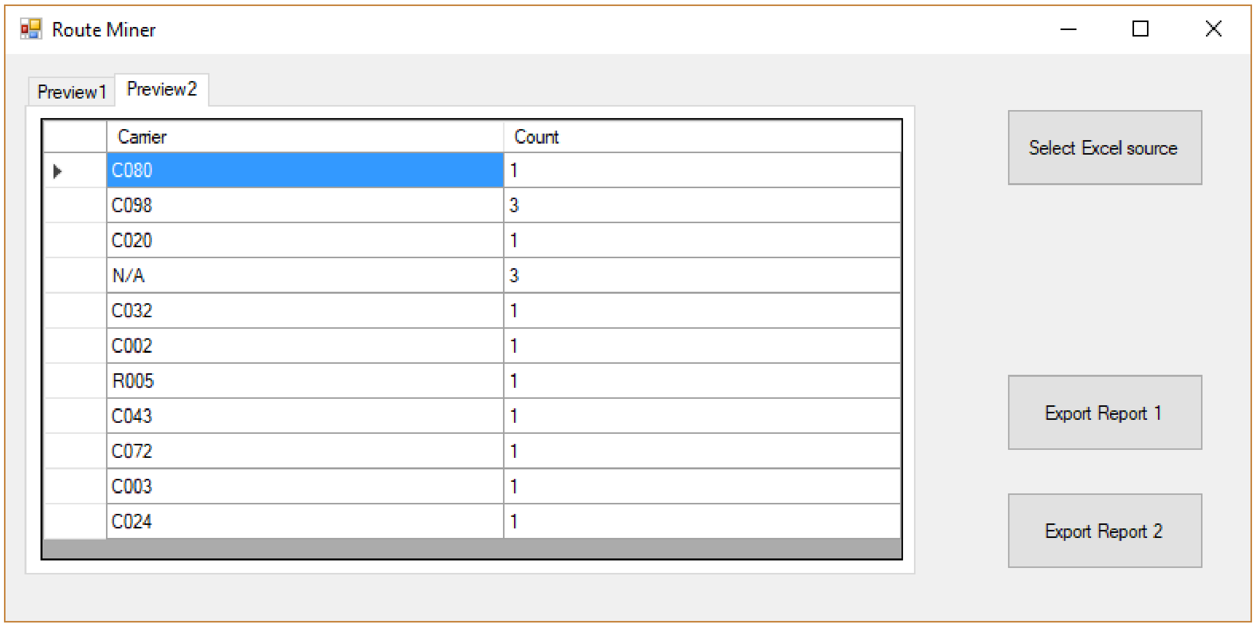
Name	Date modified	Type	Size
excelAddressTest1.xlsx	4/6/2019 12:07 PM	Microsoft Excel W...	9 KB
excelTest2.xlsx	4/4/2019 12:58 PM	Microsoft Excel W...	9 KB
testText.txt	3/22/2019 3:45 PM	TXT File	0 KB

Open file dialog pops up to select the source.

After selecting the Excel source, Route Miner will call the USPS API to validate the addresses and provide a preview of data contained in Report 1 and Report 2



Address	Carrier
4901 EVERGREEN RD DEARBORN MI 48128	C080
500 S STATE ST ANN ARBOR MI 48109	C098
1105 N UNIVERSITY AVE ANN ARBOR MI 4...	C098
341 E LIBERTY ST ANN ARBOR MI 48104	C020
8939 W 7 MILE RD DETROIT MI 48221	C032
502 E MICHIGAN AVE SALINE MI 48176	C002
1177 DEXTER ST MILAN MI 48160	R005
930 N UNIVERSITY AVE ANN ARBOR MI 48...	C098
19000 HUBBARD DR DEARBORN MI 48126	C043

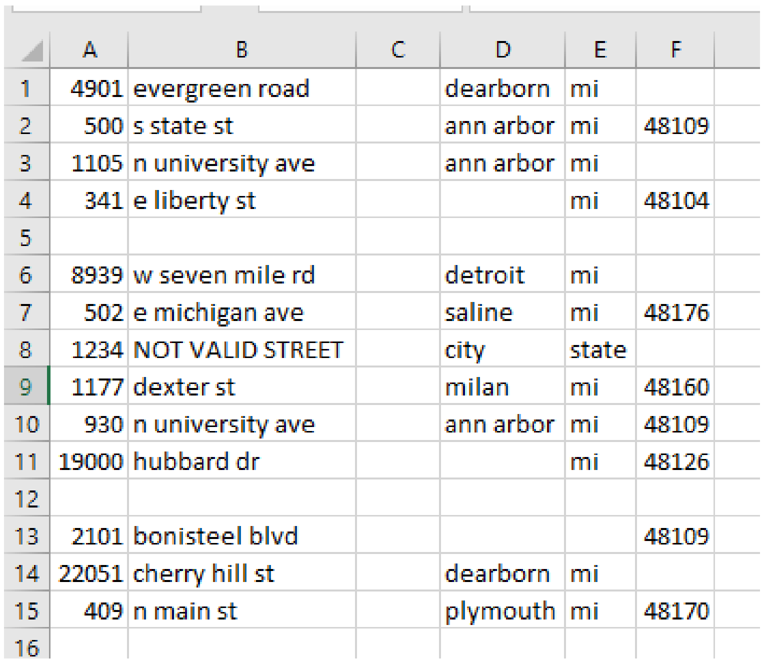


Carrier	Count
C080	1
C098	3
C020	1
N/A	3
C032	1
C002	1
R005	1
C043	1
C072	1
C003	1
C024	1

The preview page displays the content of the reports if the user were to export them to an Excel file. Preview1 shows report 1 data, Preview2 shows report 2 data.

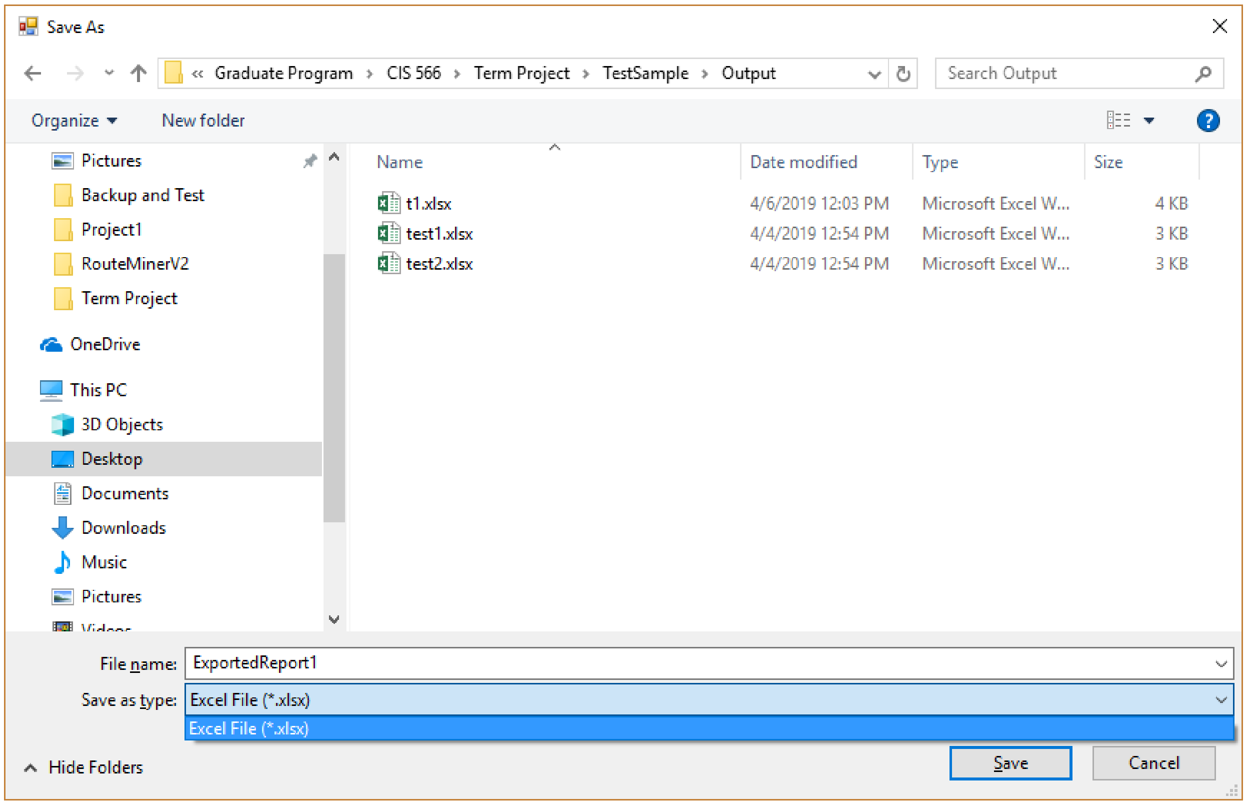
Empty rows in preview 1 indicate that the source data address is not valid/incorrect. This will be expressed in report 1 as an empty row, while showing a carrier route of “N/A” in report 2.

The “Export Report” buttons are now enabled as well.



	A	B	C	D	E	F
1	4901	evergreen road		dearborn	mi	
2	500	s state st		ann arbor	mi	48109
3	1105	n university ave		ann arbor	mi	
4	341	e liberty st			mi	48104
5						
6	8939	w seven mile rd		detroit	mi	
7	502	e michigan ave		saline	mi	48176
8	1234	NOT VALID STREET		city	state	
9	1177	dexter st		milan	mi	48160
10	930	n university ave		ann arbor	mi	48109
11	19000	hubbard dr			mi	48126
12						
13	2101	bonisteel blvd				48109
14	22051	cherry hill st		dearborn	mi	
15	409	n main st		plymouth	mi	48170
16						

For reference, this is the source Excel data, ending on row 15. There are 2 unfilled rows (row 5, 12) and an incorrectly filled row (row 8). Other rows may be missing Zip code or City as well. As long as there are sufficient information, the API can fill in the blanks.



Save As

« Graduate Program › CIS 566 › Term Project › TestSample › Output

Search Output

Organize ▾ New folder

Pictures
Backup and Test
Project1
RouteMinerV2
Term Project
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos

Name	Date modified	Type	Size
t1.xlsx	4/6/2019 12:03 PM	Microsoft Excel W...	4 KB
test1.xlsx	4/4/2019 12:54 PM	Microsoft Excel W...	3 KB
test2.xlsx	4/4/2019 12:54 PM	Microsoft Excel W...	3 KB

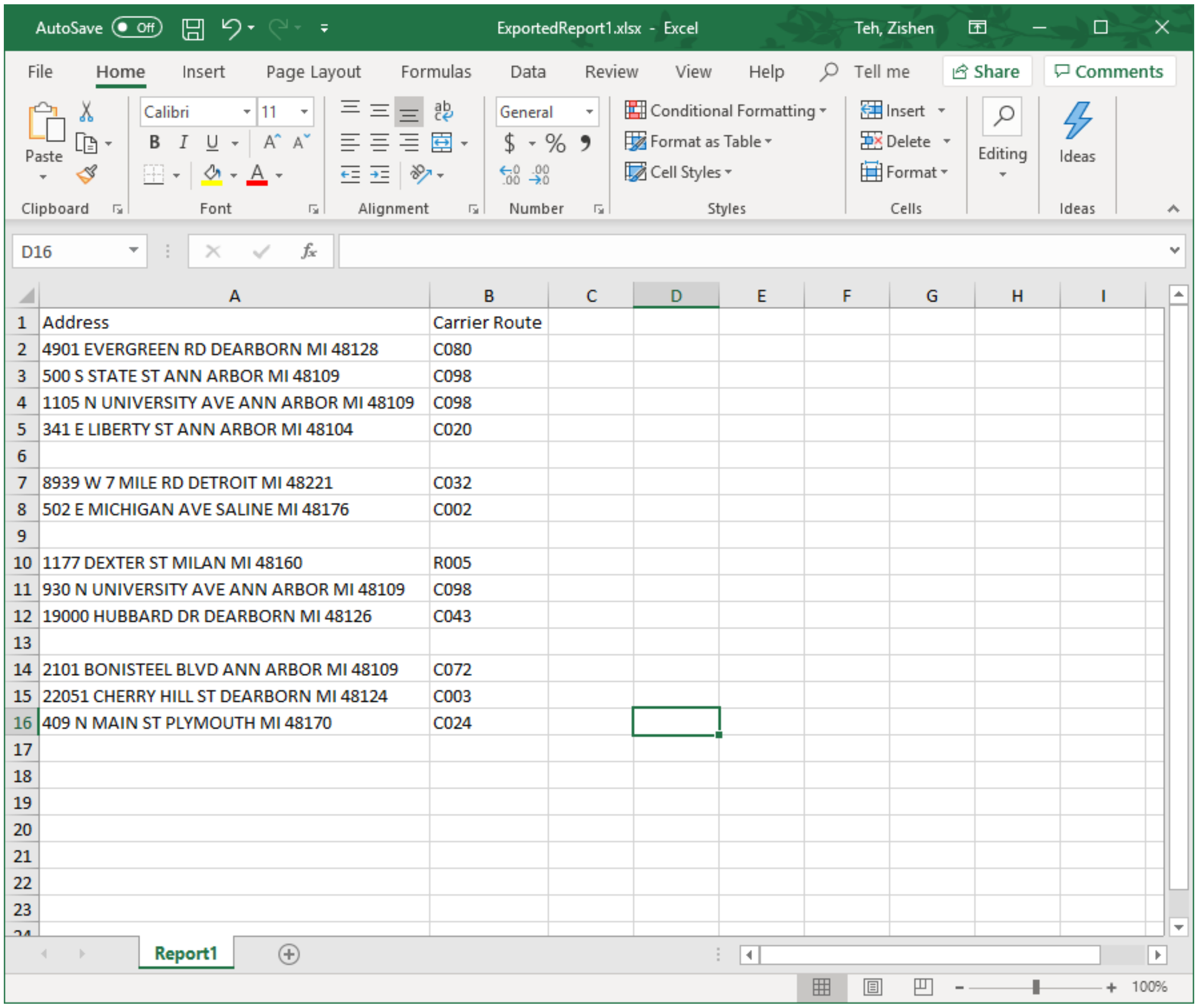
File name: ExportedReport1

Save as type: Excel File (*.xlsx)

Excel File (*.xlsx)

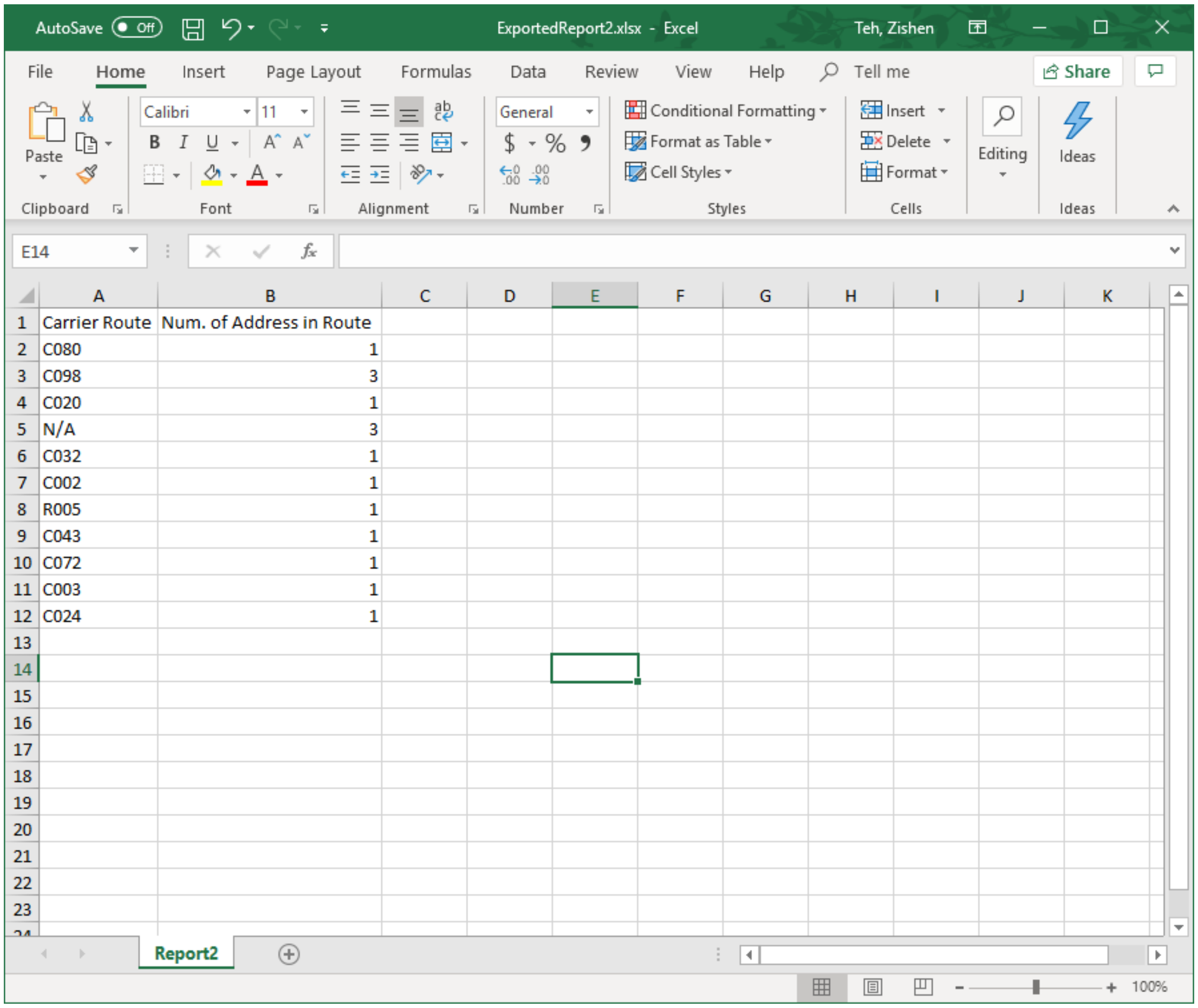
Hide Folders Save Cancel

Selecting the Export buttons will prompt the user for a destination to save the report. The only type allowed in this case is Excel File of extension .xlsx



	A	B
1	Address	Carrier Route
2	4901 EVERGREEN RD DEARBORN MI 48128	C080
3	500 S STATE ST ANN ARBOR MI 48109	C098
4	1105 N UNIVERSITY AVE ANN ARBOR MI 48109	C098
5	341 E LIBERTY ST ANN ARBOR MI 48104	C020
6		
7	8939 W 7 MILE RD DETROIT MI 48221	C032
8	502 E MICHIGAN AVE SALINE MI 48176	C002
9		
10	1177 DEXTER ST MILAN MI 48160	R005
11	930 N UNIVERSITY AVE ANN ARBOR MI 48109	C098
12	19000 HUBBARD DR DEARBORN MI 48126	C043
13		
14	2101 BONISTEEL BLVD ANN ARBOR MI 48109	C072
15	22051 CHERRY HILL ST DEARBORN MI 48124	C003
16	409 N MAIN ST PLYMOUTH MI 48170	C024

The resulting Report 1 that is exported. Here, the Address column contains the formatted address containing street number, street name, city, state, and zip code of the address. Any missing information is filled in such as city, zip, state of the address. The Carrier Route of each address is also displayed next to each valid address.

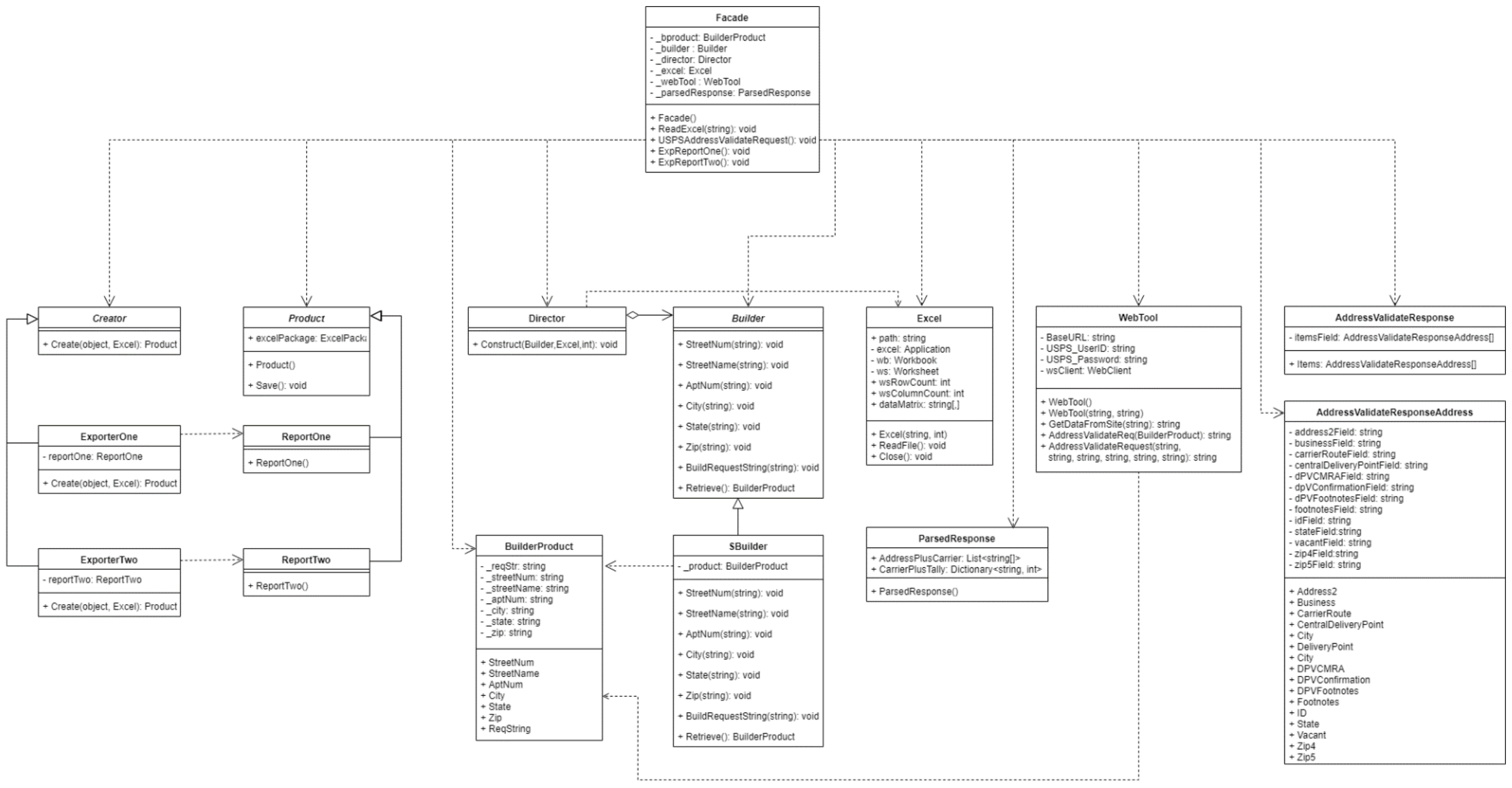


	A	B
1	Carrier Route	Num. of Address in Route
2	C080	1
3	C098	3
4	C020	1
5	N/A	3
6	C032	1
7	C002	1
8	R005	1
9	C043	1
10	C072	1
11	C003	1
12	C024	1

The resulting Report 2 that is exported. The data shown here are of the Carrier Routes that showed up from the address list. The data is ordered by whichever carrier route appeared first, while keeping count of any repeated carrier routes. There are 3 addresses that share the carrier route of "C098" which is indicated by 3 on the column next to it (the 3 addresses in Ann Arbor). As shown, there are also 3 "N/A" carrier route shown, accounting for the 3 invalid addresses from the source list (which are the 2 blanks and 1 incorrectly filled address).

Façade Pattern: The façade shows the specific request that the client has access to. The methods will delegate those requests using specific subsystem objects. The steps of importance are to:

1. Select the Excel source file and read the content
2. Use USPS AddressValidation API to get the proper address to fill in any missing information such as City, State, or Zip just in case
3. From the response, extract each address's Carrier Route, while tallying the amount of repeated Carrier Routes
4. Export either ReportOne or ReportTwo into an Excel file to show the summary of Address and CarrierRoute and how many times the CarrierRoute was repeated



Address Validation Subsystem:

The façade uses the **Director** object which creates an instance of **BuilderProduct** using the ***Builder*** interface, representing 1 XML request to be sent by **WebTool** to the USPS API. The initial response is of type **AddressValidateResponse** which supports up to five addresses in one response, but for this application we will send each XML request one-by-one, and so the response is always accessed from the first element (index 0) of the **AddressValidateResponse[]** array. From the response of type **AddressValidateResponse**, the façade will format and store the response into the **ParsedResponse** object, which contains a List and Dictionary, to prepare for the exporting function later.

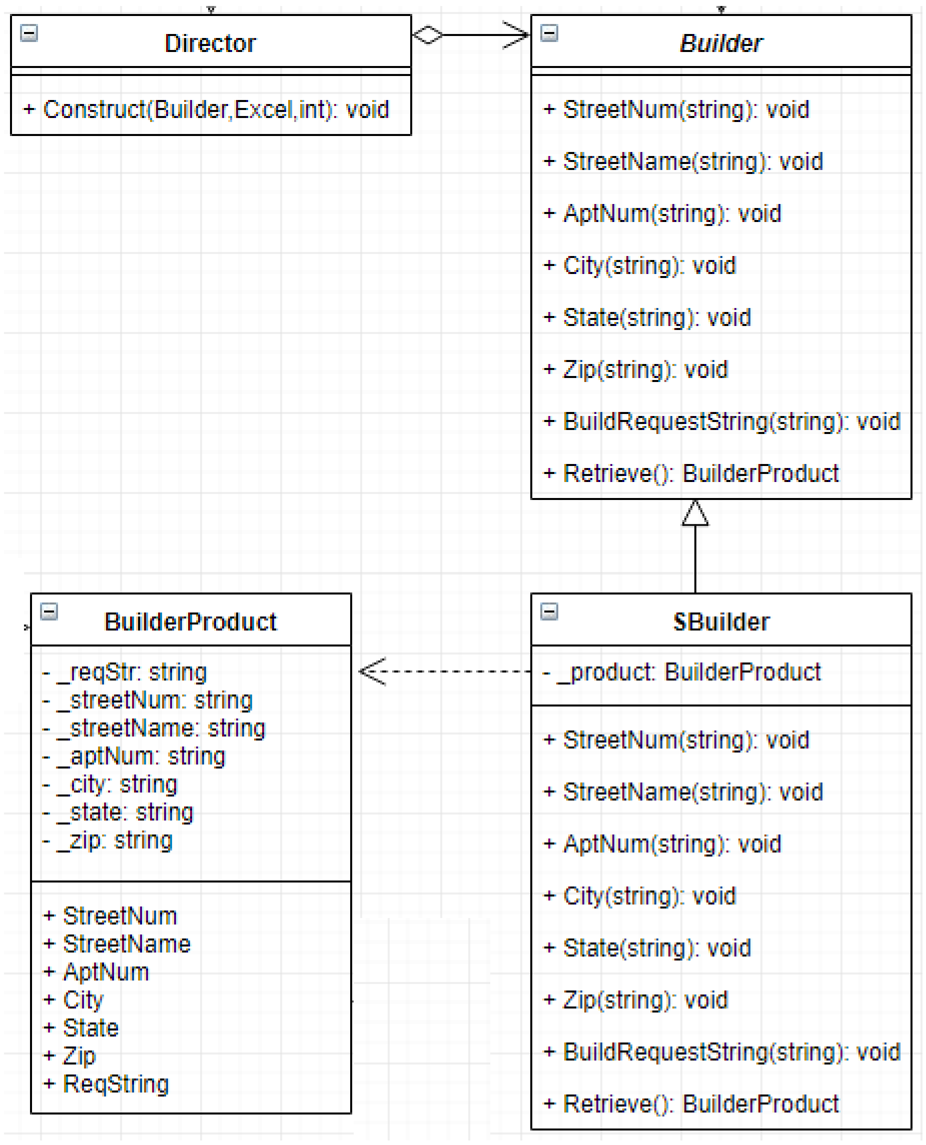
XML Request Builder Subsystem:

The façade delegates responsibility of creating XML request that is properly formatted and ready for use by the **WebTool** class by using the **Director** which will create new instances of **BuilderProduct**. This subsystem uses the builder pattern which is further explained below.

Report Exporter Subsystem:

The façade delegates responsibility of creating reports by utilizing the factory method which reports the specific report the user request. Further details are explained below as well.

Builder Pattern: The Director calls the specific concrete builder (SBuilder) to create different BuilderProduct objects based on the data from the Excel source file. Each BuilderProduct is the specific XML request that is formatted, ready for use by WebTool.



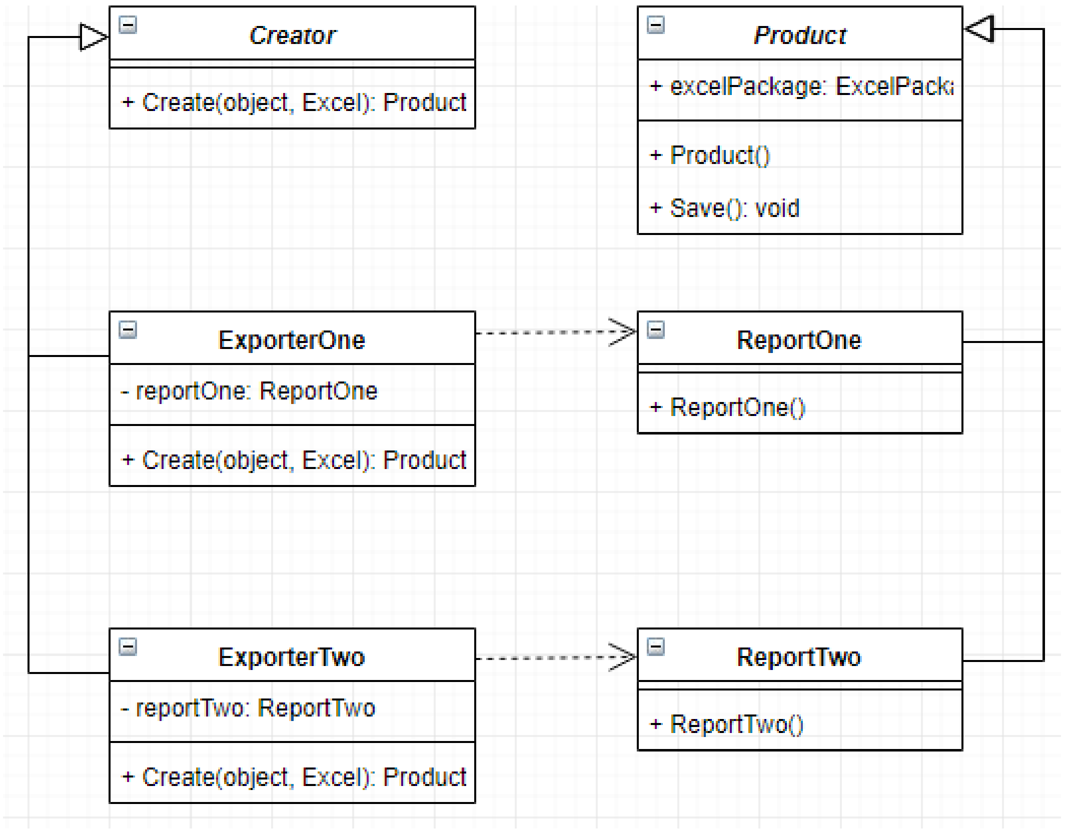
Director: Constructs the BuilderProduct object by using the *Builder* interface

Builder: The interface used to create parts of the BuilderProduct object

SBuilder: The concrete builder class which assembles the BuilderProduct by implementing the *Builder* interface. The Retrieve() returns the specific BuilderProduct that is created. Other methods are used to create specific parts of the product, in this case that would be the XML elements.

BuilderProduct: The object the concrete builder builds. This represents the **one** XML request in its entirety to be sent for AddressValidation. Additional address are encapsulated in their own instances of BuilderProduct.

Factory Pattern: The factory method creates whichever product the user requests when selecting to export either report one or report two. Both products share the same Save() function while each have different contents.



Product: Abstract class for the type of objects factory method creates

ReportOne, ReportTwo: Concrete product class which implement the *Product* abstract class. Both concrete products share the same Save() function

Creator: Abstract class that declares the factory method: Create() which returns object of type Product

ExporterOne, ExporterTwo: Concrete creator classes, which overrides the factory method to return an instance of concrete product (either ReportOne or ReportTwo)

Source Code:

Included in the zip file