UNIVERSITY OF CAPE TOWN

CSC3003S

COMPUTER SCIENCE

Africa Internet Visualizer

Authors:	*Student Numbers:*
Anele Dlamini	DLMSIL008
Khumbulani Ntuli	BYLKHU002
Simamkele Nongwe	NNGSIM004

April 26, 2021

Abstract

On today's world the use of the internet is increasing at a high rate and so the need to be able to model economic forces that drive evolution of the Internet topology and its hierarchy. Africa is one of the continents with the slow internet and that still has a big room for improvement. Visualization of the current existing Internet topology and its hierarchy is the first approach into improving the Internet interconnections and traffic exchange in Africa.

On this project we inferred Autonomous System (AS) relationships which are peer to customer, peer to peer and sibling to sibling to accurately representing AS business relationships as well as the Autonomous Systems and Internet Exchange Points. The visualization shows that the some countries in Africa rely on other countries for their internet of for traffic exchange. The coastal countries have more AS relations and a more clusted internet traffic compared to the inland countries. And some countries do not have Autonomous systems of their own for example Swaziland, we also discovered that although some countries do not have much internet interconnections other countries have more for example countries like South Africa and Mauritius.

This country can be seen as an economically driven country in Africa thus it has more IXPs and ASes active than most countries in Africa. As more countries in other continents trade with South Africa more, this puts it in an advantage of improving its economy thus internet connection is quite a vital source in this case as it provides a source of connection with other continents that are economically interested in South Africa.

Contents

List of Figures

1 Introduction

1.1 Problem Statement

As the use of the internet is increasing at a high rate it is becoming important to be able to model economic forces that drive evolution of the Internet topology and its hierarchy. inferring Autonomous System (AS) relationships is and accurately representing AS business relationships, and to provide a geographical visualization of Internet interconnections and traffic exchange in Africa is an approach to visualise these economic forces that drive evolution of the internet.

1.2 Objectives

The main objective of the project is to be able to model the economic forces that drive evolution of the Internet topology and its hierarchy. This is accomplished by developing the Africa Internet Visualiser Web Application that provide a geographical visualization of Internet interconnections and traffic exchange in Africa.

After the completion of the project we hope to visualise these economic forces that drive evolution of the internet and to accurately represent AS business relationships.

1.3 Scope

1.3.1 In Scope

- View different types of AS relationships (customer-to-provider, peer-to-peer, and sibling-to-sibling)
- View which ASNs are available in each country and their relationships.
- View which ASNs are available at each African IXP
- View all IXPs and ASNs in Africa
- View relationship links between queried countries
- System checks for new dumps of data in every 7 days and update the visual data.
- Allow for zooming , panning, and filtered visualization.

1.3.2 Out of Scope

- Registering of new IXPs, ASNs, and relationships

1.3.3 Inputs and Outputs

The user should be able to input the name of 2 countries and the output should be a display of the ASN interconnections between the two requested countries. This will be a way of filtering. Users should also be able to view the map at different levels i.e. continental level and national level. The input of a user zooming into the map, specifically a country, will result in the output of the map showing ASN relationships of that country alone. The user can also provide an input such as clicking on an IXP and the output in such a case would be that the web application displays all the ASN's available from that specific IXP. The user can filter according to the type of ASN relation type (e.g peer-to-peer) and the web application will output the display according to the filter input. The response time for these should be instant, meaning the user should not wait for any processing to happen or have any sort of delays.

1.3.4 Feasibility, Resources and Constraints

The resources to be used are datasets from CAIDA to visualize the ASN relationships data, data from Packet Clearing House for the IXP data and ASNs connected to each IXP. The application will also make use of MaxMind for IP geolocation information. The Web Application is developed on d3.js framework for data visualization, JSON for formatting the data from the APIs used, HTML and JavaScript for the structure of the web interface and handling user interactions. The development environment that is used if Django. This environment is compatible with most computing devices but might pose a constraint to devices with storage less than the required storage by the software. The server memory that is required to pre-process the data of the project might also pose as a constraint. Since this is a web application, this project needs a constant internet connection and hence will not be able to work when there is no internet connection. The stated scope is realistic as all the resources required to implement the project are available, mostly online. Thus, the time allocated is enough for a successful completion of this project.

1.4 Research Approach

The project will take an **Agile** software engineering approach.

This method provides the advantage that the project will be broken down into small iterations and ensure they are fully-tested as the final system is being implemented. As user requirements will be the driving factor of the project, this method caters for any changes that might occur with the user requirements thus ensuring that the changes are included in the respective iterations. These iterations will be presented frequently and regularly to the user to ensure an immediate response to the release. For such an approach, user involvement with the software being developed is crucial. The development cycle for the agile approach could be simply put as: *Analyse, Develop, Test.* This is done for each feature of the system iteratively until the final system is completed. The implementation of each feature will be done is such a way that it is shippable before moving to the next iteration. Features are given priority according to the user requirements.

As the time scale for the project remains fixed regardless of the changing requirements, the development team will not have any prescriptive processes to follow but rather will be required to utilize their skills to their full potential. This means that each member in the development team should be aware of their skills thus take on activities that will exploit their skills in their own development way of working.

2 Requirements Captured

2.1 Functional Requirements

The functional requirements explains how a system must behave. This is illustrate with the the behavioral diagrams, the use case and the sequence diagram.

2.1.1 Use Case

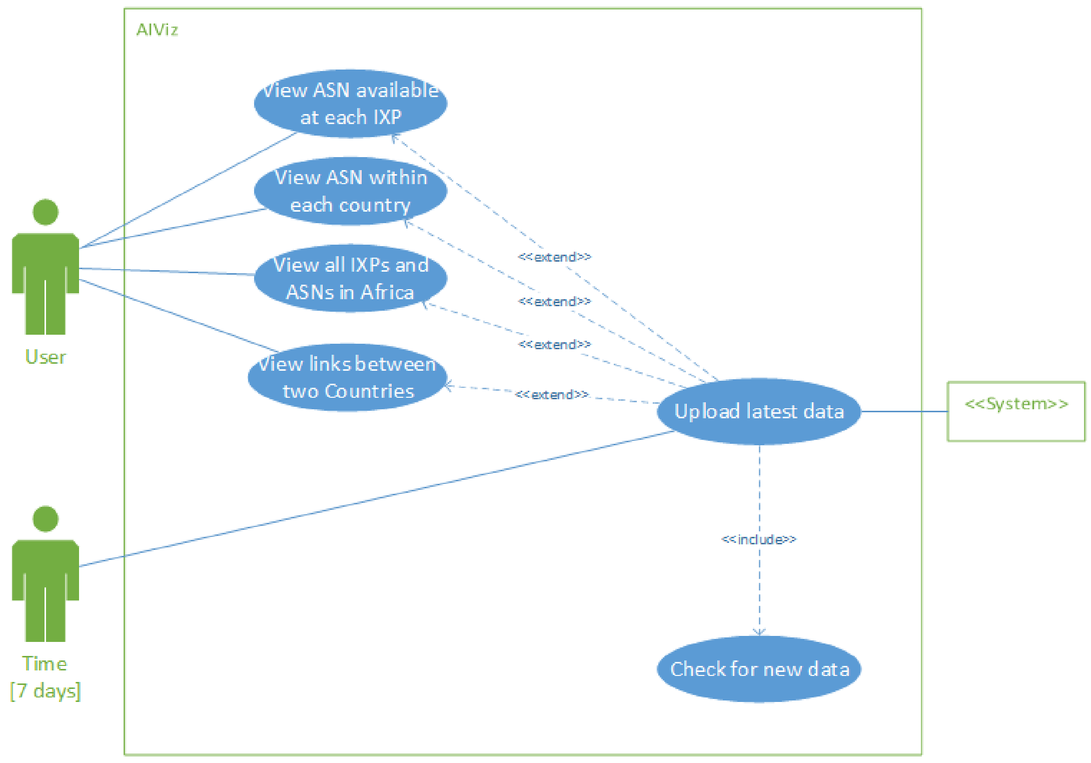

AIViz
View ASN available at each IXP
View ASN within each country
View all IXPs and ASNs in Africa
View links between two Countries
User
Time [7 days]
<<extend>>
<<extend>>
<<extend>>
<<extend>>
Upload latest data
<<System>>
<<include>>
Check for new data

Figure 1: Use Case Diagram

Figure 1 is a use case diagram for the Internet Visualizer Wep application. It illustrates the interactions that user has with the application and the relationships between use cases. The user can choose to view ASN available at each IXP, view ASN within each country, view all IXPs and ASNs in Africa, or view relationship links between two countries. All these depend on the data from different sources that is updated and uploaded into a database by the system every after 7 days.

2.1.2 Use Case Narrative

Use Case: View ASN available at each IXP
Actor: User
Main Flow:

1. Press IXP View

2. Hover over IXP
3. System displays the ASNs available at that IXP

Alternative or Exception Conditions:

1. Zoom and pan through the map
    - Hover over IXP
    - System displays the ASNs available at that IXP

Use Case: View ASN within each country
Actor: User
Main Flow:

1. Press Country View
2. System asks for a country name
3. User enters country name
4. System zooms in to that country
5. System displays the ASNs available in that country

Alternative or Exception Conditions:

1. click on the country
    - System displays the ASNs available in that country
2. Zoom into a country
    - System displays the ASNs available in that country
3. If the country name does not exist, the system displays error message

---

Use Case: View all IXPs and ASNs in Africa
Actor: User

Main Flow:

1. Press Continental View
2. System displays all IXPs in Africa
3. As the user zooms in and pans through the map, the system displays the ASNs

---

Use Case: View ASN relationship links between two countries
Actor: User
Main Flow:

1. Press Links View
2. The system asks for two country names
3. Enter two country names
4. The system displays the ASN relationship links between the two countries and the relationship types

Alternative or Exception Conditions:

1. If the user enters invalid country name
    - System displays error message

Use Case: Upload latest data
Actor: Time (system)
Main Flow:

1. System checks for new data after every 7 days
2. System uploads the latest data into the database

Exception Conditions:

1. If there is no new data
    - Process aborts

2.1.3 Sequence Diagram

Figure 2 shows the sequence diagram initially designed for the system. The difference between the initially designed sequence diagram and the actual implementation is the method file names, the functionality of the methods is the same but the names are different.

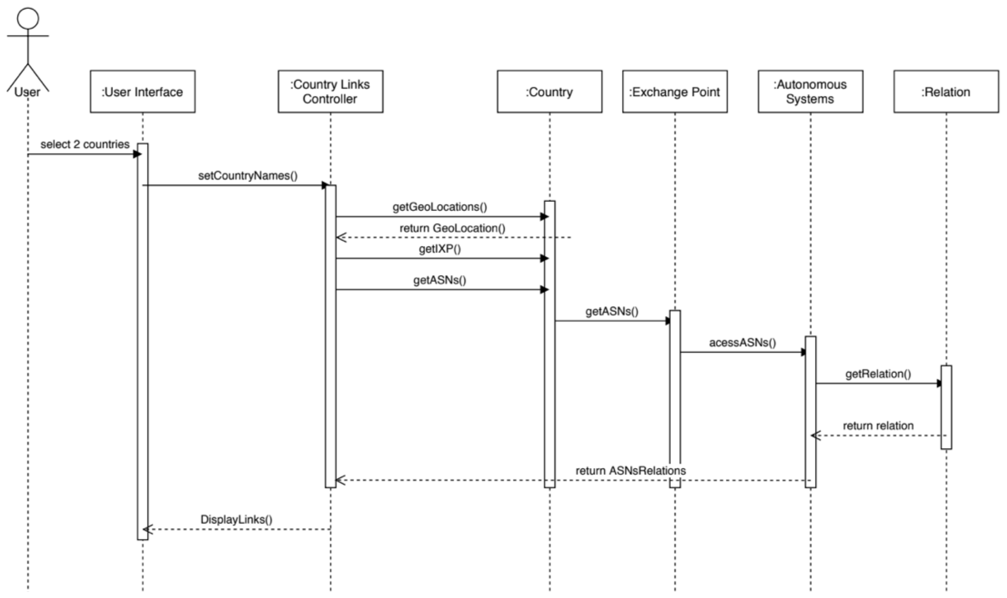


Figure 2: Sequence diagram

2.2 Non-functional Requirements

Performance requirements
The system is should be fast when rendering data from the database to the interface, it will be reliable and accurate by showing all ASNs, IXPs and relationships the same way. The system will have real time data update with its data sources for information accuracy. It should be able to handle large number of requests without crashing or freezing.

Usability requirements
The system will allow uses to easily navigate and find the internet traffic information they are interested in. It will also incorporate features that assist users in certain operations and have keys that describe the meeting of the data represented on the internet.

Volume and storage requirements
The system will use relevant data structures that will help meet the needs of the system and have enough volume of storage for any possible amount of data, the system should therefore be able to store large volume permanently without losing the information by using a permanent storage like a database system. In case of any data losses the system is going to have automatic data upload and update on every 30 days.

Configuration / compatibility requirements
Configuration requirements pertain to the choice of hardware, software, and documentation. Our system will only run on large screen devices like Desktops and laptops, and it should be conjured to work in any web browser. During the implementation and development of the system the system needs to follow some basic User interface and User experience fundamentals to ensure usability of the system.

Reliability requirements
The system should be able to update the data in the database once in a while to make sure that it has relevant data at all times. The system should use trusted API that provides consistent and reliable information. So it will regularly need preventive and corrective maintenance. Maintenance might signify scalability to grow and improve the system features and functionalities.

Backup / recovery requirements
To ensure permanent storage of data the system will use a data base to store the data that it processes from its data sources like the API. The timely auto data upload of the system will be put in place in cases of data losses to automatically upload the data again to the databases. To prevent any crashes from unavailable data resources the system should use try catch functionality to avoid crashing when the data source is unreachable or it provides unexpected information.

3 Design Overview

3.1 System Architecture

The implementation of a software architecture to represent a system ensures that the system is decomposed into subsystems and illustrates how these subsystems communicate and interact with each other. For the purpose of this project, the system makes used of a **layered system architecture**.

In a layered system, each layer is a representation of a service the system is composed of. As each layer is separate from another, changes made to one layer almost do not affect the other layers of the system and hence making changes to a layer is easier rather than making changes to the whole architecture. This enables the system to smoothly integrate user requirements as they evolve as well as easily detect and manage bugs. A layer that represents a subsystem within the overall system is created such that it addresses a different concern within the system thus the components that make a specific layer are all of the same concern. These components in a layer then have to have a strong relation while being stand alone at the same time. Futhermore, minimal interaction and relation between subsystems or layers is encouraged. The different layers are all linked to the same database, meaning that all the data that is used within each layer is from the same database. Thus, the chosen layered architecture structure is one that possess and caters for all the aforementioned characteristics. However, this structure also has downfalls such as it may affect overall performance of the application as a layer above another must be connected to that at the lower level. It is also generally difficult and expensive to scale such a structure [2].

The architecture of the system can be seen in Figure 3 below.

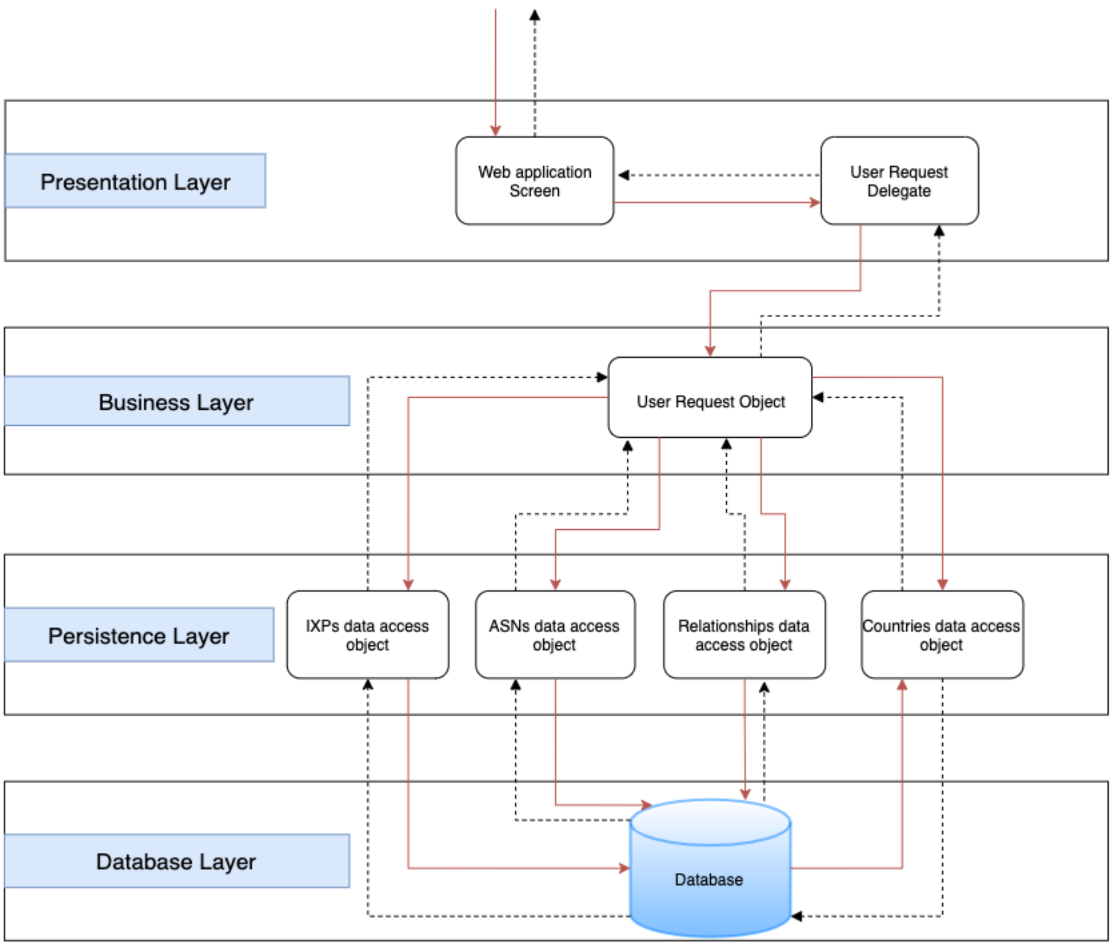


Figure 3: System Architecture Diagram [2]

The *red solid lines* indicate the flow of a request down to the database to retrieve the data, while the *black dashed lines* indicate data flowing from the database to the screen to be displayed to the user as per request.

3.1.1 Presentation Layer

This layer is responsible for providing the user interface and computation results to the user. This is where the interaction tools are presented.

For the purpose of this project, the classes that form the components of this layer are integrated in the **JavaScript (*bundle.js & index.js*), CSS *styles.css* and HTML *index.html* files**. With this, the user will be able to view the visualization map and interact with the system. In this layer, the user is able to change the view of the data (e.g *Continental view* to view IXPs in each country and *National View* to view ASNs and their relationships in each country)

3.1.2 Business Layer

This layer is responsible for determining the manner in which the application behaves. An example of the work that is done in this layer is if a user wished to view data in *Continental View*, then this layer is responsible for obtaining the *Countries* and *IXP* data from the respective objects in the lower layer and passing it to the above layer for the user to view [1].

The classes that form the components of this layer are intergrated in the **Python file**, namely the *Views.py* file.

3.1.3 Persistence Layer

In this layer, the components of this layer are responsible fore doing the technical stuff and persisting the data to the database [3]. Class objects were created inside the *Views.py* file and were passed to the *index.html* which was used at the communication medium between the JS and Python files.

3.1.4 Database Layer

This is where the database is stored, with all the table and the relations between each of them. Data is extracted and added to this layer accordingly. The data that is added into the database is mainly retrieved from external APIs. The components of this layer are found in the *Models.py* file where all the tables and table relations are defined.

3.2 Analysis Class Diagram

The analysis class diagram of the system can be seen below.

The classes as seen in Figure 4 were implemented with their respective attributes. The relationships between classes are further explained in subsection 4.4. Since the nature of the Django database used enables working with the data directly from the database, in order to extract the data from the database, objects of the tables were created thus to get the data the objects were manipulated such that data is extracted directly from the

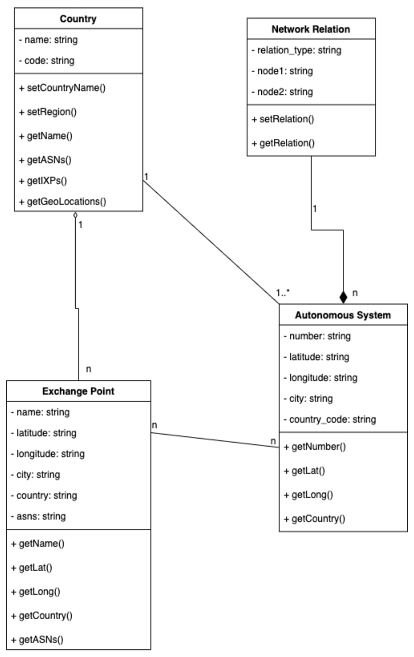


Figure 4: Analysis Class Diagram

objects to get the required data. Rather than implementing the methods as seen in Figure 4, the extraction of the data was done in such a way that it does what the methods show in the figure.

4 Implementation

This section explains in detail how the whole web application was implemented. It explains how the data was prepared and the data structures used, and how the processed data is visualized. Furthermore, it discuss significant methods of each class and how classes relates to one another.

4.1 Data Structures

Figure 5 illustrates how the data is processed and the data structures used. In the figure, circles depict the processes, diamond shape are the options, rectangles represents data structures, the open rectangle is the database, the directional connectors are the data flows. The application gets relationship data and customer cones data from the CAIDA website and store it locally. The ripe API is used to check if each ASN is in Africa or not. For each ASN that is in Africa, the ripe API is used to get the IP for each ASN. And for each ASN IP, ip-api is used to get the geolocation for each ASN (latitude, longitude, city and country). All African ASNs and their geolocation details are stored in a dictionary, and from the dictionary into the database. If two ASNs form a relationship and are from Africa, their relationship is kept in either a p2c dictionary or a p2p dictionary depending on the relationship type that exist between them. From the relationships dictionaries, relations are stored in the database. The s2s relationship is built from the customer cones dataset. Each line in the customer cones dataset presents a provider and all the customers connected to that provider. For customers that exist in the database, itertools library is used to create all combinations of two ASN per combination which are stored as sets in a list. Each combination of ASNs is checked if there exist a direct link between them (p2p relationship), and if there is direct link, the link is updated as s2s relation in the database. The PCH API is used to get all the IXP data and their details. For each African IXP, the PCH subnets API is used to get all the ASNs connected to that IXP. Since IXPs are no longer used anywhere in the code, they are stored straight into the database together with their details and their ASNs. All the African country names and country codes are kept in dictionary and stored into a database during data processing (this happens only once). The processing of whole data happens in the background, and it is updated every after 7 days.

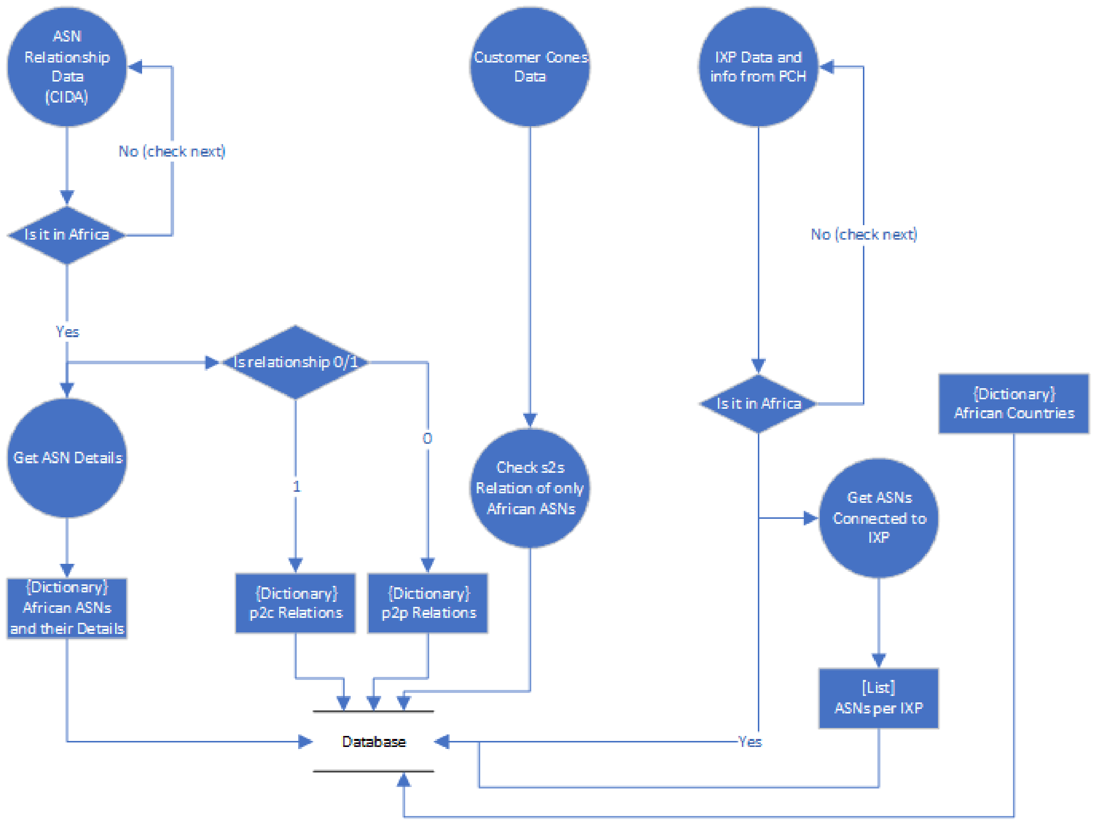

ASN Relationship Data (CIDA)
No (check next)
Is it in Africa
Yes
Is relationship 0/1
1
0
Get ASN Details
{Dictionary} African ASNs and their Details
{Dictionary} p2c Relations
{Dictionary} p2p Relations
Customer Cones Data
Check s2s Relation of only African ASNs
IXP Data and info from PCH
No (check next)
Is it in Africa
Yes
Get ASNs Connected to IXP
[List] ASNs per IXP
{Dictionary} African Countries
Database

Figure 5: Data Processing Workflow

Figure 6 illustrates a workflow for visualizing the data that is stored in the database. Since Django doesn't allow the classes to be accessed directly from python into javaScript, the data is accessed in the javaScript through HTML. The database is queried for each data class. In the javaScript file, the data from each class is kept in a dictionary, each in its own dictionary. From the dictionaries it is then filtered based on user's requests and visualized in the map.

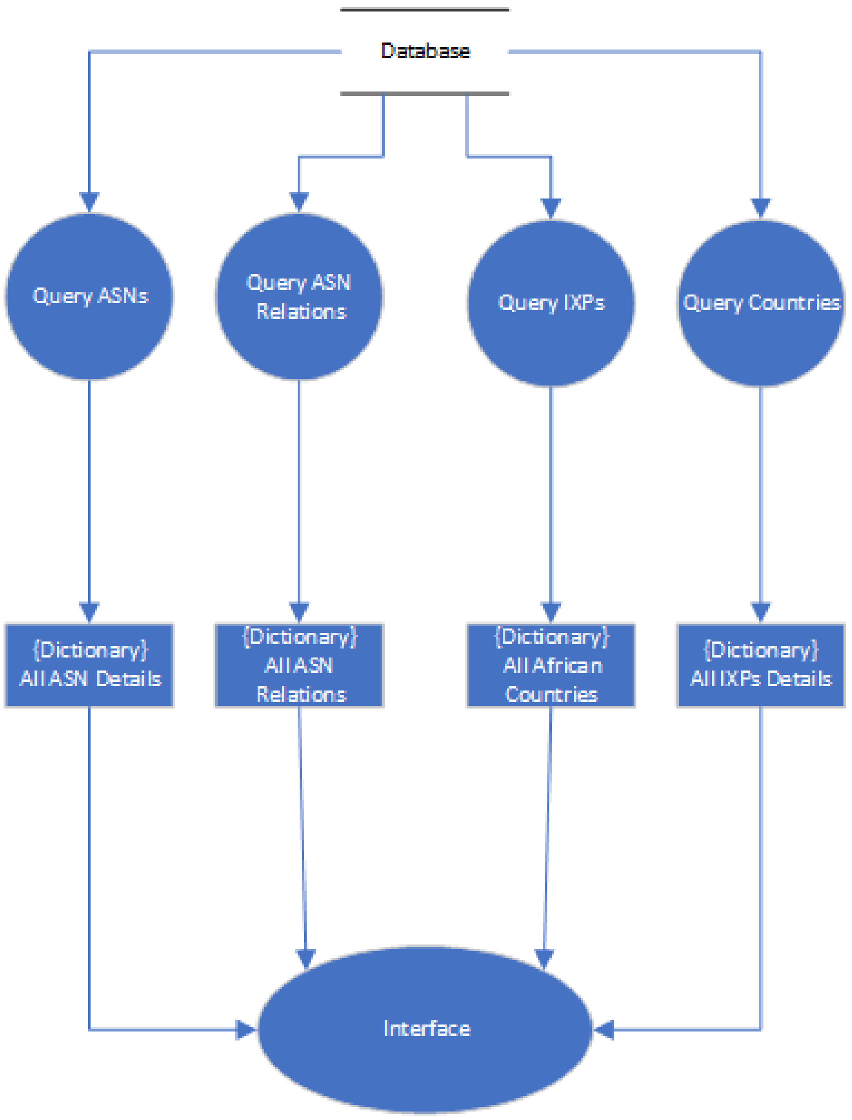

Database
Query ASNs
Query ASN Relations
Query IXPs
Query Countries
{Dictionary} All ASN Details
{Dictionary} All ASN Relations
{Dictionary} All African Countries
{Dictionary} All IXPs Details
Interface

Figure 6: Data Visualization Workflow

4.2 User Interface

The User Interface (UI) is built on three different technologies; HTML, javaScript and CSS. For the Africa Internet Visualizer application, HTML is used to describe the structure on the web app page and describing all elements in it, from labels to buttons and popups. CSS is used to set the styles for each element. JavaScript handles how the UI behaves and how the data is displayed for all user interactions. The application uses d3.js version 4 framework to visualize the data in a dynamic map.

The map is created from the JSON file which contains the topology data of all countries in the world. The d3 topojson.feature in used to store the related geometries from the topology data in the same file and create features of all countries as objects. The topojson allows the boundary lines for each country to be computed automatically. A tsv file containig textual data for each country feature is used to get the country name so that it is displayed when you hover over each country. The d3 SVG is used to render the map geometries in the browser. Since we are using a 2D display, the map spherical coordinates need to be projected to Cartesian plane. This is done using d3.geoMercator. A path ganerator (d3.geoPath) is used to approximate the projected coordinates in the SVG.

The d3 zoom and transition functions is used to allow the users to zoom and pan through the map. These functions are manipulated to allow for zooming into a clicked country. Each country object is highlighted everytime the user moves a mouse over it. This is implemented by changing the style for that country object and resetted on mouse out. Also, if the mouse in over a country, the country name is displayed. This is implement with the javascript tooltip. The tooltip is also used to display the ASNs connected on each IXP and to display the ASN number when the user hover over IXP or ASN.

4.3 Most Significant Methods

Since our project is a Django project our classes are defined as models and they directly map to the database. They do not have access controllers like private or protest so we found that it was not necessary to define access modifiers if the class attributes do not have any sort of privacy.

The Models.py is where the classes that map to the database are defined. In this class only the definition of classes is allows, we cannot define any logic in methods.

However we have the most significant methods in our View.py file called index() which one of them as a main method as it renders the interface of the web application and runs all other methods such as Makefiles, MakeASN, MakeIXP and many more others.

4.4 Special Relationships between classes

We have four classes named Country, ASN, Relations and IXP. The ASN class has Country class as a foreign key which maps each asn country code to a country object.

The IXP class contains asn attribute which is a ManyToMany relationship because in each IXP there's a collection of ASN objects.

4.5 Special Programming techniques

In our project we used Django python framework for backend which provides rich and easy to use tools to work with database without having to manage or write any SQL code.

The project involves processing large sets of data, therefore we used Parallel programming to maximise the computer resources when processing this data for faster results.

5 Program Validation and Verification

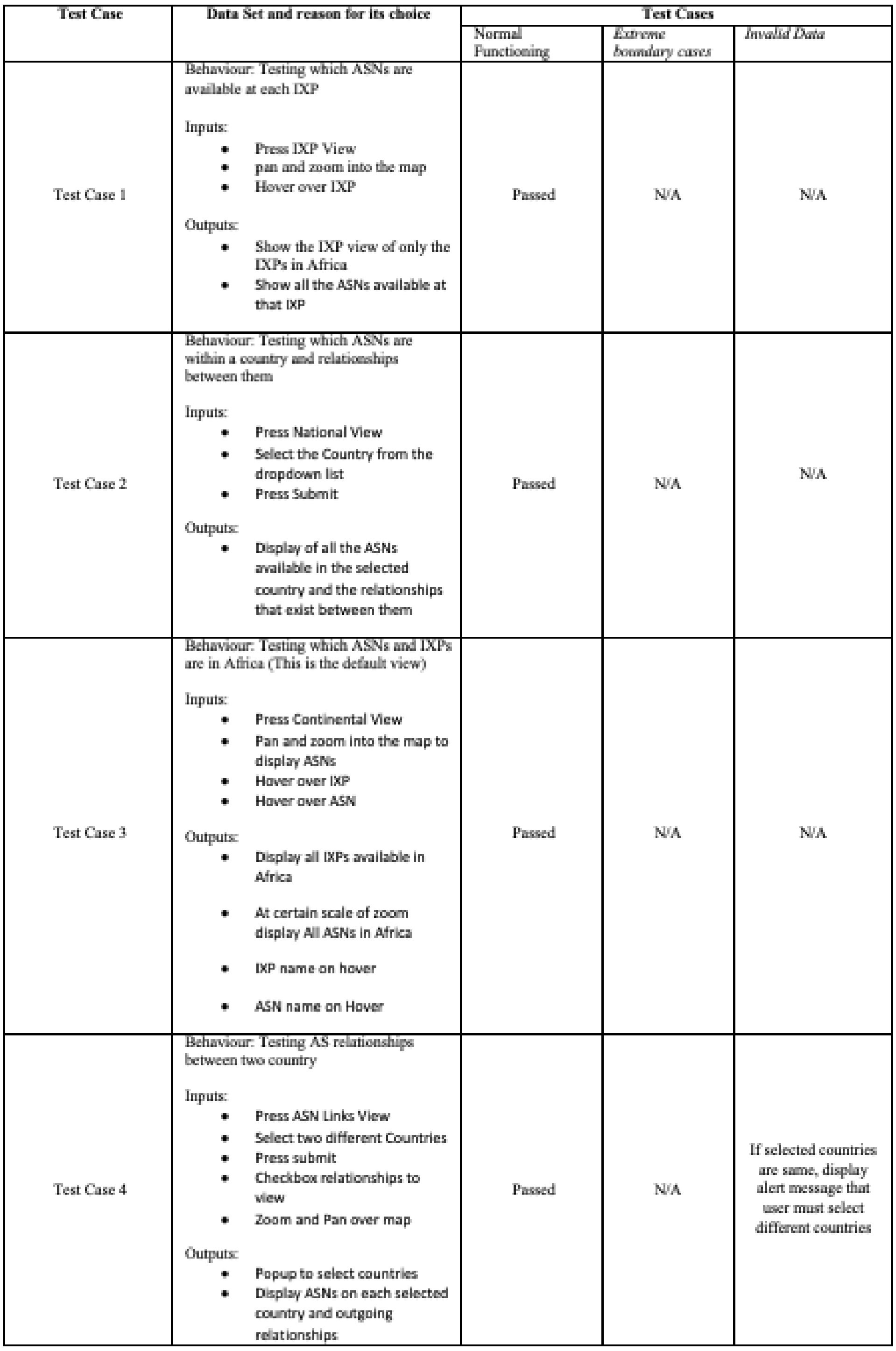
5.1 Test Plan

As the system presented is a user interactive web application, the system will be tested using **Behavioural Testing** and **Use-case based black box** techniques.

The system requires the user to be able to manipulate the viewing of the map and filter by selecting countries they wish to view information on, of which is mainly a click of a button. The system does not require the user to provide any typed out input thus the behaviour of the system will be exploited by testing all the functionalities that are presented on the web application screen. The basis of the functionality of the web application is highly reliant on the pre-processed data in each respective class thus the response of the system is one that is dependent on the functionality of the classes and methods. Using user-test cases to test the behaviour of the system will ensure that the behaviour of the implemented classes, methods and their interaction with one another is also tested as the data presented on the map is a result of the processed data. The ability of the system to respond to its manipulation according to the user requirement will ensure that the validation of the system is tested by using black-box testing techniques.

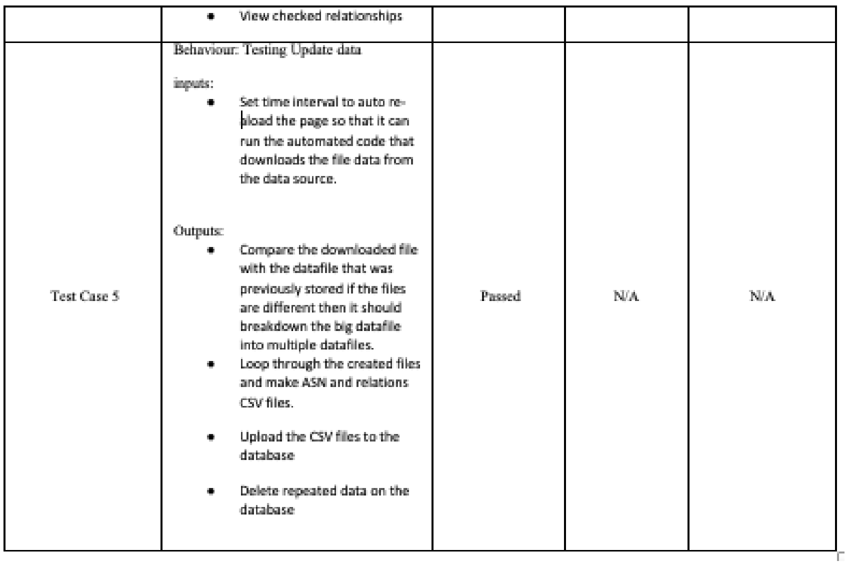
5.2 Correctness Validation

Figure (7 & 8) is the table explaining the steps taken to validate the correctness of the program. The tests test for the functionality of the web application as illustrated in the use case diagram. The figure explains the behaviour being tested, the inputs and if it behaved in the expected manner. In all the views, if the user hover over a country, IXP or an ASN, a tooltip appears displaying a country name, IXP name, or the ASN number. The user can zoom and pan the map in all the views. When a new view is selected, the map zoom resets to default.



Test Case	Data Set and reason for its choice	Test Cases		
		Normal Functioning	*Extreme boundary cases*	*Invalid Data*
Test Case 1	Behaviour: Testing which ASNs are available at each IXP Inputs: • Press IXP View • pan and zoom into the map • Hover over IXP Outputs: • Show the IXP view of only the IXPs in Africa • Show all the ASNs available at that IXP	Passed	N/A	N/A
Test Case 2	Behaviour: Testing which ASNs are within a country and relationships between them Inputs: • Press National View • Select the Country from the dropdown list • Press Submit Outputs: • Display of all the ASNs available in the selected country and the relationships that exist between them	Passed	N/A	N/A
Test Case 3	Behaviour: Testing which ASNs and IXPs are in Africa (This is the default view) Inputs: • Press Continental View • Pan and zoom into the map to display ASNs • Hover over IXP • Hover over ASN Outputs: • Display all IXPs available in Africa • At certain scale of zoom display All ASNs in Africa • IXP name on hover • ASN name on Hover	Passed	N/A	N/A
Test Case 4	Behaviour: Testing AS relationships between two country Inputs: • Press ASN Links View • Select two different Countries • Press submit • Checkbox relationships to view • Zoom and Pan over map Outputs: • Popup to select countries • Display ASNs on each selected country and outgoing relationships	Passed	N/A	If selected countries are same, display alert message that user must select different countries

Figure 7: Tests Cases Table 1



	• View checked relationships			
Test Case 5	Behaviour: Testing Update data inputs: • Set time interval to auto re-pload the page so that it can run the automated code that downloads the file data from the data source. Outputs: • Compare the downloaded file with the datafile that was previously stored if the files are different then it should breakdown the big datafile into multiple datafiles. • Loop through the created files and make ASN and relations CSV files. • Upload the CSV files to the database • Delete repeated data on the database	Passed	N/A	N/A

Figure 8: Test Cases Table2

Figure 9 shows the test results for test case 1, taking ”Harare Internet Exchange as an example”. When user clicks on the IXP View, the app shows a text telling the user which view they currently in. This is helpful because the user will not have to select the everytime they are uncertain if they are in the correct view. When the user hover over IXP, a tooltip appears displaying the IXP name and a list of ASNs connected to that IXP. In the case where there are no ASNs connected to that IXP, a tooltip displays none under ASNs connected.

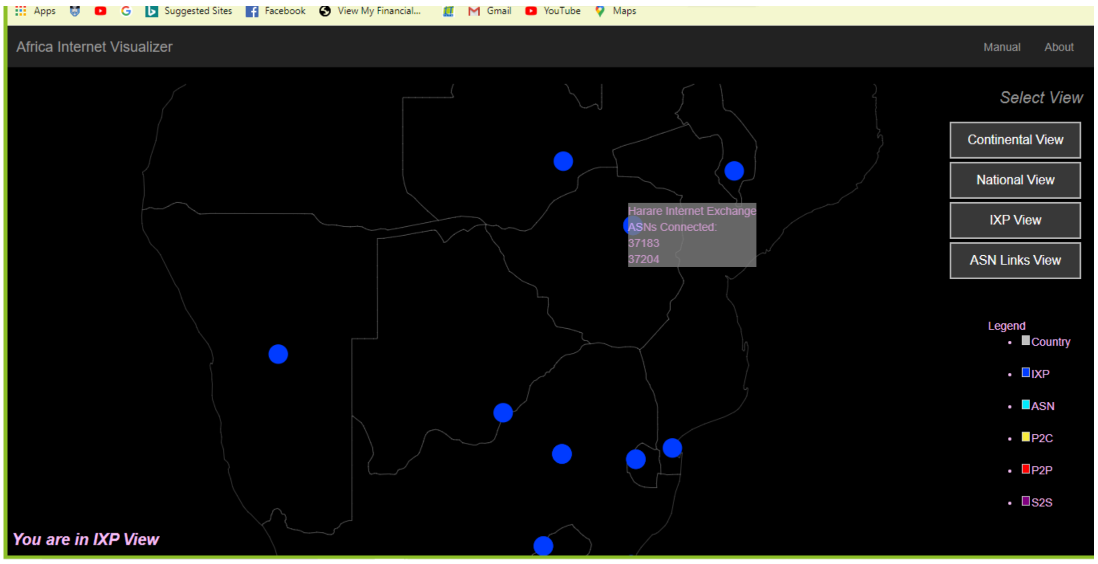


Figure 9: Test Case 1 Results

Figure (10 & 11) shows the test results for test case 2, taking ”South Africa” as an example. When the user press National View, a pop up window appears asking a user to select a country they want view (figure 10). From a dropdown, a list of all African countries appear to choose from. This style of a ”select” country was seen best as compared to user typing in the country name to avoid user typing errors. On the popup, a user can choose to ’Close’, which goes back to the previous view or ’Submit’ which takes the user to National view and displays the results. Figure 11 shows the ASNs in South Africa and relationships between them. The user can checkbox for the type of relationships they want to view for clear viewing of the lines. In the figure, s2s relationship is switched off.

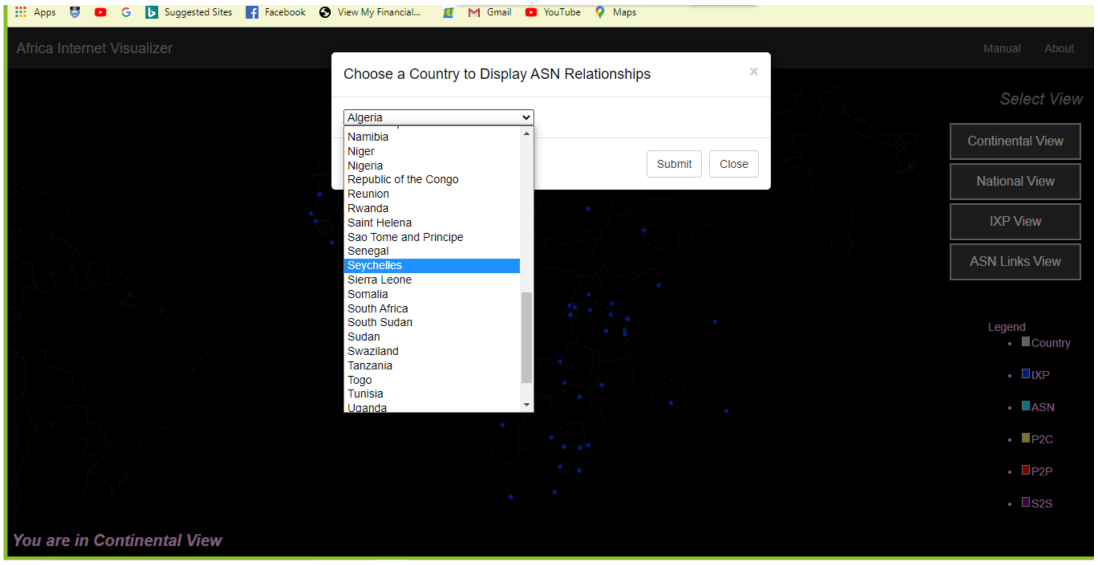


Figure 10: Test Case 2 Results: Select Country Popup

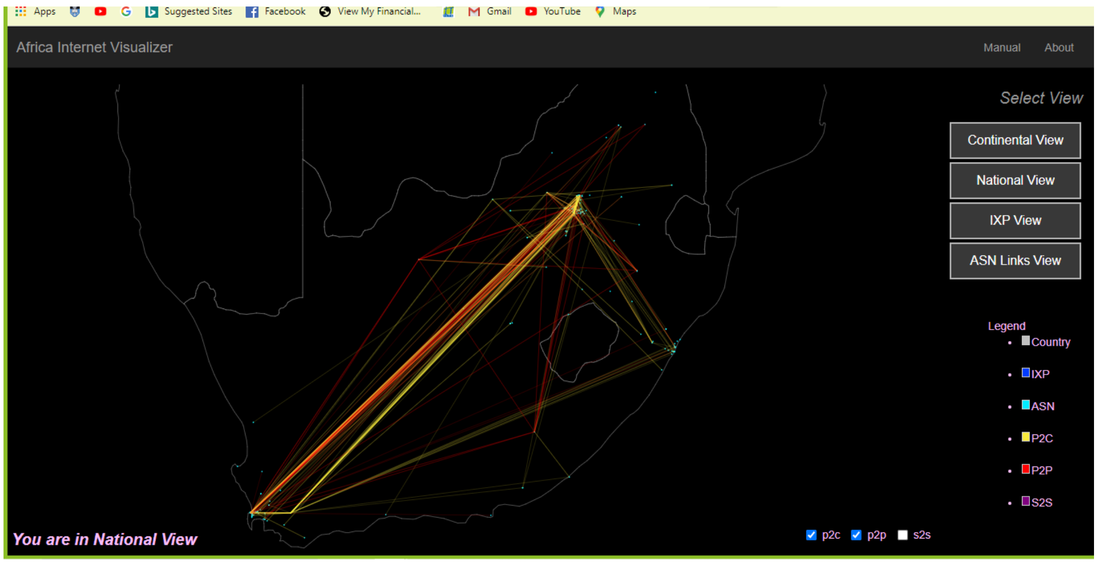


Figure 11: Test Case 2 Results: ASN Relationships

Figure (12 & 13) shows the test results for test case 3. Test 3 tests the Continental View which displays all the IXPs and ASNs in Africa. On the default zoom level, the user sees only the IXPs and if they zoom in to a certain level, it displays the ASNs and hide the

IXPs to avoid data congestion. When the user hover over IXP or the ASN, the IXP name or the ASN number is displayed.

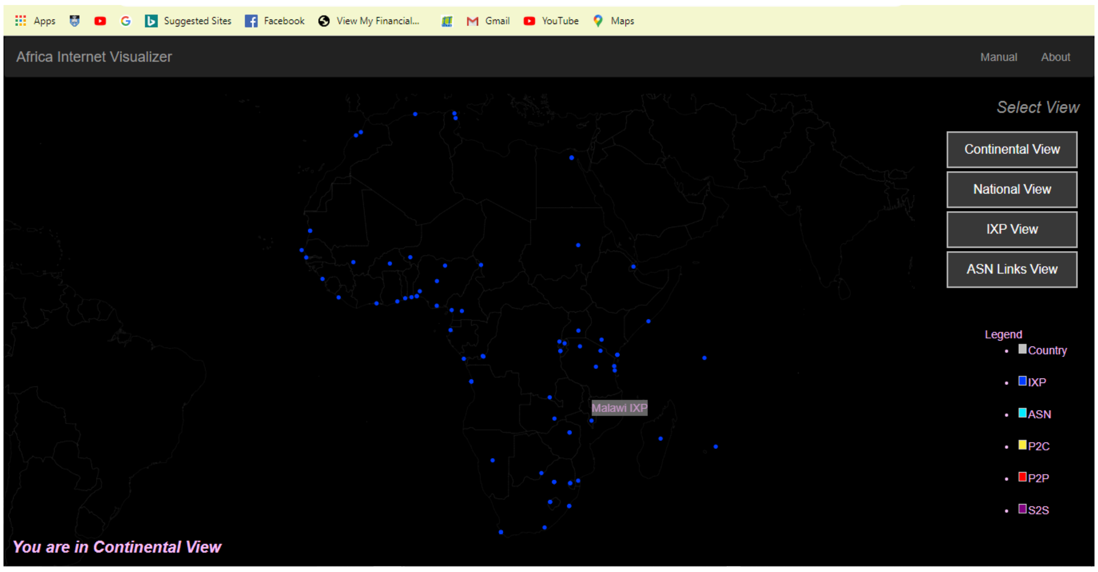


Figure 12: Test Case 3 Results: African IXPs

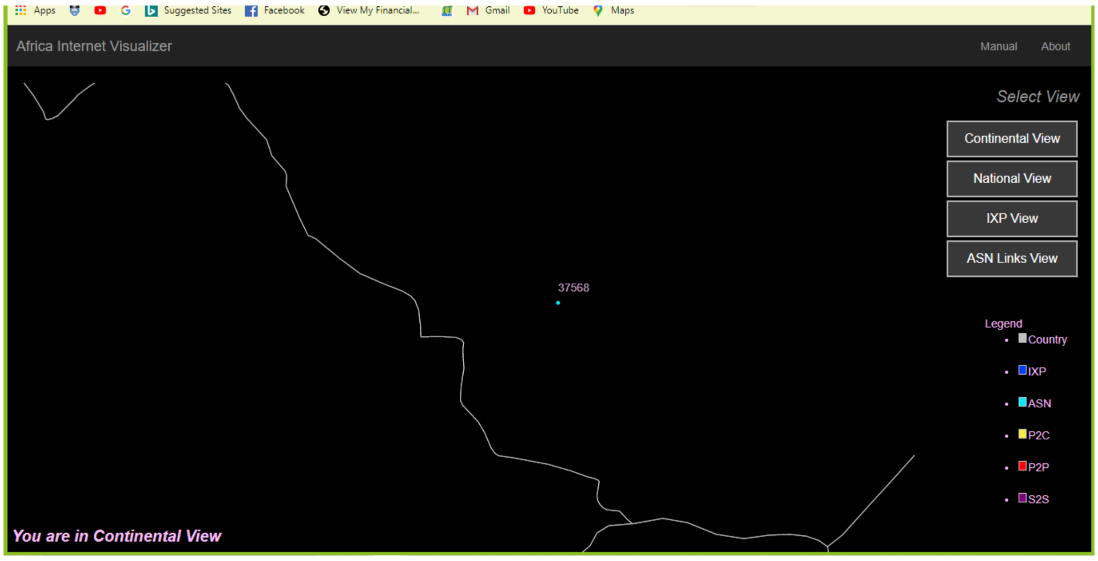


Figure 13: Test Case 3 Results: African ASN

Figure (14 & 15) shows the test results for test case 4, taking 'South Africa' and 'Mauritius' as an example. When the user select ASN Links View, a pop up window to select two countries appears. The use can select any two different African countries. If the selected countries are there same, it alerts the user to input two different countries because that would be the same as National View. the map displays the ASNs in the two selected countries and only the outgoing relationships between them. Again, the user can switch on/off the relationships. If the user select two same countries, the app alerts the user that they must select two different countries. Even though this is not really error, but the results would be as if the user viewed a National view.

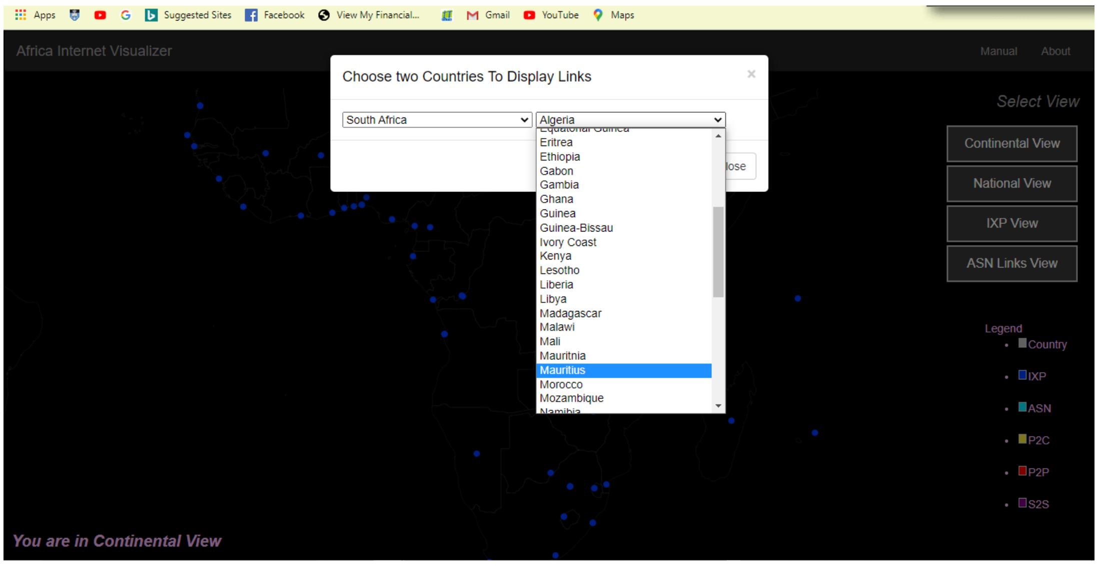


Figure 14: Test Case 4 Results: Select Countries Popup

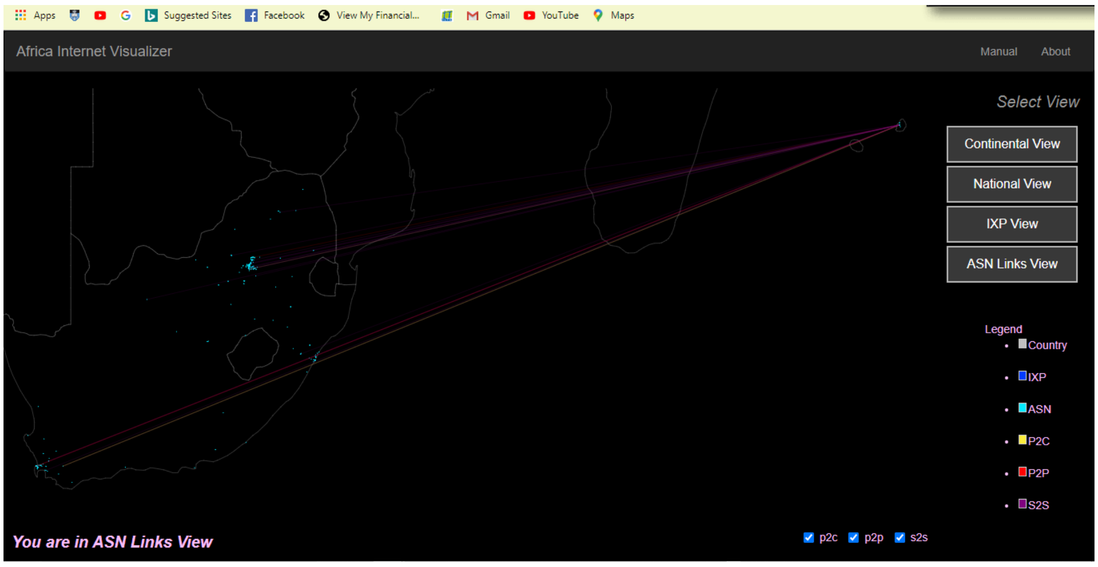
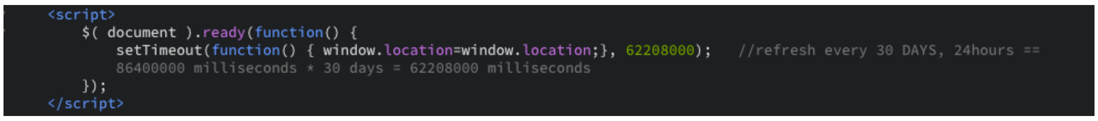


Figure 15: Test Case 4 Results: Relationship Links

Figure 16 shows the JavaScript script that is responsible for auto updating the database when the data source of ASNs and ASN relationships has been updated. The auto update happens every 30 days.

To differentiate between a page reload event that is responsible for checking if the data from the data sources has been updated, the index.html file sends an event variable to the view.py index method which should tell if the page reload was meant to auto update the data or its just another page refresh maybe from a user.

When the 30 days interval for checking data updates has been reached the files.py index method is called and initiates the process of auto updating the data in the database. The index method in files.py downloads the file from the data source, compare the downloaded file with the file that was previously on the unprocessed data folder which contains the data that was previously uploaded to the database. If these data files are not the same then the system is going to re upload the data from the recently downloaded file and delete the data file that was there before the download.

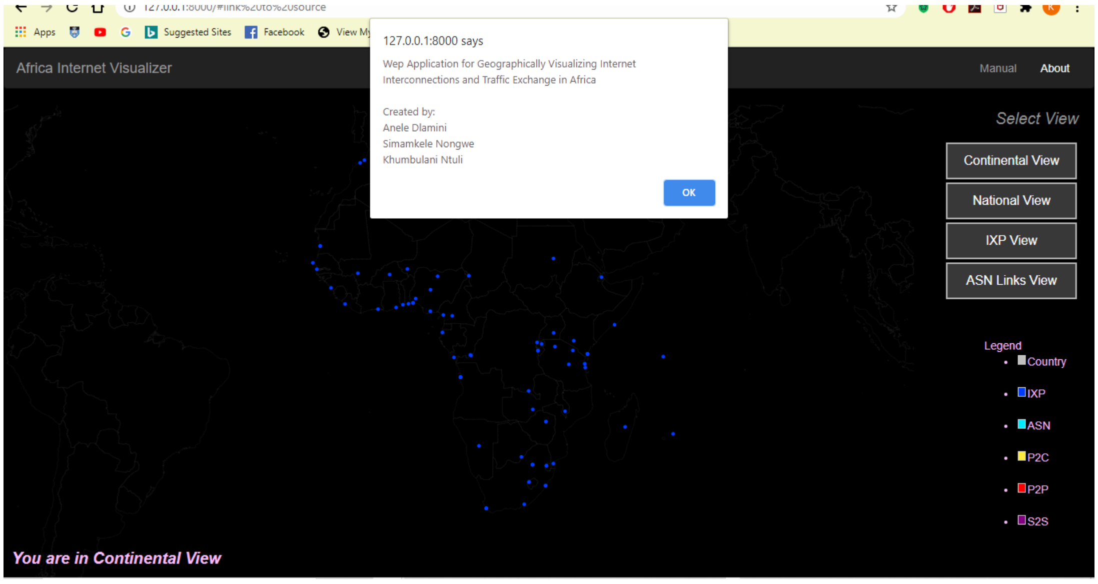


```
<script>
    $( document ).ready(function() {
        setTimeout(function() { window.location=window.location;}, 62208000);    //refresh every 30 DAYS, 24hours ==
        86400000 milliseconds * 30 days = 62208000 milliseconds
    });
</script>
```

Figure 16: Test Case 5 Results: Automatic Update of Data

5.3 Additional Features

In addition to the basic requirements of the system, the application displays the map legend showing what each colour on the map represents. The aim was to make the application as much informative as possible. On the navigation bar, the user can open the application manual by clicking 'Manual'. This opens the pdf manual in a new window tab. The 'About' button when clicked, a message alert window pops up telling the user a short summary of what the application does, and the developers of this application (figure 17).



127.0.0.1:8000 says

Wep Application for Geographically Visualizing Internet Interconnections and Traffic Exchange in Africa

Created by:
Anele Dlamini
Simamkele Nongwe
Khumbulani Ntuli

Figure 17: Additional Features: About

6 Conclusion

As the internet traffic in Africa was successfully visualised with this web application, the objective of the project has been met. The application serves as a model to the economic forces that drive evolution of the Internet topology and its hierarchy. An example of such is South Africa, a coastal country. This country can be seen as an economically driven country in Africa thus it has more IXPs and ASes active than most countries in Africa. As more countries in other continents trade with South Africa, this puts it in an advantage of improving its economy thus internet connection is quite a vital source in this case as it provides a source of connection with other continents that are economically interested in South Africa.

The overall performance of the system during the experiment was efficient and of satisfactory. All of the use cases presented and requirements captured in section 2 were met. A user is able to navigate without any hassle through the web application and the user input tools provided are user friendly. The use of drop-down boxes for user input provide the advantage of minimising user input error and also assists in those that may not be good with spelling. The system utilises a broad number of components thus showing diversity and creativity with the overall appearance of the system.

References

[1] José Esterkin. Design your software architecture using industry-standard patterns.

[2] Mark Richards. Software architecture patterns.

[3] Grzegorz Ziemoński. Layered architecture is good.