**COMMUNICABLE DIESEASE ANALYSIS AND VISUALIZATION**

**By**

**AKASHDEEP AWASTHI**

**Roll Number: 1900970140006**

**Under the Guidance of**

**Dr. RASHI AGARWAL**

**A PROJECT REPORT**

**In partial fulfillment of the requirements for post-Graduation degree**

**of**

**Master of Computer Applications**

****

**Department of Computer Applications**

**Galgotias College of Engineering and Technology**

**Greater Noida (U.P.)**

**affiliated to**

**AKTU, Lucknow**

**MAY 2022**

**COMPANY CERTIFICATE**

****

###### **DIGIPODIUM PVT. LTD.**

**CERTIFICATE**

Date: 20/May/2022

This is to certify that the dissertation entitled **“COMMUNICABLE DIESEASE ANALYSIS AND VISUALIZATION”** carried out by **Mr. AKASHDEEP AWASTHI**, student of **Master of computer applications** [2019-2022], of **GALGOTIAS COLLEGE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA**, is hereby accepted and approved as a credible work, submitted in the partial fulfilment for the requirement of degree of MCA from **AKTU, Lucknow.** It’s a Bonafede record of the work done by him under my supervision during his stay as a project trainee at **Digipodium Pvt. Ltd.** from March 25, 2022 to May 20, 2022. The system has been implemented and running successfully.

**Lakshman Rao**

(Trainer)

Digipodium Pvt. Ltd.



**Department of Computer Applications**

Galgotias College of Engineering and Technology,

Greater Noida

**CERTIFICATE**

Date: 20/May/2022

This is to certify that the dissertation entitled **“COMMUNICABLE DIESEASE ANALYSIS AND VISUALIZATION”** by **Mr. AKASHDEEP AWASTHI,** student of **Master of computer APPLICATIONS,** [2019-2022], of **GALGOTIAS COLLEGE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA, affiliated** to **AKTU, LUCKNOW,** is hereby accepted and approved as a credible work. It is further certified that this work has not been submitted for similar purpose anywhere else. His work has been found satisfactory for the partial fulfilment of the award of the degree of MCA.

**Internal Examiner External Examiner**

**Head of Department**

**DECLARATION**

I, **AKASHDEEP AWASTHI,** hereby declare that the project work entitled **“COMMUNICABLE DIESEASE ANALYSIS AND VISUALIZATION”** is an authenticated work. Under the guidance of **Dr. RASHI AGARWAL** for the partial fulfilment of the award of the degree of **MASTER of Computer APPLICATIONS** and this work has not been submitted for similar purpose anywhere else except toGALGOTIAS COLLEGE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA, affiliated **to AKTU, LUCKNOW.**

**Date: 20/May/2022 Name: AKASHDEEP AWASTHI**

**Place: Noida Roll No.:1900970140006**

**ACKNOWLEDGEMENT**

It is high privilege for me to express my deep sense of gratitude to all those faculty members who helped me in the completion of the project, especially my internal guide **Dr. RASHI AGARWAL** who was always there at hour of need.

My special thanks to **Dr. GAGAN TIWARI,** HOD- Department of Computer Applications, **Galgotias College of Engineering and Technology**, for helping me in the completion of project work and its report submission.

**Name: AKASHDEEP AWASTHI**

**Roll No.: 1900970140006**

**CONTENTS AT A GLANCE**

**Contents Page No.**

Preface

1. Introduction
   1. Introduction to the Project
   2. Platform Justification [09 - 13]
   3. Concept/Literature Study
2. System Analysis
   1. Existing System
   2. Feasibility Study [14 - 18]
   3. Proposed System
   4. Tools Used to Gather Information
3. System Design
   1. Software Requirement Specification
   2. Software Functional Specification [19 - 28]
   3. Data Flow Diagram
4. System Testing [29 - 35]
5. System Implementation [ 36 ]
6. System Input and Output Screens [ 37 - 49]
7. Some Code Lines [ 50 - 53]
8. Scope of Project, Limitations & Conclusion [ 54 - 55]
9. References [ 55 ]

**PREFACE**

As a part of MCA curriculum and in order to gain practical knowledge in the field of python and data science, I prepared this project report on "**COMMUNICABLE DIESEASE ANALYSIS AND VISUALIZATION (DATA SCIENCE IN PYTHON)**". The basic objective behind working on this project is to learn the use of techniques of data analysis and implement them using python programming language.

In this project, I have discussed many aspects like what the proposed system was, tools to gather information, requirement specification of the project, Flow diagrams, testing of the project and the final implementation.

Doing this project helped me a lot to enhance my knowledge in data science and python. This project is completely based on data analytics techniques and hence it is a bit difficult to understand complex algorithms. But the guidance of my trainer and HOD and faculty members helped me a lot in going with the flow easily.

1. **INTRODUCTION**
   1. **Introduction to the Project**

The title of project is **“COMMUNICABLE DIESEASE ANALYSIS AND VISUALIZATION (DATA SCIENCE IN PYTHON)”**.

Communicable diseases are diseases that are as a result of the causative organism spreading from one person to another or from animals to people. They are among the major causes of illnesses in Kenya and the entire Africa. These diseases affect people of all ages but more so children due to their exposure to environmental conditions that support the spread. Communicable diseases are preventable base on interventions placed on various levels of transmission of the disease.

And the very widely spread one currently the “Covid 19” is also one of the communicable diseases. So to present the report of such diseases, this project is very reliable one. As it will perform the data analytics on the dataset and presents the result in a way which a normal human can understand easily.

In a globalized world where are more and more trips and the transmission of infectious diseases is a major concern, there is a need to control and monitor the spread of disease. In this regard, there has been an increase in the development of computer applications to monitor data on various communicable diseases. Worldwide, there are a number of entities in the short term that aim to improve disease control systems, such as CDC (Center for Disease Control and Prevention of United States) and ECDC (European Center for Disease Control and Prevention) and who want to be at the forefront of disease control and prevention. To this end, they want to modernize data collection and visualization systems so that information on the state of public health is of a better quality and allows rapid and incisive responses to contain epidemiological diseases.

* 1. **Platform Justification**

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

In this project, I have used python and data science on a dataset for detecting the credit card fraud. Python has come a long way in the field of programming because of its wide range of libraries and short codes. This is the same reason why python and its libraries are widely used in data science projects.

Python is one of the world's most popular programming languages, and there are a few reasons why Python is so popular:

* Python’s syntax, or the words and symbols used in order to make a computer program work, is simple and intuitive. They're basically English words.
* Python supports various paradigms, but most people would describe Python as an object oriented-programming language. In an object-oriented programming language, everything you create is an object, different objects have different properties, and you can operate on different objects in different ways.
* Python integrates well with other software components, making it a general-purpose language that can be used to build a full end-to-end pipeline – starting with data, cleaning a model, and building that straight into production.

A data science platform is software that unifies people, tools, artifacts, and work products used across the data science lifecycle, from development to deployment. Organizations use data science platforms to create more maturity and discipline around data science as an organizational capability, instead of only a technical skill.

In many scenarios, Python is the programming language of choice for the daily tasks that data scientists tackle, and is one of the top data science tools used across industries. For data scientists who need to incorporate statistical code into production databases or integrate data with web-based applications, Python is often the ideal choice. It is also ideal for implementing algorithms, which is something that data scientists need to do often.

There are certain libraries of data science those were used in this project. These libraries enable us to easily understand the raw data that is not human understandable easily.

The libraries are:

* **Pandas:** Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.It is used for data wrangling and data manipulation because it allows a user to read data in, change it, look for missing values, read data out. Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on the top of the NumPy library. Pandas is fast and it has high-performance & productivity for users.
* **NumPy:** NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely. It is used for fast computation because it speeds up all of the different calculations that you're doing. Pandas actually uses NumPy under the hood for some of its calculations. NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array. Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open-source project.
* **Scikit-Learn:** It was originally called scikits.learn and was initially developed by David Cournapeau as a Google summer of code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA (French Institute for Research in Computer Science and Automation), took this project at another level and made the first public release (v0.1 beta) on 1st Feb. 2010. Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy. It is used for machine learning because it has all of the algorithms you'll want to use for regression, classification, and unsupervised learning. When you’re deep in the Immersive Data Science Bootcamp, you’ll be leveraging Scikit-Learn pretty heavily.
* **Matplotlib and Seaborn:** Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc. Matplotlib and Seaborn, both the libraries are used for data visualizations. With the help of these libraries, we can plot graphs which will help us in understanding the data more efficiently and effectively.

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representations for same variables for better understanding of dataset.

* 1. **Concept/Literature Study**

Communicable diseases refer to diseases that can be transmitted and make people ill. They are caused by infective agents (pathogens), e.g. bacteria and viruses, which invade the body and multiply or release toxins to cause damages to normal body cells and their functions. In severe cases, they may lead to death. These infective agents can spread from a source of infection (e.g. patients, sick animals) to a person through various routes of transmission

Communicable diseases are caused by very tiny or ganisms called germs and parasites. These germs are present everywhere- in air, water, soil, etc. When germs enter a healthy body, they multiply and upset the normal functioning of the body. This produces symptoms of a disease. If a person consumes infected food or water the HOME SCIENCE

Guidelines on Prevention of Communicable Diseases in RCHEs 7 1.2.2 Source of infection This refers to the reservoir where infective agents can live, parasitise and breed. It includes humans (e.g. patients, carriers and people with latent infections), livestock, insects and soil.

Crucial factors for the spread of communicable diseases include the infective agent, the source of infection, the mode of transmission and the host - the socalled ‘chain of infection’.

An infective agent is a microorganism (e.g. bacteria, viruses, fungi and parasite) that will cause an infection

This refers to the method of transfer by which the infective agent moves or is carried from one place to another. Some communicable diseases have more than one mode of transmission, e.g. chickenpox can be transmitted by airborne, droplet or contact transmission

• Implement standard precautions and additional infection control precautions based on the mode of transmission of respective communicable diseases to prevent evolution into outbreaks such as maintaining proper hand hygiene, enhancing environmental cleaning and disinfection, proper handling and disposal of body fluid, secretion and excreta, wearing surgical masks when having respiratory symptoms.

• If outbreaks of communicable diseases are suspected, promptly notify the Central Notification Office (CENO) of Centre for Health Protection (CHP), the Licensing Office of Residential Care Homes for the Elderly (LORCHE) of Social Welfare Department (SWD) and the Community Geriatric Assessment Team (CGAT) of Hospital Authority (HA) (if applicable) for follow-up investigation

**Why visualization?**

1. Partner with a group of stakeholders that have a problem

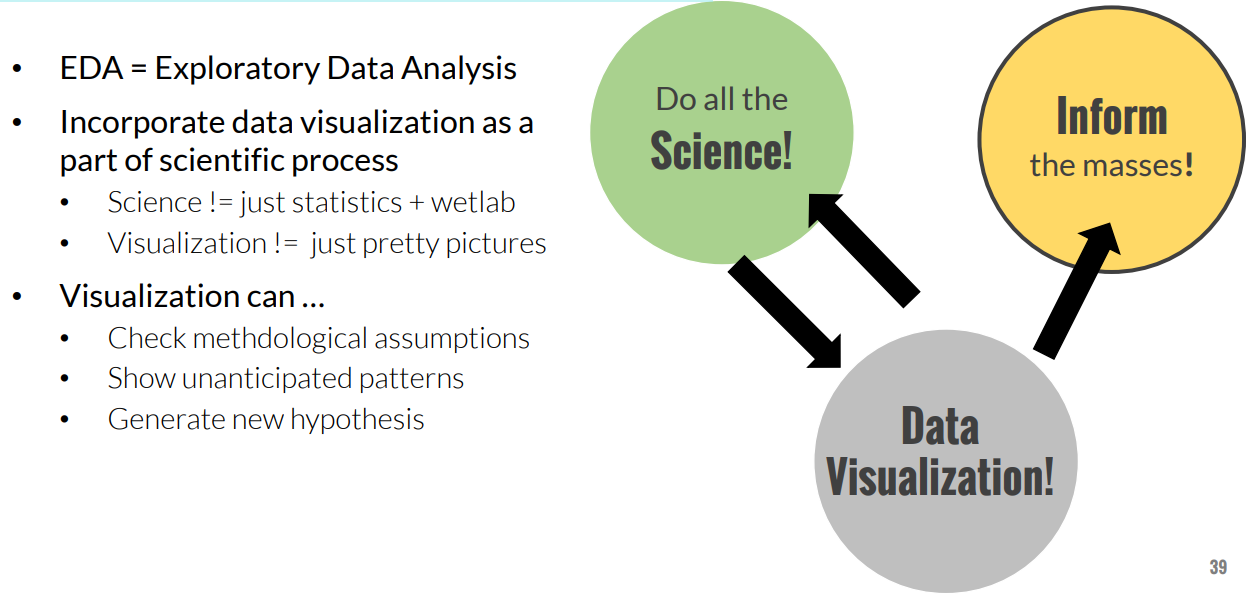
2. Ask what data stakeholders use (is it available)?

3. Ask what stakeholders do with the data [tasks]

4. Explore if other visualizations have addressed this problem and set of tasks

5. Test multiple alternatives (including new ones you develop) with stakeholders

6. Gather qualitative & quantitative evaluation data



* We identified articles published in English from January 1, 1980 to June 30, 2013 from five bibliographic databases. Articles with a primary focus on infectious disease visualization tools, needs of public health users, or usability of information visualizations were included in the review.

1. **SYSTEM ANALYSIS**
   1. **Existing System**

A myriad of new tools and algorithms have been developed to help public health professionals analyze and visualize the complex data used in infectious disease control. To better understand approaches to meet these users' information needs, we conducted a systematic literature review focused on the landscape of infectious disease visualization tools for public health professionals, with a special emphasis on geographic information systems (GIS), molecular epidemiology, and social network analysis.

The objectives of this review are to:

(1) identify public health user needs and preferences for infectious disease information visualization tools;

(2) identify existing infectious disease information visualization tools and characterize their architecture and features;

(3) identify commonalities among approaches applied to different data types;

(4) describe tool usability evaluation efforts and barriers to the adoption of such tools.

* 1. **Feasibility Study**

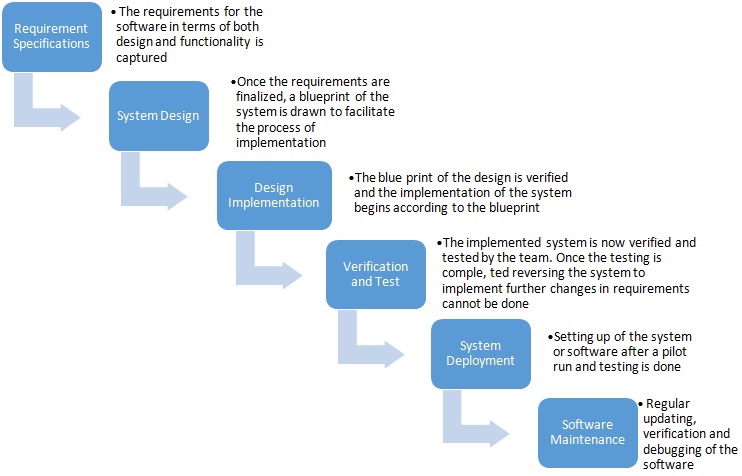
Feasibility Study in Software Engineering is a study to evaluate feasibility of proposed project or system. Feasibility study is one of stage among important four stages of Software Project Management Process. As name suggests feasibility study is the feasibility analysis or it is a measure of the software product in terms of how much beneficial product development will be for the organization in a practical point of view. Feasibility study is carried out based on many purposes to analyze whether software product will be right in terms of development, implantation, contribution of project to the organization etc.

When the client approaches the organization for getting the desired product developed, it comes up with rough idea about what all functions the software must perform and which all features are expected from the software. Referencing to this information, the analysts do a detailed study about whether the desired system and its functionality are feasible to develop.

This feasibility study is focused towards goal of the organization. This study analyzes whether the software product can be practically materialized in terms of implementation, contribution of project to organization, cost constraints and as per values and objectives of the organization. It explores technical aspects of the project and product such as usability, maintainability, productivity and integration ability. The output of this phase should be a feasibility study report that should contain adequate comments and recommendations for management about whether or not the project should be undertaken.

* **Types of Feasibility Study:** The feasibility study mainly concentrates on bellow five mentioned areas. Among these Economic Feasibility Study is most important part of the feasibility analysis and Legal Feasibility Study is less considered feasibility analysis.
* **Technical Feasibility:** In Technical Feasibility current resources both hardware software along with required technology are analyzed/assessed to develop project. This technical feasibility study gives report whether there exists correct required resources and technologies which will be used for project development. Along with this, feasibility study also analyzes technical skills and capabilities of technical team, existing technology can be used or not, maintenance and up-gradation is easy or not for chosen technology etc.
* **Operational Feasibility:** In Operational Feasibility degree of providing service to requirements is analyzed along with how much easy product will be to operate and maintenance after deployment. Along with this, other operational scopes are determining usability of product, determining suggested solution by software development team is acceptable or not etc.
* **Economic Feasibility:** In Economic Feasibility study cost and benefit of the project is analyzed. Means under this feasibility study a detail analysis is carried out what will be cost of the project for development which includes all required cost for final development like hardware and software resource required, design and development cost and operational cost and so on. After that it is analyzed whether project will be beneficial in terms of finance for organization or not.
* **Legal Feasibility:** In Legal Feasibility study project is analyzed in legality point of view. This includes analyzing barriers of legal implementation of project, data protection acts or social media laws, project certificate, license, copyright etc. It can be said that it is done to know if proposed project conforms legal and ethical requirements.
* **Schedule Feasibility:** In Schedule Feasibility Study mainly timelines/deadlines are analyzed for proposed project which includes how many times teams will take to complete final project which has a great impact on the organization as purpose of project may fail if it can’t be completed on time.
* **Resource feasibility:** Describe how much time is available to build the new system, when it can be built, whether it interferes with normal business operations, type and number of resources required, dependencies, and developmental procedures with company revenue prospectus.
* **Need of Feasibility Study:** Feasibility study is so important stage of Software Project Management Process as after completion of feasibility study it gives a conclusion of whether to go ahead with proposed project as it is practically feasible or to stop proposed project here as it is not right/feasible to develop or to think/analyze about proposed project again. Along with this Feasibility study helps in identifying risk factors involved in developing and deploying system and planning for risk analysis also narrows the business alternatives and enhance success rate analyzing different parameters associated with proposed project development.

**The following illustration is a representation of the different phases of the Waterfall Model:**



* 1. **Proposed System**

This strategic objective is the strengthening of multidisciplinary infectious disease research in Baltic, European and global dimensions, as a critically important part of health research, by unlocking the research potential at Riga Stradiņš University (RSU).

The overarching objective will result in a coherent work plan based on five operational objectives: establishing two new laboratories at RSU and providing their necessary infrastructure: Laboratory of Digital Immunological Visualization and Laboratory of Infectious Diseases Modelling; upgrading competences of personnel in immunology visualization and in-silico and mathematical modelling in infectious diseases and rare disease research; fostering integration in the ERA by profiling infectious disease as the specialization area of RSU in tandem with Latvia’s 3S priority in biomedicine and medical technologies; rebuilding the innovation system at RSU according to the principles of the Innovation Union; and obtaining quality evaluation, which will foster improved participation in the Horizon2020 program.

* 1. **Tools Used to Gather Information**

Information gathering is one of the most important part of the project. We often use certain information gathering techniques to do the task. Some of the techniques are as follows:

* **Brainstorming:** Brainstorming is used in requirement gathering to get as many ideas as possible from group of people. Generally used to identify possible solutions to problems, and clarify details of opportunities.
* **Document Analysis:** Reviewing the documentation of an existing system can help when creating AS–IS process document, as well as driving gap analysis for scoping of migration projects. In an ideal world, we would even be reviewing the requirements that drove creation of the existing system – a starting point for documenting current requirements. Nuggets of information are often buried in existing documents that help us ask questions as part of validating requirement completeness.
* **Focus Group:** A focus group is a gathering of people who are representative of the users or customers of a product to get feedback. The feedback can be gathered about needs/opportunities/ problems to identify requirements, or can be gathered to validate and refine already elicited requirements. This form of market research is distinct from brainstorming in that it is a managed process with specific participants.
* **Interface analysis:** Interfaces for a software product can be human or machine. Integration with external systems and devices is just another interface. User centric design approaches are very effective at making sure that we create usable software. Interface analysis – reviewing the touch points with other external systems is important to make sure we don’t overlook requirements that aren’t immediately visible to users.
* **Interview:** Interviews of stakeholders and users are critical to creating the great software. Without understanding the goals and expectations of the users and stakeholders, we are very unlikely to satisfy them. We also have to recognize the perspective of each interviewee, so that, we can properly weigh and address their inputs. Listening is the skill that helps a great analyst to get more value from an interview than an average analyst.
* **Observation:** By observing users, an analyst can identify a process flow, steps, pain points and opportunities for improvement. Observations can be passive or active (asking questions while observing). Passive observation is better for getting feedback on a prototype (to refine requirements), where active observation is more effective at getting an understanding of an existing business process.
* **Prototyping:** Prototyping is a relatively modern technique for gathering requirements. In this approach, you gather preliminary requirements that you use to build an initial version of the solution - a prototype. You show this to the client, who then gives you additional requirements. You change the application and cycle around with the client again. This repetitive process continues until the product meets the critical mass of business needs or for an agreed number of iterations.
* **Requirement Workshops:** Workshops can be very effective for gathering requirements. More structured than a brainstorming session, involved parties collaborate to document requirements. One way to capture the collaboration is with creation of domain-model artifacts (like static diagrams, activity diagrams). A workshop will be more effective with two analysts than with one.
* **Reverse Engineering:** When a migration project does not have access to sufficient documentation of the existing system, reverse engineering will identify what the system does. It will not identify what the system should do, and will not identify when the system does the wrong thing.
* **Survey/Questionnaire:** When collecting information from many people – too many to interview with budget and time constraints – a survey or questionnaire can be used. The survey can force users to select from choices, rate something (“Agree Strongly, agree…”), or have open ended questions allowing free-form responses. Survey design is hard – questions can bias the respondents.

1. **SYSTEM DESIGN**
   1. **Software Requirement Specification**

The production of the requirements stage of the software development process is Software Requirements Specifications (SRS) (also called a requirements document). This report lays a foundation for software engineering activities and is constructing when entire requirements are elicited and analyzed. SRS is a formal report, which acts as a representation of software that enables the customers to review whether it (SRS) is according to their requirements. Also, it comprises user requirements for a system as well as detailed specifications of the system requirements.

The SRS is a specification for a specific software product, program, or set of applications that perform particular functions in a specific environment. It serves several goals depending on who is writing it. First, the SRS could be written by the client of a system. Second, the SRS could be written by a developer of the system. The two methods create entirely various situations and establish different purposes for the document altogether. The first case, SRS, is used to define the needs and expectation of the users. The second case, SRS, is written for various purposes and serves as a contract document between customer and developer.

**SRS must include the following sections:**

* **Purpose:** Definitions, system overview, and background.
* **Overall description:** Assumptions, constraints, business rules, and product vision.
* **Specific requirements:** System attributes, functional requirements, database requirements.

**Following are some features of SRS:**

* This document bridges gap between user and developer.
* Documents board imaginations into a structural layout.
* Used for measuring initial costs and efforts.
* Works as an agreement between communicating parties.

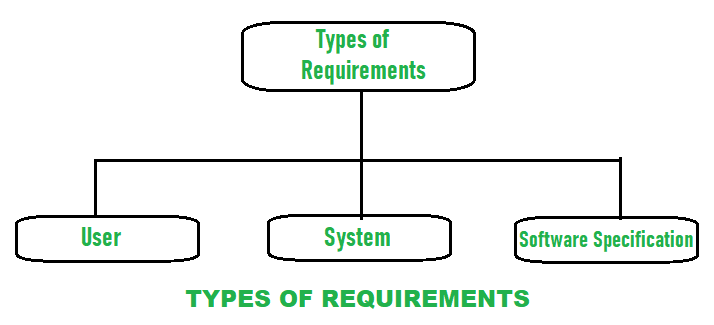
**Characteristics of good SRS:** Following are the features of a good SRS document:

* **Correctness:** User review is used to provide the accuracy of requirements stated in the SRS. SRS is said to be perfect if it covers all the needs that are truly expected from the system.
* **Completeness:** The SRS is complete if, and only if, it includes the following elements:
  + - * All essential requirements, whether relating to functionality, performance, design, constraints, attributes, or external interfaces.
      * Definition of their responses of the software to all realizable classes of input data in all available categories of situations.
      * Full labels and references to all figures, tables, and diagrams in the SRS and definitions of all terms and units of measure.
* **Consistency:** The SRS is consistent if, and only if, no subset of individual requirements described in its conflict.
* **Unambiguousness:** SRS is unambiguous when every fixed requirement has only one interpretation. This suggests that each element is uniquely interpreted. In case there is a method used with multiple definitions, the requirements report should determine the implications in the SRS so that it is clear and simple to understand.
* **Ranking for importance and stability:** The SRS is ranked for importance and stability if each requirement in it has an identifier to indicate either the significance or stability of that particular requirement. Typically, all requirements are not equally important. Some prerequisites may be essential, especially for life-critical applications, while others may be desirable. Each element should be identified to make these differences clear and explicit. Another way to rank requirements is to distinguish classes of items as essential, conditional, and optional.
* **Modifiability:** SRS should be made as modifiable as likely and should be capable of quickly obtain changes to the system to some extent. Modifications should be perfectly indexed and cross-referenced.
* **Verifiability:** SRS is correct when the specified requirements can be verified with a cost-effective system to check whether the final software meets those requirements. The requirements are verified with the help of reviews.
* **Traceability:** The SRS is traceable if the origin of each of the requirements is clear and if it facilitates the referencing of each condition in future development or enhancement documentation.
* **Design Independence:** There should be an option to select from multiple design alternatives for the final system. More specifically, the SRS should not contain any implementation details.
* **Testability:** An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.
* **Understandable by the customer:** An end user may be an expert in his/her explicit domain but might not be trained in computer science. Hence, the purpose of formal notations and symbols should be avoided too as much extent as possible. The language should be kept simple and clear.
* **The right level of abstraction:** If the SRS is written for the requirements stage, the details should be explained explicitly. Whereas, for a feasibility study, fewer analysis can be used. Hence, the level of abstraction modifies according to the objective of the SRS.

**Properties of a good SRS document:** The essential properties of a good SRS document are the following:

* **Concise:** The SRS report should be concise and at the same time, unambiguous, consistent, and complete. Verbose and irrelevant descriptions decrease readability and also increase error possibilities.
* **Structured:** It should be well-structured. A well-structured document is simple to understand and modify. In practice, the SRS document undergoes several revisions to cope up with the user requirements. Often, user requirements evolve over a period of time. Therefore, to make the modifications to the SRS document easy, it is vital to make the report well-structured.
* **Black-box view:** It should only define what the system should do and refrain from stating how to do these. This means that the SRS document should define the external behaviour of the system and not discuss the implementation issues. The SRS report should view the system to be developed as a black box and should define the externally visible behaviour of the system. For this reason, the SRS report is also known as the black-box specification of a system.

Requirement simply means a thing that is needed or wanted. Requirement engineering is process of defining requirement, establishing, documenting it in a proper manner, and to keep quality of requirements of a customer from a system, and the limitations under which it operates and is developed. It is first activity of software engineering. Requirements are something that is needed to satisfy whether by designing, product, or process of a software system. Requirements can be classified as:



* **User requirements:** User requirement simply means needs of users that should be fulfilled by software system. It is documented in a User Requirement Document (URD). Overall statements are generally written in natural language plus a description of the services system provides and its operational constraints. User requirement is good if it is clear and short, results in increasing overall quality, increases productivity, is traceable, etc.
* **System Requirements:** System requirement simply means needs of system to run smoothly and efficiently. It is a structured document that gives a detailed description of system functions, services, and operational constraints. It requires many hardware and software resources. If these hardware and software resources are not or less available, then it may result in system failure or causes problems during performance. Between client and contractor, it is written as a contract to define all requirements that are needed to be implemented to increases productivity.
* **Software specification:** It is a detailed description of software system requirements with the help of which designing and implementation can be done to develop software. For software developers, software specification is usually written that makes it easier for the developer to understand overall requirement of software.

**Two main types of requirements of a system:**

* **Functional requirements:** Functional requirements are mandatory which means it is compulsory and needed to be fulfilled. They generally describe and define features of end product of software system and simply focuses on what the end product does. These are the requirements that a system should accomplish or do like calculations, data manipulations, etc.

A system should provide a statement of service which describes how system reacts to inputs provided and should be clear and how a system reacts in a particular situation. Functional requirements are type of requirements that depends upon type of software as different software has a different functional requirement, system on which software is used as it heavily affects functions of software and users to fulfill their requirements. Functional requirement of users is high-level abstract statements. it generally describes of what system should whenever required but system functions should be described in detail by functional system requirements.

* **Non-Functional Requirements:** Non-functional requirements are not mandatory which means that they are not compulsory to be fulfilled. The non-functional requirements define system properties and system performance. Different properties of a system are there which can be Reliability, response time, maintainability, availability, storage requirements. It simply focuses on how the end product works and it is not very easy and hard to find out non-functional requirements and captured as a quality attribute. Testing includes performance, stress, security testing, etc. Non-functional requirements are more disapproved and if the non-functional requirements are not fulfilled then complete system is of no use.

**Example:** Product requirement, organizational requirement, external requirement, capacity or storage requirement, resources requirement, overall performance requirement time etc.

* 1. **Software Functional Specification**

A Functional Requirement (FR) is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behaviour, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements in Software Engineering are also called Functional Specification.

In software engineering and systems engineering, a Functional Requirement can range from the high-level abstract statement of the sender's necessity to detailed mathematical functional requirement specifications. Functional software requirements help you to capture the intended behaviour of the system. These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So, it’s important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behaviour under specific conditions.

**For instance:** A search feature allows a user to hunt among various invoices if they want to credit an issued invoice.

**Following are some features of FRS:**

* This document elaborates on the functions to the user.
* Helps user to understand interactions of software and its behavior.
* It helps designing testing parameters for software.
* Provides better aid for wire-frames and conceptual diagrams.

**Functional Requirements of a system should include the following things:**

* Details of operations conducted in every screen.
* Data handling logic should be entered into the system.
* It should have descriptions of system reports or other outputs.
* Complete information about the workflows performed by the system.
* It should clearly define who will be allowed to create/modify/delete the data in the system.
* How the system will fulfill applicable regulatory and compliance needs should be captured in the functional document.

**Benefits of Functional Requirement:** Here, are the pros/advantages of creating a typical functional requirement document:

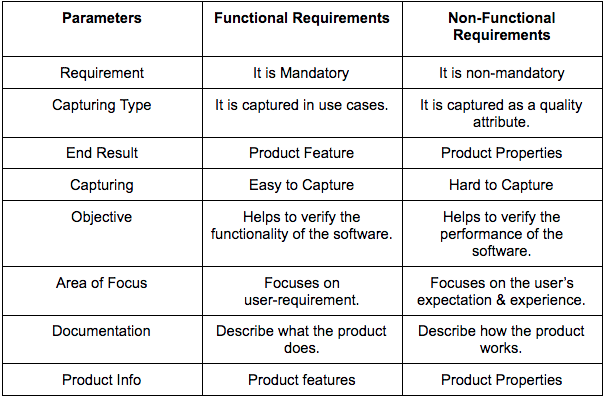
* Helps you to check whether the application is providing all the functionalities that were mentioned in the functional requirement of that application.
* A functional requirement document helps you to define the functionality of a system or one of its subsystems.
* Functional requirements along with requirement analysis help identify missing requirements. They help clearly define the expected system service and behavior.
* Errors caught in the Functional requirement gathering stage are the cheapest to fix.
* Support user goals, tasks, or activities

**Best practice of Functional Requirement:** Important best practice for developing functional requirement document is as follows:

* Do not combine two requirements into one. Keep the requirements granular.
* You should make each requirement as complete and accurate as possible.
* The document should draft all the technical requirements.
* Map all requirements to the objectives and principles which contributes to successful software delivery.
* Elicit requirements using interviews, workshops and casual communications.
* If there is any known, verified constraint which materially affects a requirement then it is a critical state that should be documented.
* It is necessary that you document all the assumption in the document.

**There are certain differences between functional and non-functional requirements.**

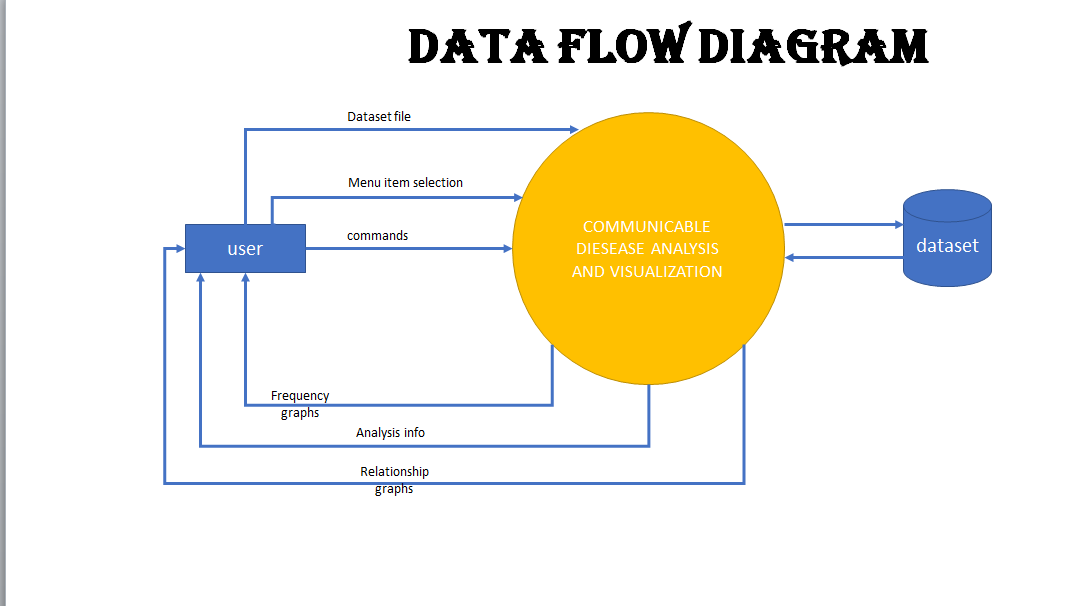
**These differences are as follows:**

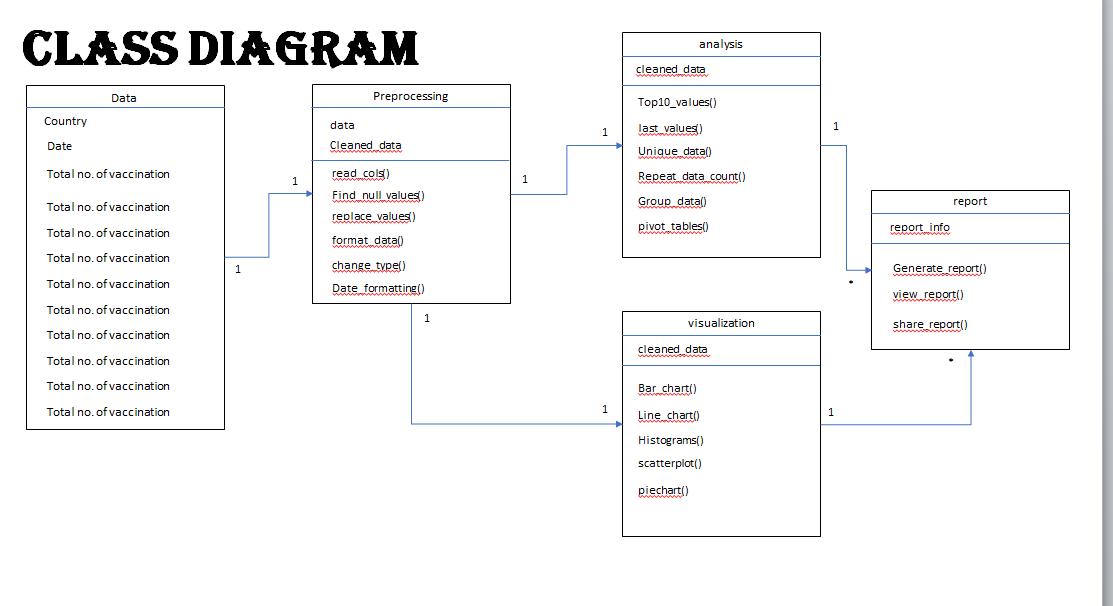
****

* 1. **Data Flow Diagram**

The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modelling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

* **Components of DFD:** The Data Flow Diagram has 4 components:
* **Process:** Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence.
* **Data Flow:** Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represents material along with information that is being moved. Material shifts are modelled in systems that are not merely informative. A given flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi-directional.
* **Warehouse:** The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The data warehouse can be viewed independent of its implementation. When the data flow from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updations.
* **Terminator:** The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity. Modelled systems also communicate with terminator.
* **0 Level DFD:** It is also known as a context diagram. It’s designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

****

****

1. **SYSTEM TESTING**

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing

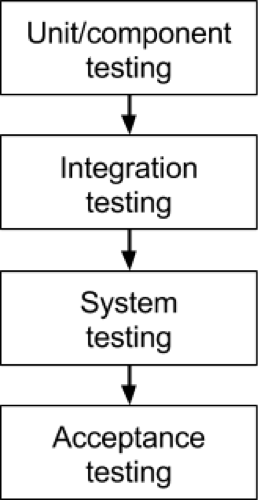
detects defects within both the integrated units and the whole system. The result of system testing is the observed behaviour of a component or a system when it is tested.

System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behaviour of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS). System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing.

System Testing includes testing of a fully integrated software system. Generally, a computer system is made with the integration of software (any software is only a single element of a computer system). The software is developed in units and then interfaced with other software and hardware to create a complete computer system. In other words, a computer system consists of a group of software to perform the various tasks, but only software cannot perform the task; for that software must be interfaced with compatible hardware. System testing is a series of different type of tests with the purpose to exercise and examine the full working of an integrated software computer system against requirements.

**Hierarchy of Testing Levels:** There are generally four recognized levels of testing:

1. Unit/Component Testing
2. Integration testing
3. System testing
4. Acceptance testing



1. **Unit/component testing:** The most basic type of testing is unit, or component, testing. Unit testing aims to verify each part of the software by isolating it and then perform tests to demonstrate that each individual component is correct in terms of fulfilling requirements and the desired functionality.

This type of testing is performed at the earliest stages of the development process, and in many cases, it is executed by the developers themselves before handing the software over to the testing team. The advantage of detecting any errors in the software early in the day is that by doing so the team minimizes software development risks, as well as time and money wasted in having to go back and undo fundamental problems in the program once it is nearly completed.

1. **Integration testing:** Integration testing aims to test different parts of the system in combination in order to assess if they work correctly together. By testing the units in groups, any faults in the way they interact together can be identified. There are many ways to test how different components of the system function at their interface; testers can adopt either a bottom-up or a top-down integration method.

In bottom-up integration testing, testing builds on the results of unit testing by testing higher-level combination of units (called modules) in successively more complex scenarios. It is recommended that testers start with this approach first, before applying the top-down approach which tests higher-level modules first and studies simpler ones later.

1. **System testing:** The next level of testing is system testing. As the name implies, all the components of the software are tested as a whole in order to ensure that the overall product meets the requirements specified. System testing is a very important step as the software is almost ready to ship and it can be tested in an environment which is very close to that which the user will experience once it is deployed.

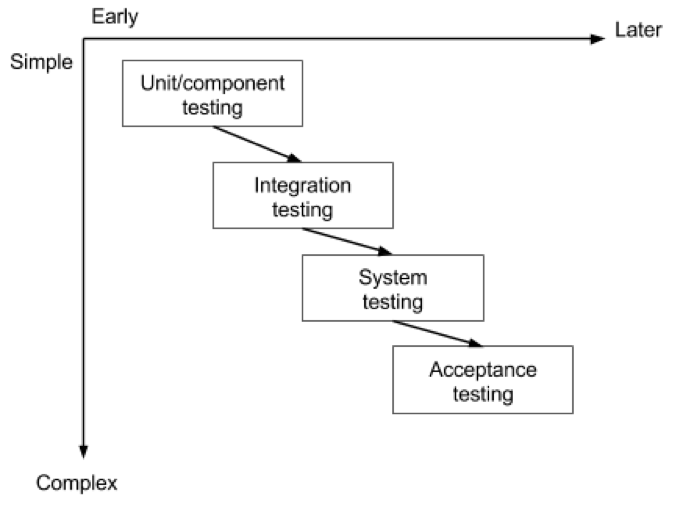
System testing enables testers to ensure that the product meets business requirements, as well as determine that it runs smoothly within its operating environment. This type of testing is typically performed by a specialized testing team.

1. **Acceptance testing:** Finally, acceptance testing is the level in the software testing process where a product is given the green light or not. The aim of this type of testing is to evaluate whether the system complies with the end-user requirements and if it is ready for deployment. The testing team will utilize a variety of methods, such as pre-written scenarios and test cases to test the software and use the results obtained from these tools to find ways in which the system can be improved.

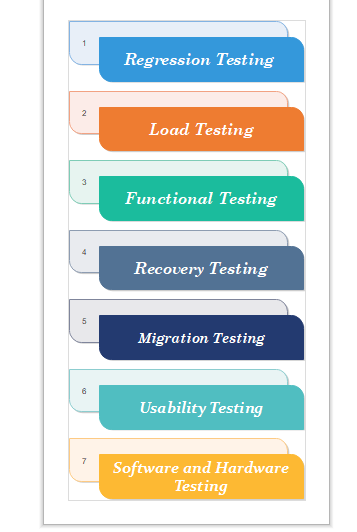
The scope of acceptance testing ranges from simply finding spelling mistakes and cosmetic errors, to uncovering bugs that could cause a major error in the application. By performing acceptance tests, the testing team can find out how the product will perform when it is installed on the user’s system. There are also various legal and contractual reasons why acceptance testing has to be carried out.

**The testing sequence:** These four types of testing cannot be applied haphazardly during development. There is a logical sequence that should be adhered to in order to minimize the risk of bugs cropping up just before the launch date. By progressively testing the simpler components of the system and moving on the bigger, more complex groupings, the testers can rest assured they are thoroughly examining the software in the most efficient way possible.

The four levels of testing shouldn’t only be seen as a hierarchy that extends from simple to complex, but also as a sequence that spans the whole development process from the early to the later stages.



**Types of System Testing:** System testing is divided into more than 50 types, but software testing companies typically uses some of them. These are listed below:



* **Regression Testing:** Regression testing is performed under system testing to confirm and identify that if there's any defect in the system due to modification in any other part of the system. It makes sure, any changes done during the development process have not introduced a new defect and also gives assurance; old defects will not exist on the addition of new software over the time.
* **Load Testing:** Load testing is performed under system testing to clarify whether the system can work under real-time loads or not.
* **Functional Testing:** Functional testing of a system is performed to find if there's any missing function in the system. Tester makes a list of vital functions that should be in the system and can be added during functional testing and should improve quality of the system.
* **Recovery Testing:** Recovery testing of a system is performed under system testing to confirm reliability, trustworthiness, accountability of the system and all are lying on recouping skills of the system. It should be able to recover from all the possible system crashes successfully. In this testing, we will test the application to check how well it recovers from the crashes or disasters.
* **Migration Testing:** Migration testing is performed to ensure that if the system needs to be modified in new infrastructure so it should be modified without any issue.
* **Usability Testing:** The purpose of this testing to make sure that the system is well familiar with the user and it meets its objective for what it supposed to do.
* **Software and Hardware Testing:** This testing of the system intends to check hardware and software compatibility. The hardware configuration must be compatible with the software to run it without any issue. Compatibility provides flexibility by providing interactions between hardware and software.

1. **SYSTEM IMPLEMENTATION**

The project can be implemented on a system having following minimum requirements:

Minimum Hardware Requirement:

* Intel i3 processor or AMD Ryzen 3 processor
* 4GB RAM
* 500 GB Hard Drive or 128 GB SSD
* Internet Connection

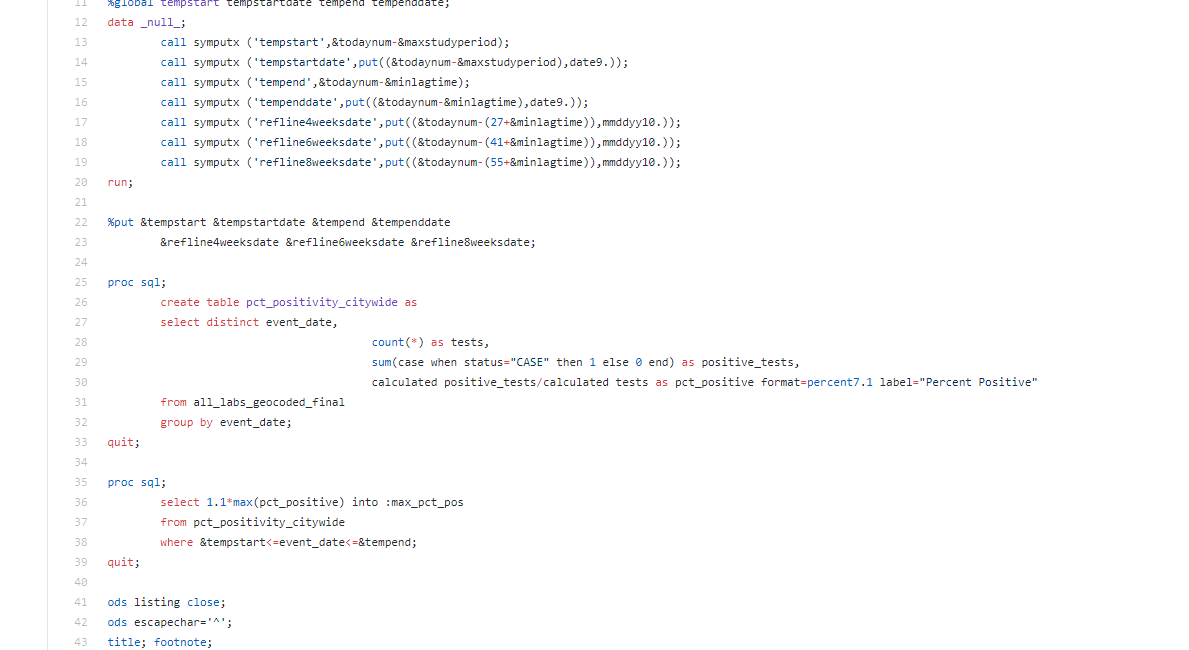
Software Requirement:

* Python 3.x.x or Later
* Streamlit (Latest)
* VS Code
* PIP(Latest)

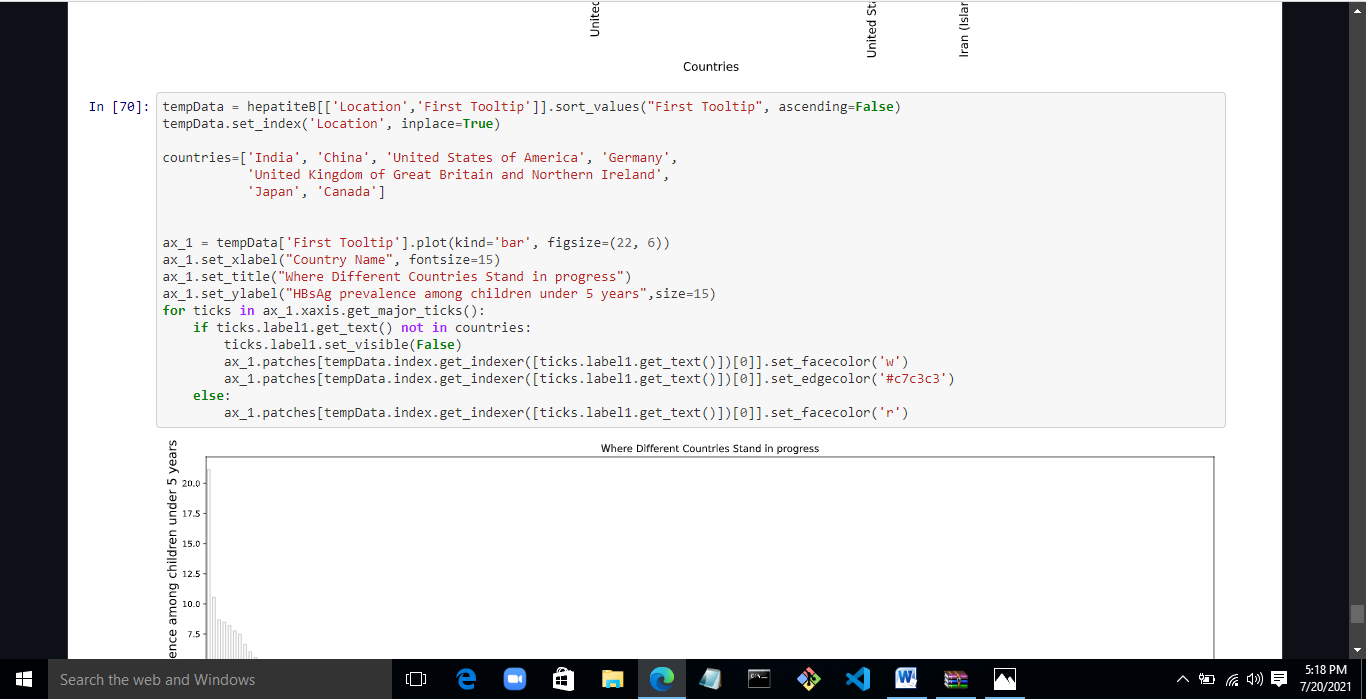
1. **SYSTEM INPUT AND OUTPUT SCREENS**

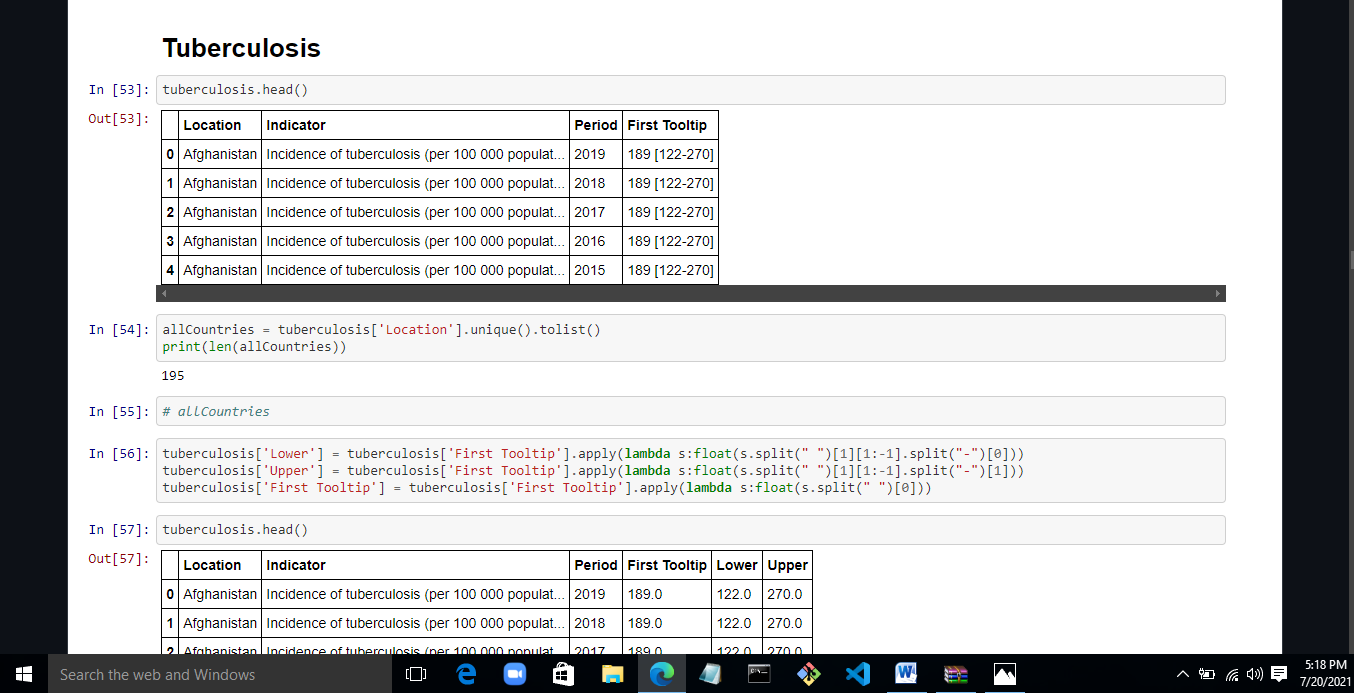
**INPUT SCREENS:**

* **COVID\_ANALYSIS\_city-wide\_percent\_positivity\_graphs.sas**

****

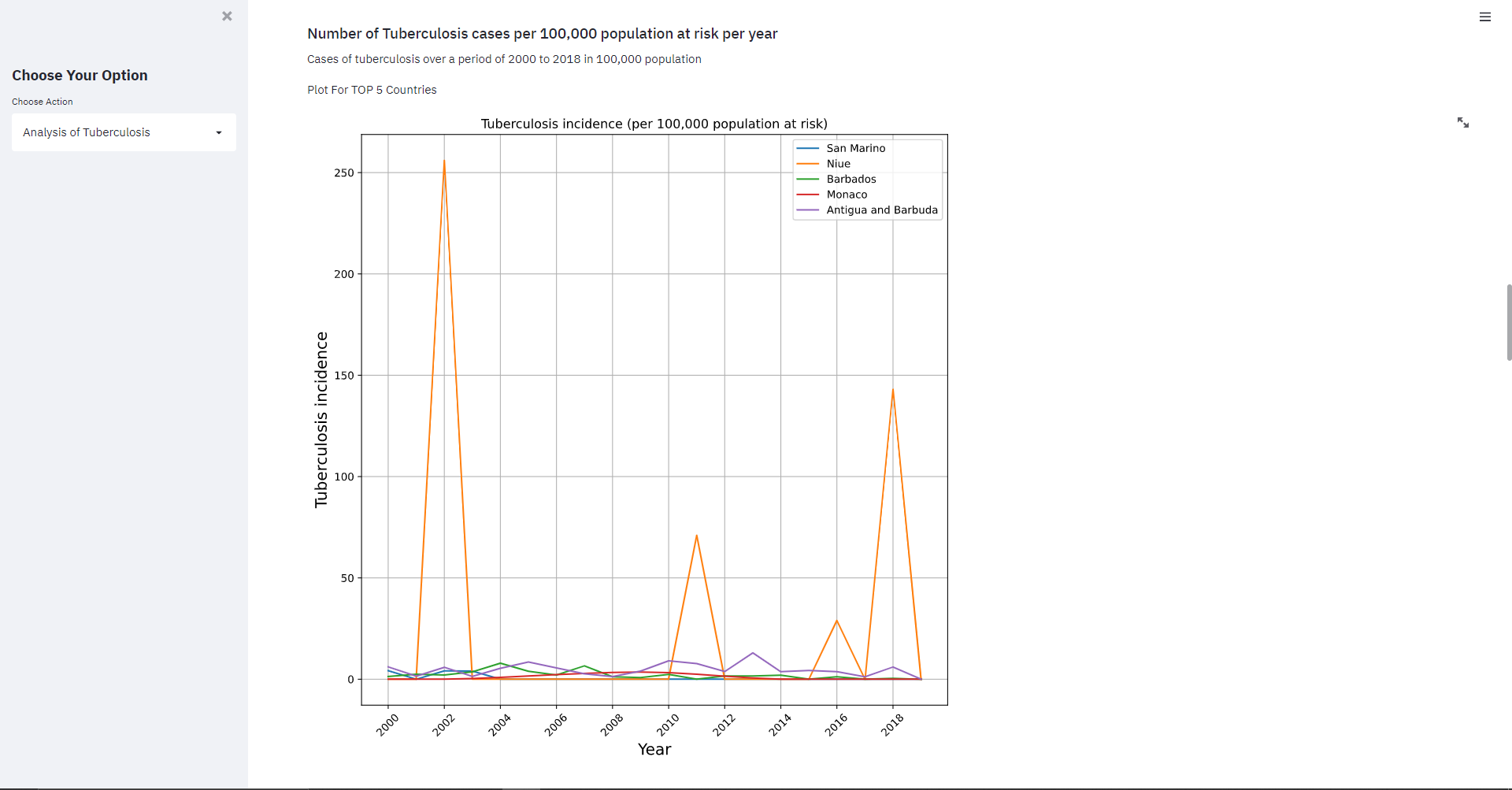
****

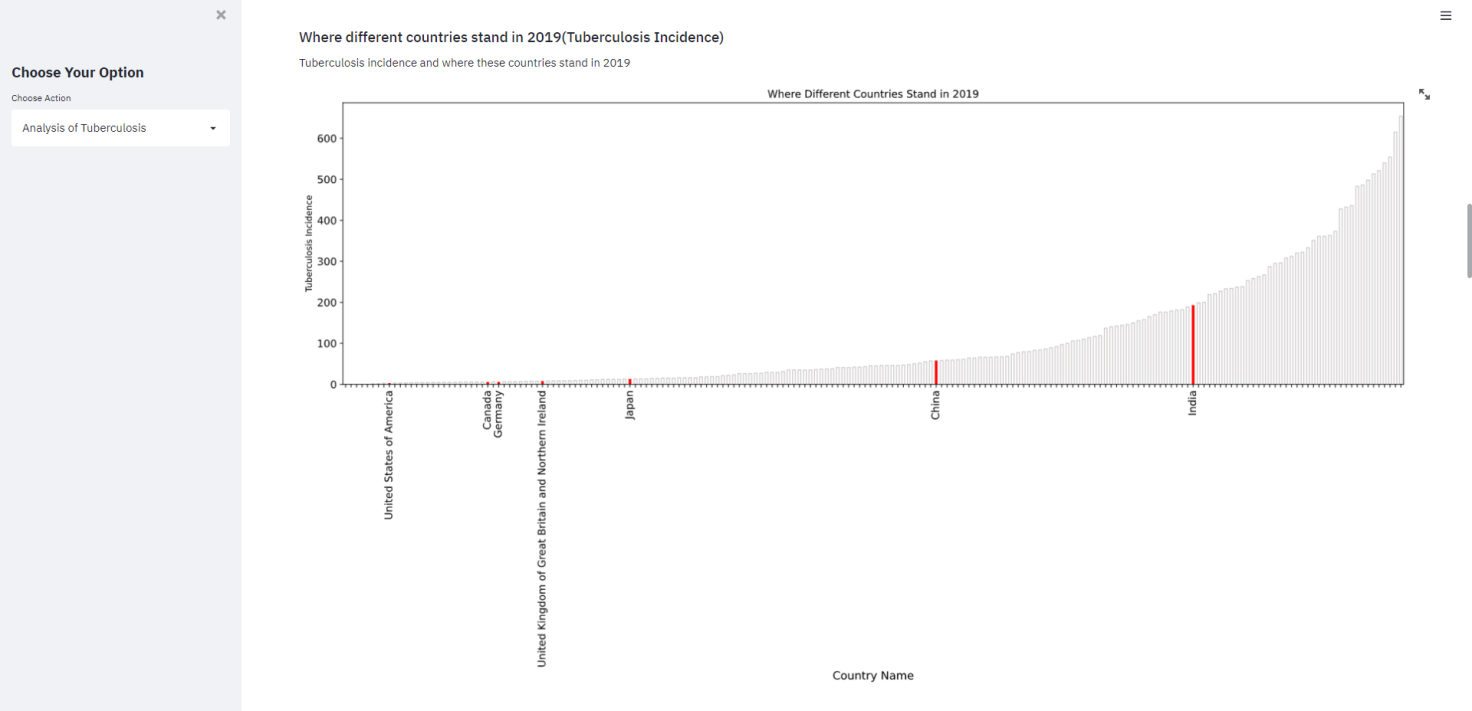


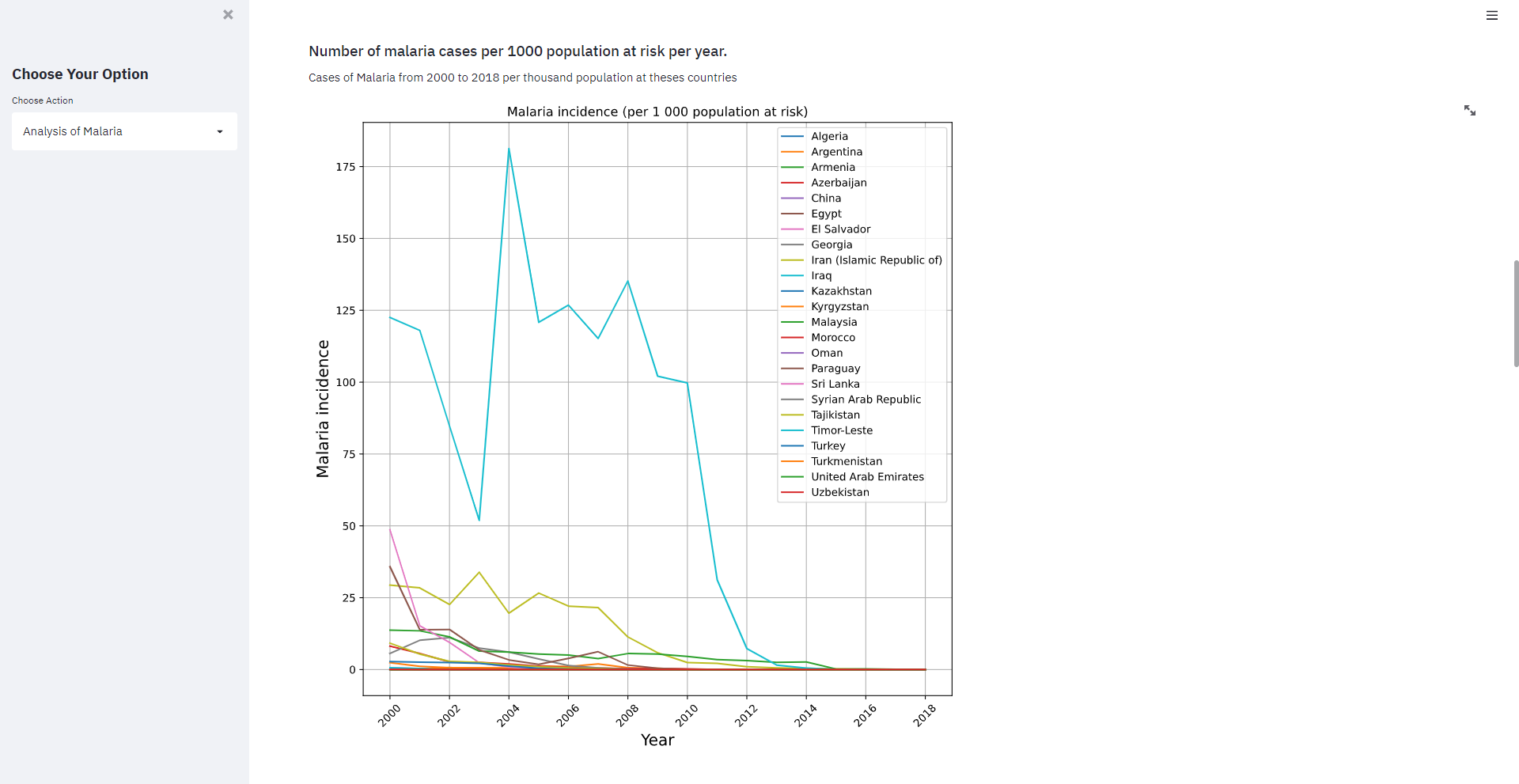


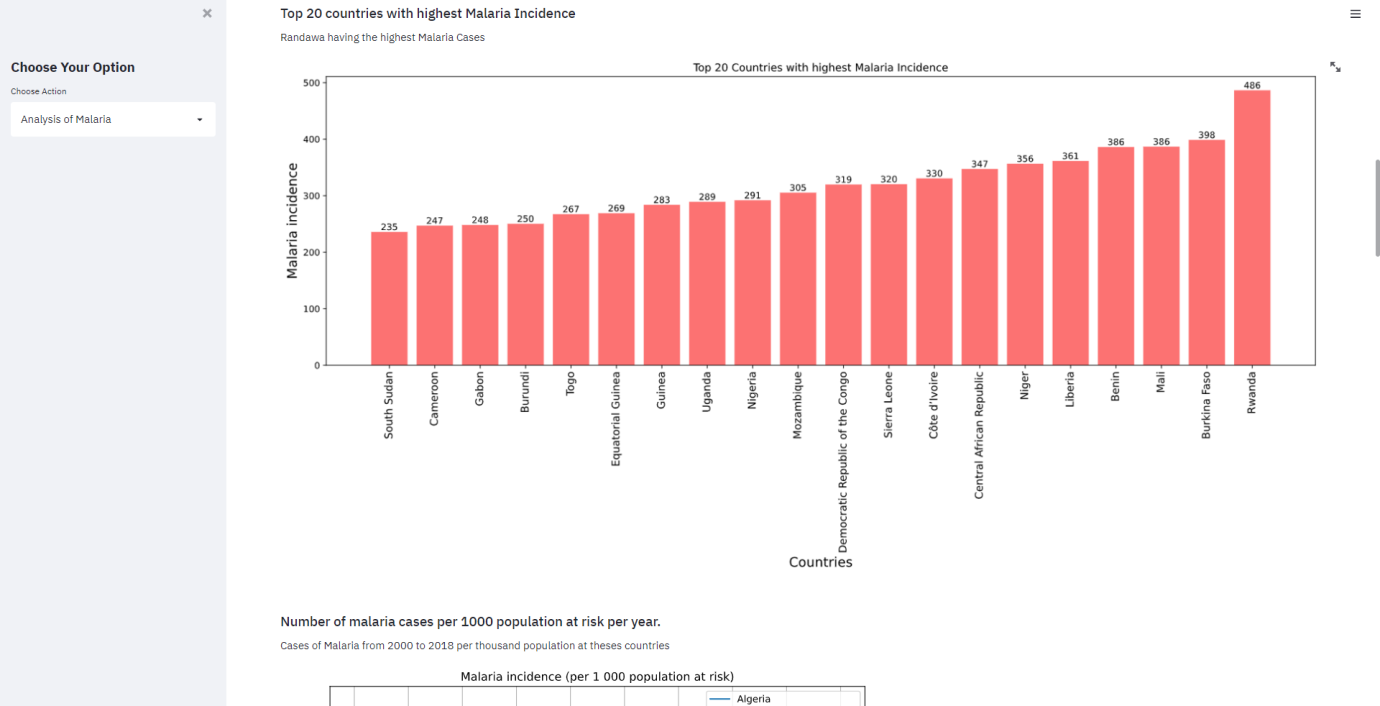


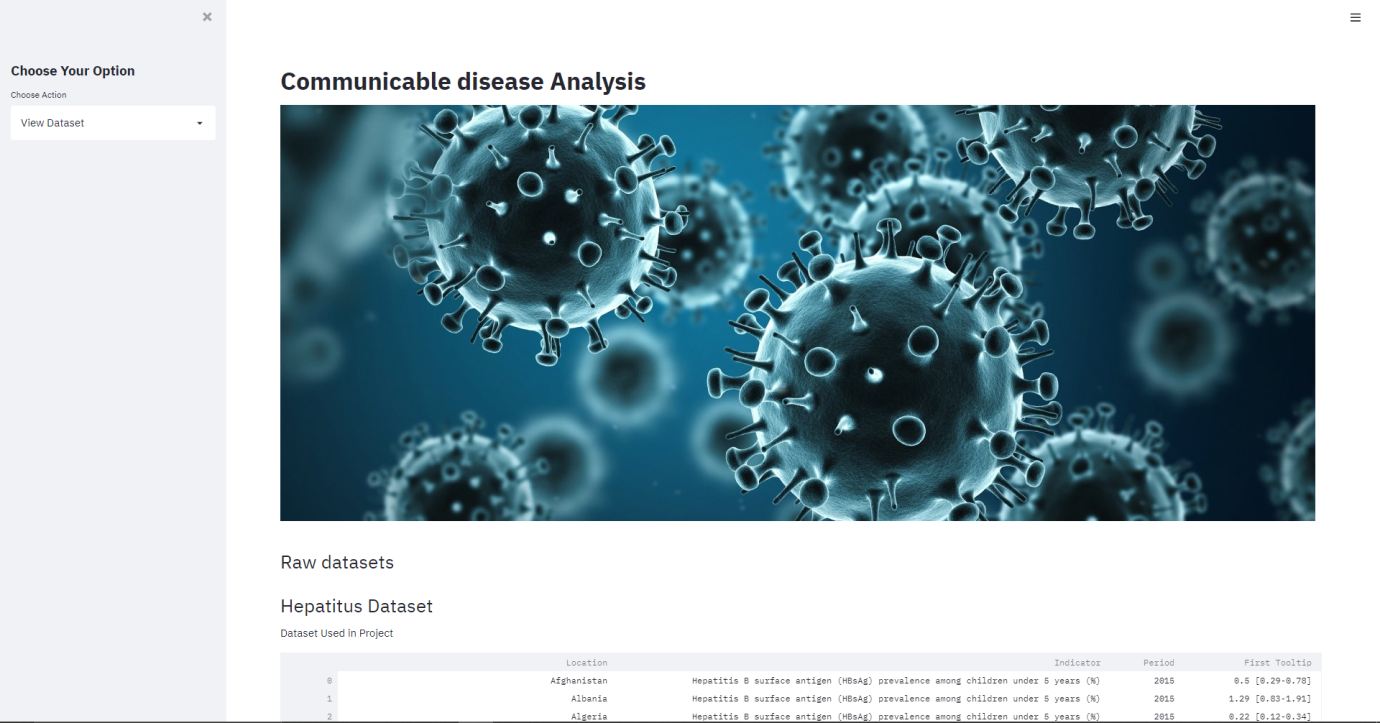
**OUTPUT SCREENS:**

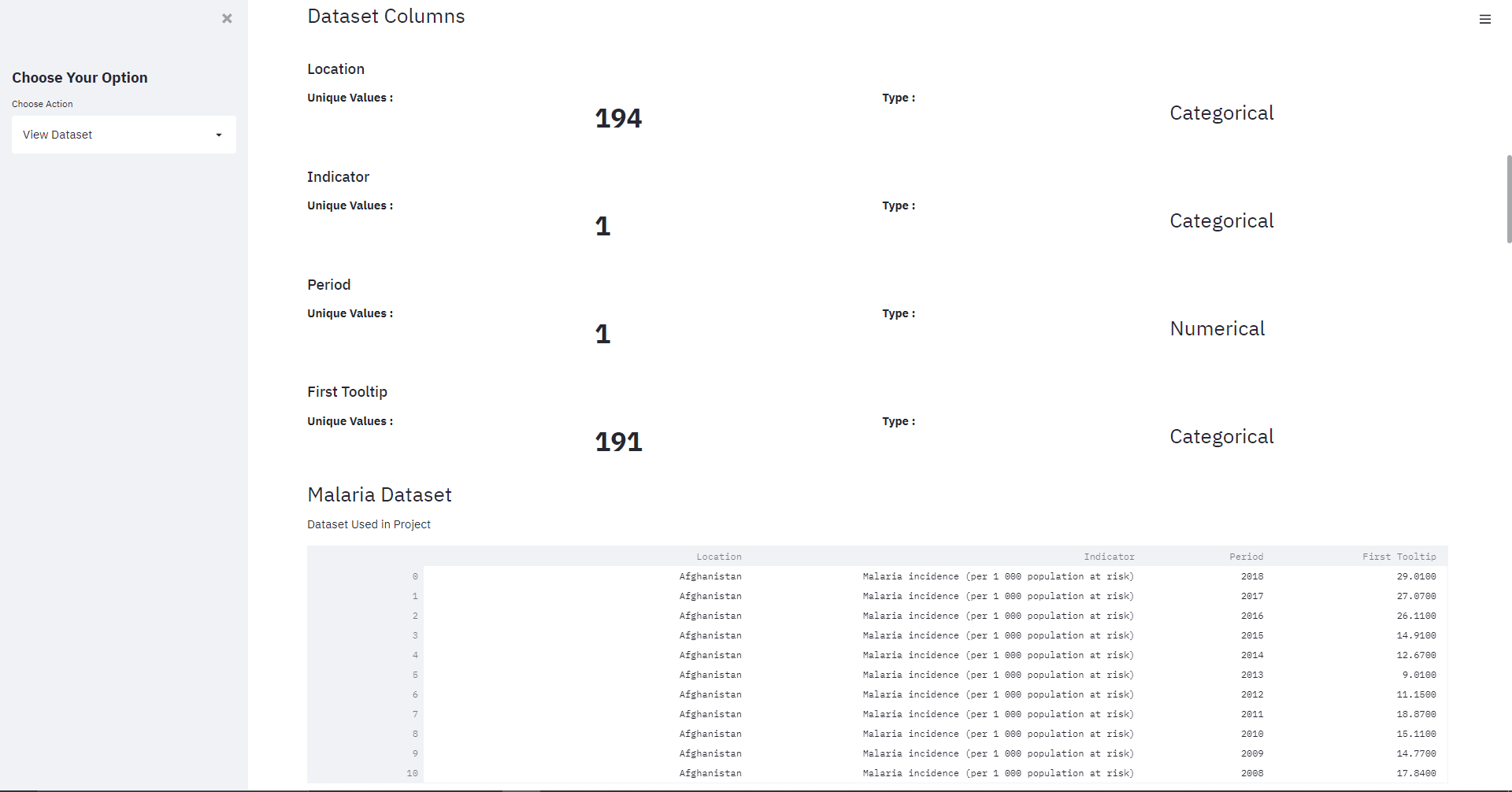


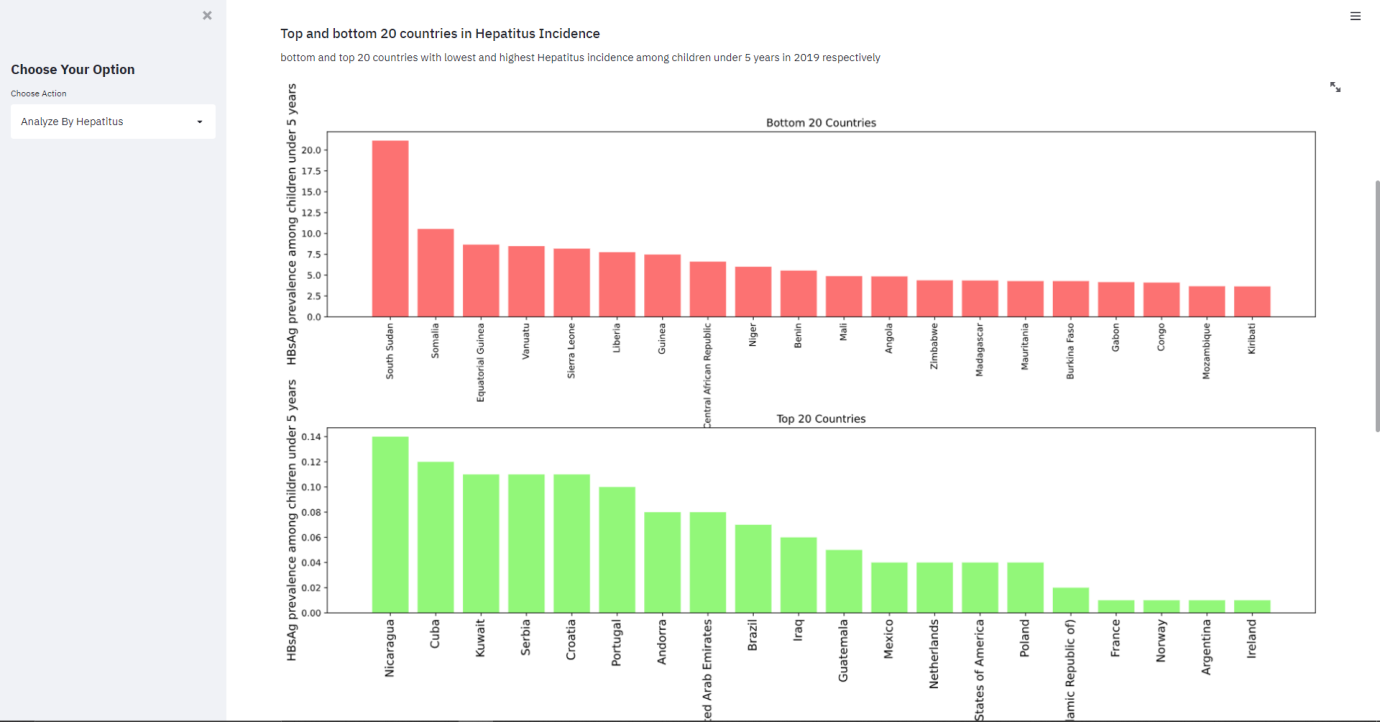


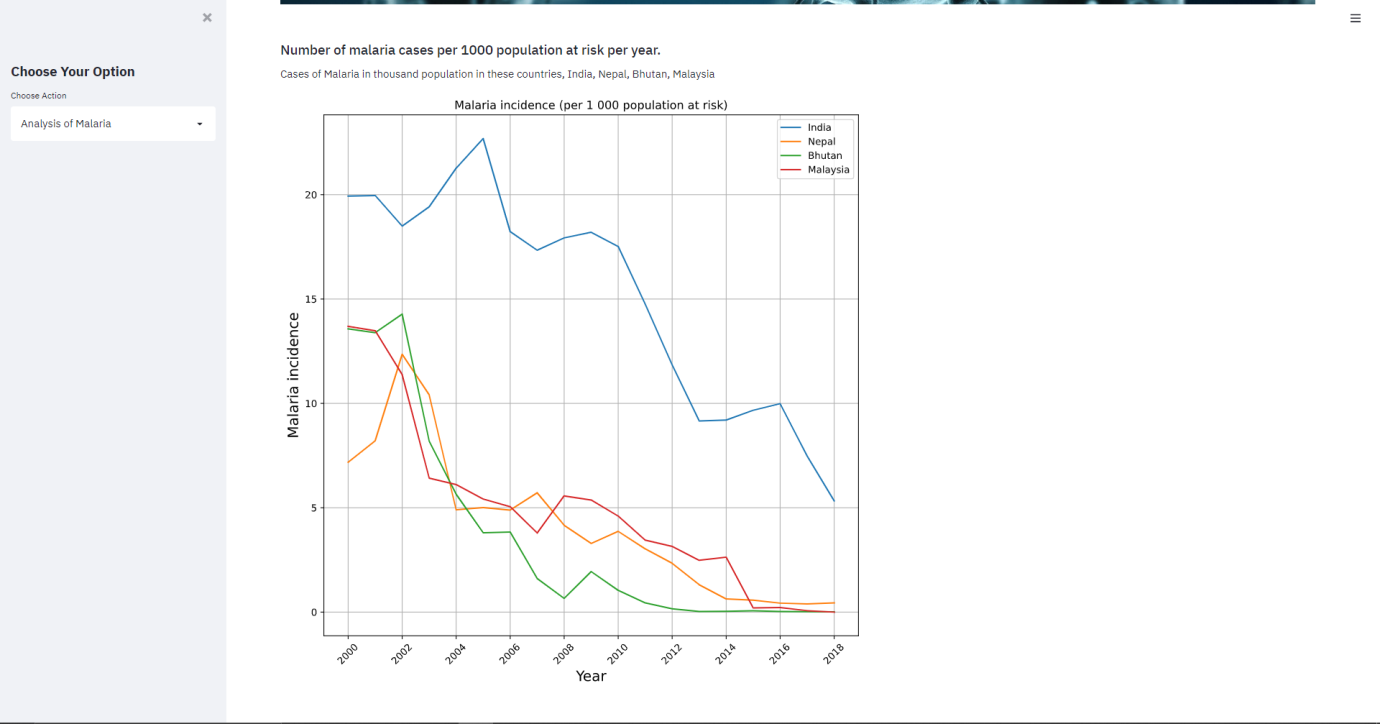












**SOME CODE LINES:**

**import** **os**

**for** dirname, \_, filenames **in** os.walk('/kaggle/input'):

**for** filename **in** filenames:

print(os.path.join(dirname, filename))

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

**import** **seaborn** **as** **sns**

**import** **warnings**

warnings.filterwarnings('ignore')

malaria = pd.read\_csv("datasets/incedenceOfMalaria.csv")

tuberculosis = pd.read\_csv("datasets/incedenceOfTuberculosis.csv")

hepatiteB = pd.read\_csv("datasets/hepatitusBsurfaceAntigen.csv")

HIV = pd.read\_csv("datasets/newHivInfections.csv")

NTDs = pd.read\_csv("datasets/interventionAgianstNTDs.csv")

**def** compareCountries(countries):

fig = plt.figure(figsize = (10,10))

ax1 = plt.subplot2grid((1,1),(0,0))

**for** country **in** countries:

**if** country **not** **in** allCountries:

print(f"Country **{**country**}** DO NOT EXIST ")

**return**

tempData = malaria[malaria['Location'] == country]

tempData['Period'] = pd.to\_datetime(tempData['Period'], format = '%Y')

ax1.plot(tempData['Period'], tempData['First Tooltip'], label=country)

**for** tick **in** ax1.get\_xticklabels():

tick.set\_rotation(45)

plt.grid()

plt.legend()

plt.xlabel("Year",size=15)

plt.ylabel("Malaria incidence", size=15)

plt.title("Malaria incidence (per 1 000 population at risk)")

plt.show()

tempData = malaria[malaria['Period'] == 2018]

tempData.sort\_values('First Tooltip', inplace=**True**)

tempDataBot = tempData[-20:]

fig = plt.figure(figsize=(20,6))

ax = plt.subplot2grid((1,1),(0,0))

rects = ax.bar(tempDataBot['Location'], tempDataBot['First Tooltip'], color = "#fc7272")

**for** tick **in** ax.get\_xticklabels():

tick.set\_rotation(90)

autoLabel(rects)

plt.xticks(size=12)

plt.xlabel("Countries", size=15)

plt.ylabel("Malaria incidence",size=15)

plt.title("Top 20 Countries with highest Malaria Incidence")

plt.show()

tempData = malaria[malaria['Period'] == 2018]

tempData = tempData[tempData['First Tooltip'] == 0.0]

bestCountries=tempData['Location'].values.tolist()

tempData = malaria[malaria.Location.isin(bestCountries)]

tempData = tempData[tempData['Period'] == 2000]

tempData.sort\_values('First Tooltip', ascending=**False**, inplace=**True**)

fig = plt.figure(figsize=(20,6))

ax = plt.subplot2grid((1,1),(0,0))

rects = ax.bar(tempData['Location'], tempData['First Tooltip'], color = "#92f779")

**for** tick **in** ax.get\_xticklabels():

tick.set\_rotation(90)

autoLabel(rects)

plt.xticks(size=12)

plt.xlabel("Countries",size=15)

plt.ylabel("Progress", size=15)

plt.title("Best Progress over year")

plt.show()

tempData2018 = malaria[malaria['Period'] == 2018]

tempData2000 = malaria[malaria['Period'] == 2000]

tempData2018.reset\_index(inplace=**True**)

tempData2000.reset\_index(inplace=**True**)

tempData = tempData2018.join(tempData2000, lsuffix='\_18', rsuffix='\_00')

tempData['progress'] = tempData['First Tooltip\_00'] - tempData['First Tooltip\_18']

tempData = tempData[['Location\_18', 'progress']].sort\_values('progress', ascending=**False**)

tempDataTop = tempData[:20]

tempDataBot = tempData[-20:]

fig = plt.figure(figsize=(20,10))

ax1 = plt.subplot2grid((1,1), (0,0), rowspan=1, colspan=1)

rects1 = ax1.bar(tempDataTop['Location\_18'], tempDataTop['progress'], color = '#92f779')

rects1 = ax1.bar(tempDataBot['Location\_18'], tempDataBot['progress'], color = '#fc7272')

**for** tick **in** ax1.get\_xticklabels():

tick.set\_rotation(90)

plt.xticks(size=12)

plt.xlabel("Countries",size=15)

plt.ylabel("Progress",size=15)

plt.title("Progress of Different Countries")

plt.show()

**def** compareCountries(countries):

fig = plt.figure(figsize = (10,10))

ax1 = plt.subplot2grid((1,1),(0,0))

**for** country **in** countries:

**if** country **not** **in** allCountries:

print(f"Country **{**country**}** DO NOT EXIST ")

**return**

tempData = tuberculosis[tuberculosis['Location'] == country]

tempData['Period'] = pd.to\_datetime(tempData['Period'], format = '%Y')

ax1.plot(tempData['Period'], tempData['First Tooltip'], label=country)

**for** tick **in** ax1.get\_xticklabels():

tick.set\_rotation(45)

plt.grid()

plt.legend()

plt.xlabel("Year",size=15)

plt.ylabel("Tuberculosis incidence", size=15)

plt.title("Tuberculosis incidence (per 100,000 population at risk)")

plt.show()

tempData = tuberculosis[tuberculosis['Period'] == 2019]

tempData.sort\_values('First Tooltip', inplace=**True**)

tempDataBot = tempData[-20:]

tempDataTop = tempData[:20]

fig = plt.figure(figsize=(20,10))

ax1 = plt.subplot2grid((2,1),(0,0))

rects1 = ax1.bar(tempDataTop['Location'], tempDataTop['First Tooltip'], color = "#92f779")

**for** tick **in** ax1.get\_xticklabels():

tick.set\_rotation(90)

ax1.set\_ylabel("Tuberculosis incidence",size=15)

ax1.set\_title("Bottom 20 Countries with lowest Tuberculosis Incidence")

ax2 = plt.subplot2grid((2,1),(1,0))

rects2 = ax2.bar(tempDataBot['Location'], tempDataBot['First Tooltip'], color = "#fc7272")

**for** tick **in** ax2.get\_xticklabels():

tick.set\_rotation(90)

ax2.set\_xlabel("Countries", size=15)

ax2.set\_ylabel("Tuberculosis incidence",size=15)

ax2.set\_title("Top 20 Countries with highest Tuberculosis Incidence")

plt.xticks(size=12)

plt.subplots\_adjust(hspace = 0.6)

tempData = tuberculosis[tuberculosis['Period'] == 2019]

tempData.sort\_values('First Tooltip', inplace=**True**)

tempData.reset\_index(inplace=**True**)

tempData.set\_index('Location',drop=**True**,inplace=**True**)

countries=['India', 'China', 'United States of America', 'Germany',

'United Kingdom of Great Britain and Northern Ireland',

'Japan', 'Canada']

ax\_1 = tempData['First Tooltip'].plot(kind='bar', title ="graph", figsize=(22, 6), fontsize=12)

ax\_1.set\_xlabel("Country Name", fontsize=13)

ax\_1.set\_title("Where Different Countries Stand in 2019")

ax\_1.set\_ylabel("Tuberculosis Incidence")

**for** ticks **in** ax\_1.xaxis.get\_major\_ticks():

**if** ticks.label1.get\_text() **not** **in** countries:

ticks.label1.set\_visible(**False**)

ax\_1.patches[tempData.index.get\_indexer([ticks.label1.get\_text()])[0]].set\_facecolor('w')

ax\_1.patches[tempData.index.get\_indexer([ticks.label1.get\_text()])[0]].set\_edgecolor('#c7c3c3')

**else**:

ax\_1.patches[tempData.index.get\_indexer([ticks.label1.get\_text()])[0]].set\_facecolor('r')

1. **SCOPE OF PROJECT, LIMITATIONS AND CONCLUSION**

**Scope of the Project:**

As the momentum to scale up the global response to communicable diseases increases, public health practitioners need to constantly review their performance in detecting and responding to communicable diseases. At the same time, they should account for the planned activities, policies and resources to a variety of stakeholders. The staff working at different levels of surveillance need to report accurate data in a timely manner to the next higher level to ensure timely and effective responses to contain communicable disease outbreaks. They may be required to report on progress to partners and donors, but most importantly, surveillance information should be used locally to address and resolve problems related to control of communicable diseases and to strengthen evolving programs. Monitoring and evaluation are keys to establishing and maintaining effective and efficient surveillance and response systems.

This project will help in data analyzing very efficiently.

**Limitations:**

There are certain limitations of this project. These are as follows:

* Dataset is always huge in size.
* Dataset must be prepared by the college itself.
* This project has a limitation like if someone is very naive and they are not able to understand the bar charts, histograms etc. then for those this project of no use.

**CONCLUSION**

The system developed, had as its main objective to provide a means of surveillance of notifiable diseases in world, in real time. This goal was achieved, and it is now possible to use the Labs to obtain real-time information about the current state of laboratory notifications of notifiable diseases in the world. In terms of the requirements, were all achieved, since it is possible to view in real-time information about the incidence of notifiable diseases by several factors such as geography, age groups and gender. The system also allows you to see trends and developments over time.

As the volume and complexity of infectious disease data increases, public health professionals must synthesize highly disparate data to facilitate communication with the public and inform decisions regarding measures to protect the public's health. Our review identified several themes: consideration of users' needs, preferences, and computer literacy; integration of tools into routine workflow; complications associated with understanding and use of visualizations; and the role of user trust and organizational support in the adoption of these tools. Interoperability also emerged as a prominent theme, highlighting challenges associated with the increasingly collaborative and interdisciplinary nature of infectious disease control and prevention. Future work should address methods for representing uncertainty and missing data to avoid misleading users as well as strategies to minimize cognitive overload.

**References**

1. <https://link.springer.com/article/10.1007/s41060-018-0102-5>
2. <https://its.ucsc.edu/project-management/docs/brown-bag-docs/project-roles-and-resp-for-presentation.pdf>
3. <https://www.kaggle.com/benroshan/factors-affecting-campus-placement>
4. <https://www.python.org/>
5. <https://numpy.org/>
6. <https://pandas.pydata.org/>
7. <https://matplotlib.org/>
8. <https://rpubs.com/ngjunhong/exercise1>
9. <https://www.atlantis-press.com/article/25853260.pdf>