A close up of a sign

Description automatically generated

AUTHOR’S NAME:

**MUNEEB UR REHMAN (160655)**

**ZAEEM DASTAGHIR (160335)**

**IN DOC DATA ANALYSIS**

Bachelor of Science in Computer Science

SUPERVISOR NAME: MUHAMMAD SHOAIB MALIK

CO-SUPERVISOR NAME: IMRAN IHSAN

DEPARTMENT OF COMPUTER SCIENCE

AIR UNIVERSITY, ISLAMABAD

May 2020.

© Muneeb, Zaeem, 2020

**Certificate**

We accept the work contained in this report titled “In Doc Data Analysis”, written by Muneeb ur Rehman And Zaeem Dastaghir as a confirmation to required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

**Approved by:**

Supervisor: Muhammad Shoaib Malik

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

External Examiner:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Project Coordinator: Imran Ihsan

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Head of the Department: Dr. Mehdi Hassan

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

June 2020.

i

ii

iii

“We think someone else, someone smarter than us, someone more capable, someone with more resources will solve that problem.

But there isn’t anyone else.”

Regina Dugan

**Table of content**

Table of Contents

[**Abstract:** 12](#_Toc43402111)

[**Acknowledgment:** 14](#_Toc43402112)

[**Chapter 1: Introduction** 16](#_Toc43402113)

[**1.1** **Background:** 16](#_Toc43402114)

[**1.2** **Objective:** 17](#_Toc43402115)

[**1.3 Scope:** 17](#_Toc43402116)

[**1.4 Problem Statement:** 18](#_Toc43402117)

[**Chapter 2: Literature Review** 20](#_Toc43402118)

[**2.1 Related Projects:** 20](#_Toc43402119)

[**2.1.1 Janiform IntraDocument Analytics for Reproducible Research** 20](#_Toc43402120)

[**Chapter 3: Requirement Specification** 22](#_Toc43402121)

[**3.1 Existing System:** 22](#_Toc43402122)

[**3.2 PROPOSED SYSTEM:** 22](#_Toc43402123)

[**Chapter 4: Design** 25](#_Toc43402124)

[**4.1 SYSTEM ARCHITECTURE** 25](#_Toc43402125)

[**4.2 DESIGN CONSTRAINTS** 25](#_Toc43402126)

[**4.3 DESIGN METHODOLOGY** 26](#_Toc43402127)

[**4.4 GUI DESIGN** 27](#_Toc43402128)

[**Chapter 5: System Implementation** 30](#_Toc43402129)

[**5.1 System Architecture** 30](#_Toc43402130)

[**5.1.1 Architecture Description:** 30](#_Toc43402131)

[**5.1.2 Project Flow** 31](#_Toc43402132)

[**5.1.3 Tools and Technology:** 32](#_Toc43402133)

[**Chapter 6: System Testing Evaluation** 34](#_Toc43402134)

[**6.1 GUI TESTING** 34](#_Toc43402135)

[**6.2 USABILITY TESTING** 34](#_Toc43402136)

[**6.3 COMPATIBILITY TESTING** 34](#_Toc43402137)

[**6.4 LOAD TESTING** 35](#_Toc43402138)

[**6.5 INSTALLATION TESTING** 35](#_Toc43402139)

[**6.6 EXCEPTION HANDLING** 36](#_Toc43402140)

[**6.7 Final Outputs:** 36](#_Toc43402141)

[**Chapter 7: Conclusion** 41](#_Toc43402142)

[**Appendix A** 43](#_Toc43402143)

[**USER MANUAL** 43](#_Toc43402144)

[**References:** 47](#_Toc43402145)

**List of Figures**

[Figure 1 29](#_Toc43403555)

[Figure 2 31](#_Toc43403556)

[Figure 3 32](#_Toc43403557)

[Figure 4 38](#_Toc43403558)

[Figure 5 39](#_Toc43403559)

[Figure 6 40](#_Toc43403560)

[Figure 7 43](#_Toc43403561)

[Figure 8 44](#_Toc43403562)

[Figure 9 45](#_Toc43403563)

[Figure 10 46](#_Toc43403564)

**List of Tables**

**Acronyms and Abbreviation**

**PDBF**: Portable Database File

**JS**: JavaScript

**HTML**: Hyper Text Markup Language

**PDF**: Portable Document Format

**OLAP**: Online Application Processing

**SQL**: Structured Query Language

# **Abstract:**

The research work done is always somehow dependent on the previous findings. So, for the further advancements and testing the authenticity of the research and availability of the previous datasets, algorithms and others important contents is very critical. However, our research culture differs in unusual ways. Most of the times only the well observed findings and polished results are included

However, the algorithms, input datasets, raw result datasets, as well as scripts that were used to produce the results in the original research document are sporadically made obtainable for the public reviewers and typically not available to other researchers also. Here it leads to springing up of many questions related to the legitimacy of the research work, data sets and results published.

Unfortunately, till now instead of a long workflow which produced results for research paper, only the final result is published in journals.

Hence some of the steps which have played vital role in the research results are skipped. This is very forlorn and has been denounced in several research communities. In our system we have argued on one fragment of the problem and our dated perspective on what a research document is and how it can be refined or should look like.

As a remedy, we introduce Janiform document format. These documents are of hybrid format, i.e. they are at the same time a standard static pdf as well as a highly dynamic (offline) HTML-document. PDbF’s allow you to access the raw data and reproduce your own graphs from the raw data all of this *within* a portable document.

One of the many problems that are being faced by the research community is of raw data unavailability. As of today, data-science is becoming a hot field. Most of the research work is being carried out in this branch. Most of the times the data upon which several operations were performed become obsolete and is not available to reviewers and hence it becomes a hectic task for further advancements in that work. What we have proposed is to embed that data within a document. This would also allow the long-term preservation of the data. It would also resolve its availability questions as it is just one click away from the reader or a reviewer.

On parallel to this we also have provided visualization of the data within offline HTML format of the document. Hence, we have addressed these

issues with our utmost efforts and we hope that our work will somehow ease the issue faced in making the research work more organized.

We demonstrated a system that authorize its users to compile Janiform documents smoothly from within LATEX editors. This tool allows to preserve the workflow of raw measurement data to its culminated graphical output through all transitioning steps.

# **Acknowledgment:**

I would like to express profound gratitude to my guide Sir Muhammad Shoaib Malik and Sir Imran Ihsan for their invaluable support, encouragement, supervision and useful suggestion throughout this project work. Their moral support and continuous guidance enabled me to complete my work successfully.

I am grateful for the corporation and continuous encouragement for my honorable Dean of the Computer and AI Dr. Kashif Kifayat and Head of Department Dr. Mehdi Hassan.

Last but not the least, I am thankful and indebted to all those who helped me directly or in directly in completion of this project and project report.

Muneeb ur Rehman Zaeem Dastaghir,

**Bachelor of Science of Computer Science**

**Chapter 1**

# **Chapter 1: Introduction**

In our project we are working on the problem faced by the research community most of the times. As the availability of data along with research results and benchmarks is very important in every research conducted so far. Here arise an issue of data availability and its validation. If someone wants to carry a research on already published research paper, it is importunate for the researcher to get their hands on the previous conducted research and data which was previously gathered by the author. Unfortunately, in many cases data is not available. So, one has to regather all the previous data or to contact the previous author for the data set, which god knows better is valid or not.

Our motive is to address these problems and to come up with simple and viable solution. Our proposed solution is to embed that data with the PDBF document which in turn eradicates the question of data availability. As an additive feature we also aim to provide ML algorithms that can be applied on the data set within the PDBF document.

We do not aim to provide all the ML algorithms, but to prove that this concept can be applied realistically our motivation is to implement one model and its results on appropriate dataset.

## **Background:**

Over the course of time with the computer sciences has revolutionized. The main issue which is being faced by the research community is the management, providence and preservence of the data. However, one of the many issues of our research publication culture is that we only publish a brief summary of the final results of the long projects. Only the well-polished graphs, brief result outputs and diagrams are not sufficient for the reviewer to check the validity and the quality of the research work done by the authors.

This issue has been discussed and criticized in many research and scientific communities but the efficient till date no one has come up with the efficient solution to meet the benchmarks.

One of the very basic solution that has being used is “**ask and provide**” method. One should have to contact the author and ask them to provide the reviewer with the data and algorithms of the research problems. In many cases this is not a viable solution. Some also have reported that the actual data do not even exists. In these cases, the reviewer have to improvise and gather the data again for performing the same testing and experimentation, which is a very time consuming and hectic task. Even after such tasks the results are not reproduced with the precision that has been mentioned or shown in the published research papers. This in turn puts a question mark on the validity of the research work.

To avoid this drawback a viable and efficient solution that should address the problem stated above is importunate. This is where our Final Year Project idea comes from. The project statement clearly describes the solution that is devised by our group to eradicate this issue.

## **Objective:**

The objectives of our project are as follows:

* Make research paper easier to understand for the reviewer.
* Embed related data inside a document.
* Allow the user to create ML models on embedded data.
* Provide useful graphical visualizations of data.
* Allow manipulation of visualization by the reviewer for validating correctness of tested data.

## **1.3 Scope:**

Our motivation behind choosing this project is the gap between the problem and its solution. Our project directly addresses the predicaments with very simple and viable solution. The toolkit we aim to develop is providing functionalities like data embedding and viewing within the document. In order to achieve and deliver these functionalities our proposed solution is to make research papers dynamic. We have achieved dynamicity by making research paper an offline HTML document in which dynamic functions are being provided for the reviewer.

We also have affirmed intentions of developing add on feature of providing the reviewer with machine learning algorithms. These machine learning algorithms can be applied on the embedded data. We also hope for producing results by the applied algorithm. This task is just for a spoof concept of work that it can be done using the resources that we have today.

Soon it can be a possibility that it would become possible that is impossible today.

## **1.4 Problem Statement:**

This project plans to tackle this issue by producing Janiform reports from LaTeX documents. A Janiform document is a document that is valid PDF and HTML simultaneously. This enables the researchers to embed raw dataset inside the document and permits different reviewers to see that dataset and furthermore produce ML models on the embedded datasets utilizing the dynamic HTML version of the document.

**Chapter 2**

# **Chapter 2: Literature Review**

The problem of data and results irreproducibility is always an issue for the research community. Once a research paper is published, the data which is used by the author for the results is not always available for the reviewers. They must contact the author or to search for other online repositories for having their hands on the raw dataset. This project aims to solve this problem by generating Janiform documents from LaTeX files. A Janiform document is a document that is valid PDF and HTML at the same time. This allows the author to embed raw dataset within the document and allows other users such as reviewers to view that dataset and also generate ML models on the embedded datasets using the dynamic HTML version of the document.

## **2.1 Related Projects:**

There is only single research related to this project and that research is the base of our project.

### **2.1.1 Janiform IntraDocument Analytics for Reproducible Research**

This research shows that the research community always face a problem of data availability and irreproducibility of results, if they want to get the algorithm, graphs, dataset of the research they had personally had to contact the author of the research. That’s why this project has come into the spotlight as this problem is solved by introducing a portable database file (PDBF). These files are Janiform i.e; static PDF and highly dynamic HTML at the same time. This PDBF file also allows the user to access raw data behind the graph, perform OLAP style analysis and reproduce your own graph from the raw data, all within the same file. The user cam generates PDBF document from LaTeX files.

**Chapter 3**

# **Chapter 3: Requirement Specification**

## **3.1 Existing System:**

The system proposed us as project is novel. No such system exists which can provide similar functionality like our proposed system is proffering. There are some systems which are using similar approaches like the one proposed by our system.

For Example:

**R markdown** which is another documentation tool which provides the results insight through **R-Language** integration for producing results to be used in document. The resulting document of R-Markdown is pdf file. Although that document is integrated with the visual elements like charts, plots etc. All these elements are irreproducible and static images of results generated using **R** code.

These systems have many shortcomings like what if another person wants to recreate results or want to look on the data which was used to produce results. As today links of the datasets are provided but still questions arise about the authenticity of data. In some of the cases these links don’t even provide data unless the paper is published in renowned journals.

Another drawback is less responsiveness. As most of the research papers include figures like charts tables etc. These figures are of utmost importance as it is best possible way to summarize findings and results produced. What if these figures can be made interactive. In this way it would become easy to evaluate research results. Conventional systems lack such properties.

## **3.2 PROPOSED SYSTEM:**

Our vital aim is to make document dynamic and more responsive. We have been working on hybrid document format for our project. These documents have more than one valid formats. One can easily switch between formats by simply renaming the file.

The simple document format that is used most of the times just for reading purpose is PDF. So we are also using this format for our document. Accompanying it, as the document is in hybrid format another format will be html.

In html version of the document the reader will be provided with the services like downloading data file into local storage, viewing data table, performing SQL queries on data table, viewing plot of the data inserted into the data table.

Another functionality that we are working on to include is providing pre-trained ML model that can be applied on to the embedded data. This service can only be available in html format of the document.

**Chapter 4**

# **Chapter 4: Design**

## **4.1 SYSTEM ARCHITECTURE**

Our PDBF compiler coined as JANIC is written in java. Its architectural methodologies can be subdivided into further processes.

Firstly, a valid latex file is passed as an argument to Janic. Janic compiler will then invokes the standard latex compiler to validate and read the file. After that a PDbf-configuration is produced in first step. In addition to configuration file a simple pdf document will also be produced as draft version. Configuration file will then be read again by the Janic for relative placements of visual elements and dynamic content that has to be placed in the final hybrid document.

One can also include visual elements that are produced using in-document data for pdf version of document. For this purpose, it is also critical to produce static snapshots of those elements. These snapshots are very helpful to make a static document.

So, for this task we have used Phantom’s which produces static snaps of the visual elements. This is for the pdf view of the document.

Now after successfully creating static snapshots and determining the relative positions of textual and visual elements. Janic compiler will again read the PDbf-configuration file. This time it is done for final placements of the elements in dynamic document.

Now a final output is produced which is a hybrid document.

## **4.2 DESIGN CONSTRAINTS**

Our system, as previously mentioned is based upon java. Moreover, Latex is also very vital component in out toolkit. In designing process, we have faced problems like:

* How to efficiently handle different input files.
* How to provide enough solution of files passing to and fro in classes.
* How to keep our work as simple as possible for the reviewers and computer science related community.

So, that they can easily add their spadework to make our system more efficient and up-to the future advancements.

Keeping in-view the mentioned concerns. We hope that the designed system will meet the expectations and accomplish the given task in effectual manner. As we have aimed to embed data into the html-document. We have faced data related hurdles which are not negligible because of complexity and size related concerns.

Most of the times the data files provided as the standalone links are in csv or textual form. So, keeping in mind the majority we have designed our system to perform in well-ordered way for these data files. Dealing this issue was very important because in this way the efficiency and compilation time is at stake. It is not yet impossible to embed data files having too much size and incompatible data types. These reasons do not seem big but when we add them, they become considerable and adds up different factors that can in turn make our system’s overall performance at risk.

We have assumed that every targeted user of our system has basic know how of Latex and can write simple LaTeX documents. The knowledge of command prompt will also help in using the system without any hurdles.

## **4.3 DESIGN METHODOLOGY**

We have done our utmost to design our system with low complexity and easy to modify in future.

Our implementation of the system is divided into two stages.

1. Pre-Compilation stage.
2. Compilation stage.

In this way it will be very easy for the the reviewer to understand the working of system. Another good factor is that, modifications can be enacted easily without effecting the other components of system.

**Pre-Compiler:**

The logical implementation, file handling, component related methods are invoked in this stage. Every component of the hybrid document has its own method declared separately with all the related logical providence.

**Compiler:**

Here the finalized results along with the authentication flags is received and output is produced. Flags also serve as the error specifiers of the methods invoked in pre-compilation stage. Hence it becomes easy to trace the error back to the line where the system stopped execution.

This class hierarchy considered because it addresses the problem’s solution in efficient and easy to understand manner. Other designs were also considered but this one was most feasible in our system’s development.

## **4.4 GUI DESIGN**

Our system does not require that much graphical interface. It is because our toolkit is not any mobile or web-based application. They required graphical interfaces for attracting the users and making the product look new and refreshed.

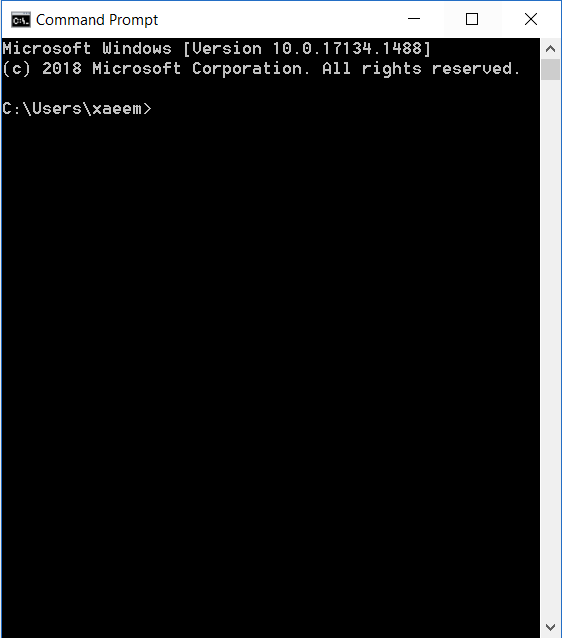
As we can address our system as the document editing tool and keeping in view the other document producing tools none of them have graphics like MS Word, WPS Office etc.

This was the main reason due to which there’s no need to develop a separate interface.

Our user will use LaTeX editor for writing .tex file. Now it depends on the user and availability of LaTeX editors. One can any of them like MikTeX, TexMaker, TexWorks etc. These editors are not that complex, and one can easily understand their working methodology.

As for keeping the system easy to use we have utilized the OS built-in Command line interpreter which is commonly known as command prompt.

After successful completion of writing latex file. User must start command prompt. A window will open having black background.



Figure

Here a user is required to type custom command to evoke JANIC compiler which after some processing creates a Janiform document.

**Chapter 5**

# **Chapter 5: System Implementation**

## **5.1 System Architecture**

A picture containing clock

Description automatically generated

Figure

### **5.1.1 Architecture Description:**

The Architecture of our project is as follow:

* LaTeX file will be passed as an input to PDBF compiler.
* PDBF compiler will generate Janiform document which is a valid PDF and HTML document at the same time.
* PDF document will be static document without any dynamic functionalities.
* HTML document will be dynamic version of PDBF document. All functionalities like accessing embedded data and creating ML models will be done through this file.
* Data goes where the document goes.

### **5.1.2 Project Flow**

A screenshot of a cell phone

Description automatically generated

Figure

**Working:**

This flow shows the working of our project as the user write it’s LaTeX file in TeX Studio/MiKTex the user gives that LaTeX file as an input to the PDBF compiler. The PDBF compiler compiles the file and generate a Janiform document which is a static PDF as well as dynamic HTML versions of the document and that dynamic HTML document with all functionalities like accessing raw data, embedded data and creating ML through that file. The benefit of PDBF file is the raw data and the embedded data in the document will also goes where the file goes that means anybody can access that data.

### **5.1.3 Tools and Technology:**

**Any LaTeX Distribution:**

On the first step of using this toolkit a user must have active and fully functioning Latex distribution along-with the Latex text editor. This two software’s must have to be already installed on personal computer.

The Latex distribution we recommend for the users is MikTex with the text editor TeX Maker. These are easily available and can be installed from any online source.

**IntelliJ:**

IntelliJ is the most renowned java IDE that is easily available. Now most of the java development is being done using this IDE.

The vital part of our project is based upon java and we have used IntelliJ. One of the reasons behind using this is its ease of use and also a large community of programmers are using this IDE. Due to which vast numbers of online blogs and helping material is available in case of difficulties.

**VS Code:**

Development of dynamic part is done using visual studio code. It is a light-weight code editor which allows wide array of programming languages. Almost all of the important debugging tools and languages are compatible. One can also install very small sized extensions for different languages and task.

We have used VS-CODE for our JavaScript, HTML and CSS part of the project. This code editor is also popular among programming communities for its very interactive and easy to use functions.

**Chapter 6**

# **Chapter 6: System Testing Evaluation**

## **6.1 GUI TESTING**

Our project is a research-based project. So, no specific GUI is designed for users. Although tools through which users can compile Janiform document are:

1. Latex Editor.
2. Command Prompt.

Any Latex editor can be used to write tex file for our toolkit but MikTex is suggested as we ourselves have used it.

## **6.2 USABILITY TESTING**

The main problem that majority of the people will face in compiling a Janiform document is of writing a LaTeX file. The latex is not commonly used document editor, only handful of the people have known how of LaTeX.

Most of the tasks done in latex are through using commands or tags. LaTeX also use packages for different tasks. One of the man hectic task is to include relative packages for specific commands that one wants to use.

So, for the first-time users of the toolkit it is recommended to watch tutorials or have some of the exposure with the basics of using LaTeX as a document editor.

## **6.3 COMPATIBILITY TESTING**

In this section, we describe the operating system that was used to develop and run our proposed system. We will also detail about our systems compatibility with other platforms.

The system that we have developed is just for the personal computers but not for any mobile devices. As far as the compatibility is concerned, we just have tested our system on Windows Operating System. All of the necessary tools for compiling are available on windows OS easily.

Apart from it we’ve tested some of the dynamic functions. They might become an issue for some of the users. It is advised to install and use the latest versions of LaTeX. Although if you are not facing any problems in compiling the document but have issues regarding the dynamic functionalities. It might be possible that the web browser you are using for HTML rendering do not support that JavaScript and html properties. This usually happens because of the frequent updates and changes in the web-browser.

One must catch up with these changes for smooth and sound working of the Janiform document.

Along it we have developed our system using java and JavaScript and the operating system used is Windows 10.

Major platforms such as Linux, Mac OS X and Windows support and provide compilers for java and JavaScript. So, it stands to reason that no issue will be presented on majority of operating systems. However, we have not implemented our software on a variety of machines running different operating systems. Therefore, we cannot state with absolute certainty which exact of systems are compatible. Refer to the documentation of each individual software component to determine compatibility.

## **6.4 LOAD TESTING**

The system we have developed accepts LaTeX file as an argument. After that further processing is done. In further steps the LaTeX compiler is invoked, and other processing work is done. As we have embedded the data file in the document. So, the data size and output document size is directly proportional.

So, it is advised not to attach too big data file into the document as it would add more time to the final output. It also increases the document size. In some cases it will become impractical to sit and wait for final output document if too big data file is attached.

## **6.5 INSTALLATION TESTING**

Installation of toolkit is the first step that decides that how this thing will work in the future. It is the most vital step in using the system. So mostly it is advised to install the program carefully to get rid of the unusual errors and problems that would cause wastage of time.

In our case it is advised to use Latest JAVA Runtime Environment and check whether it is working or not. Working of the toolkit is dependent on this step.

## **6.6 EXCEPTION HANDLING**

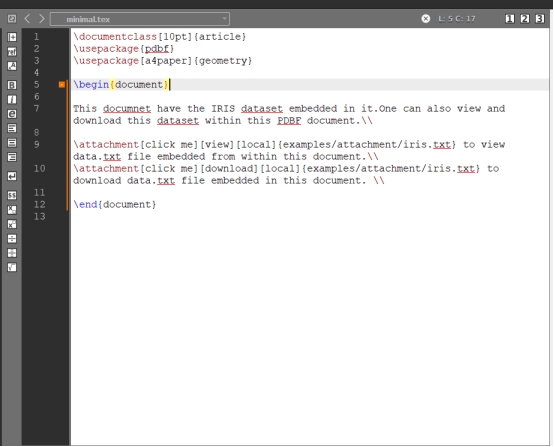
An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. Exception handling is the process of responding to the occurrence of exceptions in such a way that they no longer disturb the program’s execution.

During the development of the proposed system several errors and exceptions were encountered. To deal with the exceptions, exception handling methods were implemented using try and catch keywords.

While development of system we have handled the exceptions implemented try-catch methods to avoid the system crash.

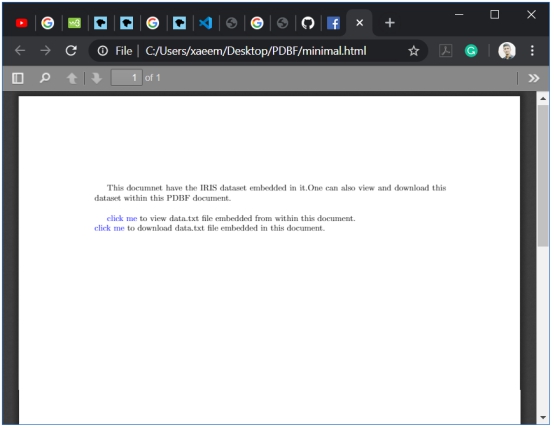
Implementations of the conditional statements are also used where it is critical to avoid user held errors while using this toolkit.

## **6.7 Final Outputs:**



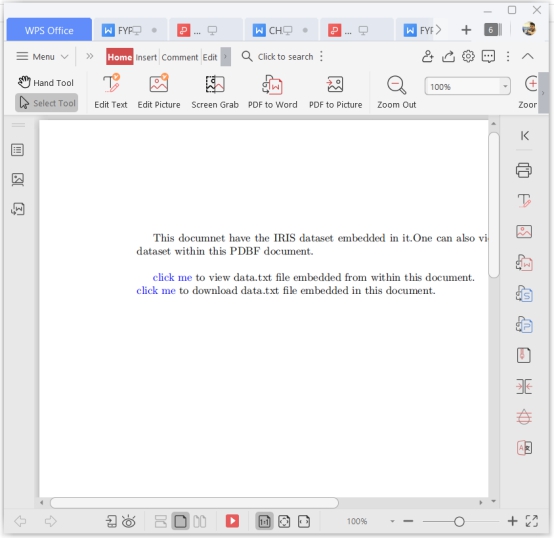
Figure

Latex Document Specifying the latex command for embedding data.



Figure

Html Document through which user can view and download data set within the document.



Figure

Pdf version of the document which is static and do not provide any functionalities.

**Chapter 7**

# **Chapter 7: Conclusion**

This system has opened a new horizon of portable data files and hybrid documents. As the documents have two formats pdf and HTML. So, there are many possibilities and ways to improve this system. The dynamic copy of the document has the potential of further advancements which can be done. As till now we have accomplished the task of embedding the data into the document which solved the issue of raw data availability and preservence for the long terms. Yet there are many horizons that are still undiscovered.

We cannot provide many of the functionalities till now due to the lack of technologies and unavailability of resources.

Who knows that in future it might become possible. In future we want to embed trained and untrained machine learning models. These models can then be applied on embedded data and can also be trained on data within the document. We would also like to provide unconstrained data file limit by implementing most efficient data compression methods, but we cannot say when it would all become reality.

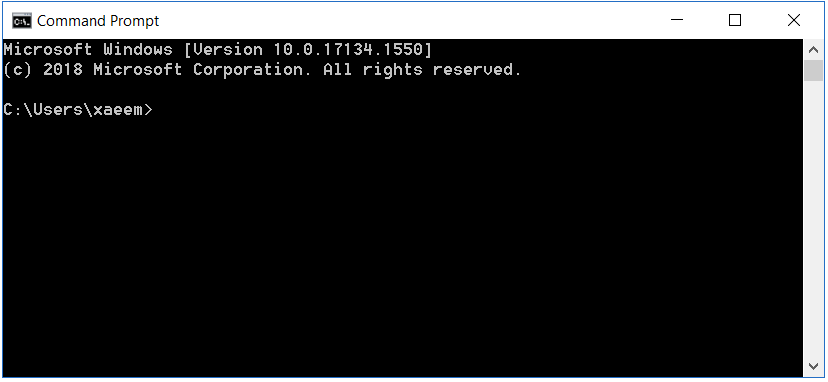
Someday it will be possible for us to make our future assumptions into reality, because we know that someday, someone can do this work more efficiently than us.

# 

# **Appendix A**

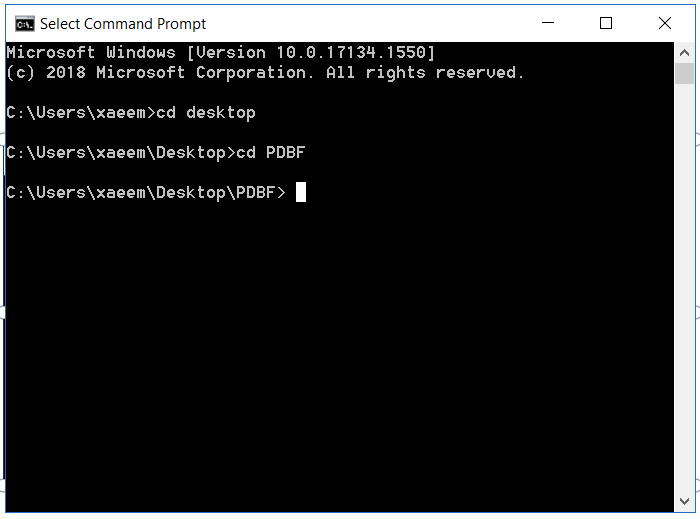
## **USER MANUAL**

* Download toolkit from git or run it from your personal device if you have this toolkit already in your local storage.
* Save your latex file in the folder where toolkit is located.
* Run command prompt as administrator.



Figure

* Change the working directory using DOS commands to the folder where toolkit is stored in local storage.

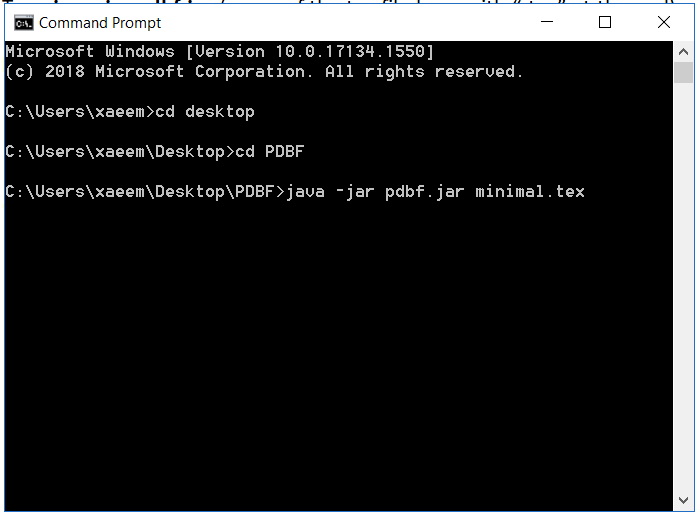


Figure

* Now after changing the working directory.

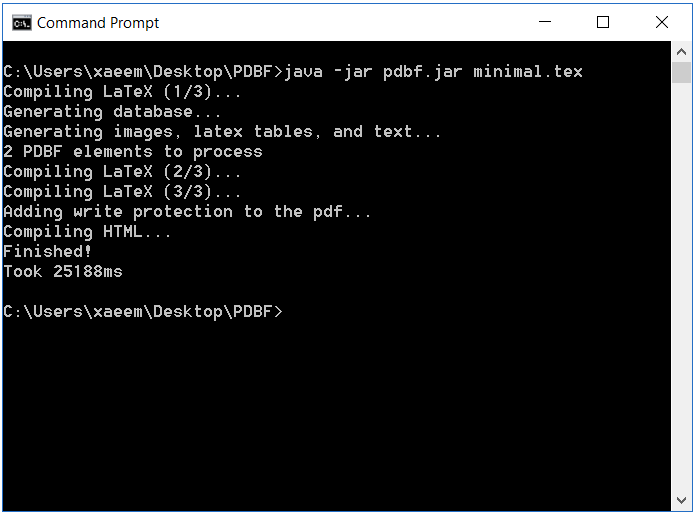
Type **java -jar pdbf.jar** (name of the tex file here with “.tex” at the end).

For example **java -jar pdbf.jar hello\_world.tex**.



Figure

* After that the compilation steps would start and on successful completion HTML file will be generated as output. Change that file name to .pdf and it is valid pdf also.



Figure

# **References:**

**[1] Research Paper Repository:** Janiform Intra-Document Analytics for Reproducible Research

[https://bigdata.unisaarland.de/publications/p1972- dittrich.html](https://bigdata.uni-saarland.de/publications/p1972-dittrich.html)

**[2] GitHub Repository:**

<https://github.com/xaeem-hussain/IN-DOC-RESULT-ANALYSIS?fbclid=IwAR1Ov8nljDYxkpvIg5rX-hQwcCzGUJE90aCZmwF6TQFJYvc_aLYRgTP2IrQ>