**A PRELIMENERY REPORT ON**

**USING DEEP LEARNING TO BUILD AN ACCURATE INTEGRATION AND DIFFERENTIATION SYSTEM**

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

**BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)**

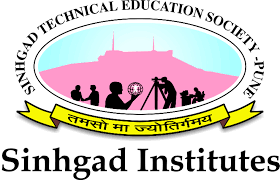
##### SUBMITTED BY

JEET BARDAPURKAR B150364240

CHAITANYA GOSKI B150364317

POOJA SWAMI B150364507

LOMESH GAME B150364307



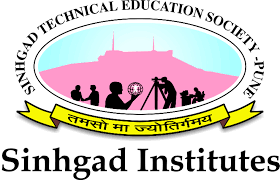
## DEPARTMENT OF COMPUTER ENGINEERING

## STES’S SMT. KASHIBAINAVALECOLLEGE OF ENGINEERING

VADGAON BK, OFF SINHGAD ROAD, PUNE 411041

## SAVITRIBAI PHULE PUNE UNIVERSITY

## 2020-2021



**CERTIFICATE**

This is to certify that the project report entitles

**“USING DEEP LEARNING TO BUILD AN ACCURATE INTEGRATION AND DIFFERENTIATION SYSTEM”**

Submitted by

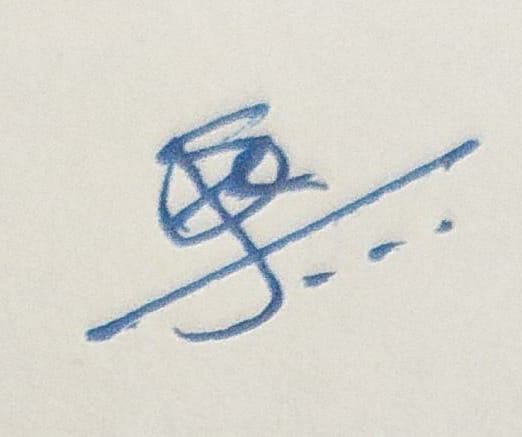
JEET BARDAPURKAR Exam No : B150364240

CHAITANYA GOSKI Exam No : B150364317

POOJA SWAMI Exam No : B150364507

LOMESH GAME Exam No : B150364307

is a bonafide student of this institute and the work has been carried out by him/her under the supervision of **Prof. S.Y. KULKARNI** and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University, for the award of the degree of **Bachelor of Engineering** (Computer Engineering).



**(Prof.(Mrs.) Snehal Y. Kulkarni)** **(Dr. P. N. Mahalle)**

Guide Head,

Department of Computer Engineering Department of Computer Engineering

**(Dr. A. V. Deshpande)**

Principal,

Smt.Kashibai Navale College of Engineering Pune – 41

Place : Pune

Date : 1/04/2021

**ACKNOWLEDGEMENT**

We would like to thank our guide prof. S.Y. Kulkarni ma’am for providing the necessary guidance, motivation and encouragement for taking up this project. Her contribution to this project has been invaluable.

We would also like to thank our Head of Department sir, Dr. P.N. Mahalle and our principal Dr. A.V. Deshpande sir, for making available all the required resources and guidance which helped us to finish our tasks in various stages of our project.

Lastly, we thank our family and friends for their support, love and blessings without which this project would have not been possible.

**JEET BARDAPURKAR**

**CHAITANYA GOSKI**

**POOJA SWAMI**

**LOMESH GAME**

**ABSTRACT**

Computer algebra systems have been used to solve complex mathematical computations ever since the first computer algebra system Macsyma was released in 1978. These computer algebra systems are used in many diverse fields to solve complicated problems, like mathematics, aviation, architecture, data science, research, and many more. However, these systems can be slow and sometimes even fail to produce an answer in the stipulated time. Also, the algorithms used in these systems can be too convoluted for a human to understand. To solve these problems for integration and differentiation, deep learning, more specifically Natural Language Processing can be applied. System can represent mathematical equations in the form of sequences of characters and generate datasets of sequences representing equations with their integrated forms and their differentiated forms. These datasets can be used to train seq2seq model which can give the solutions to new problems. This model can be used by anyone to solve their integration and differentiation problems for a wide variety of applications.

**TABLE OF CONTENTS**

LIST OF FIGURES vi

LIST OF TABLES vii

**CHAPTER TITLE PAGE NO.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr. No.** | | | | **Title of Chapter** | | | **Page No.** |
| **01** | | | | **Introduction** | | | 1 |
|  | 1.1 | | | Overview | | | 1 |
|  | 1.2 | | | Motivation | | | 2 |
|  | 1.3 | | | Problem Definition and Objectives | | | 2 |
|  | 1.4 | | | Project Scope & Limitations | | | 3 |
|  | 1.5 | | | Methodologies of Problem solving | | | 3 |
| **02** | | | | Literature Survey | | | 5 |
| **03** | | | | Software Requirements Specification | | | 8 |
|  | | 3.1 | | Assumptions and Dependencies | | | 8 |
|  | | 3.2 | | Functional Requirements | | | 8 |
|  | | 3.3 | | External Interface Requirement | | | 8 |
|  | |  | | 3.3.1 | | User Interface | 8 |
|  | |  | | 3.3.3 | | Communication Interface | 9 |
|  | | 3.4 | | Nonfunctional Requirements | | | 9 |
|  | |  | | 3.4.1 | | Performance Requirements | 9 |
|  | |  | | 3.4.4 | | Software Quality Attributes | 9 |
|  | | 3.5 | | System Requirements | | | 9 |
|  | |  | | 3.5.1 | | Database Requirements | 9 |
|  | |  | | 3.5.2 | | Software Requirements | 10 |
|  | |  | | 3.5.3 | | Hardware Requirements | 10 |
|  | | 3.6 | | Analysis Models: SDLC Model to be applied | | | 10 |
| **04** | | | | System Design | | | 12 |
|  | | 4.1 | | System Architecture | | | 12 |
|  | | 4.2 | | Mathematical Model | | | 13 |
|  | | 4.3 | | Data Flow Diagrams | | | 14 |
|  | | 4.4 | | Entity Relationship Diagrams | | | 16 |
|  | | 4.5 | | UML Diagrams | | | 17 |
| **05** | | | | Project Plan | | | 20 |
|  | | | 5.1 | Project Estimate | | | 20 |
|  | | |  | 5.1.1 | Reconciled Estimates | | 20 |
|  | | |  | 5.1.2 | Project Resources | | 20 |
|  | | | 5.2 | Risk Management | | | 21 |
|  | | |  | 5.2.1 | Risk Identification | | 21 |
|  | | |  | 5.2.2 | Risk Analysis | | 21 |
|  | | |  | 5.2.3 | Overview of Risk Mitigation, Monitoring, Management | | 25 |
|  | | | 5.3 | Project Schedule | | | 25 |
|  | | |  | 5.3.1 | Project Task Set | | 25 |
|  | | |  | 5.3.2 | Task Network | | 26 |
|  | | |  | 5.3.3 | Timeline Chart | | 26 |
|  | | | 5.4 | Team Organization | | | 27 |
|  | | |  | 5.4.1 | Team structure | | 27 |
|  | | |  | 5.4.2 | Management reporting and communication | | 28 |
| **06** | | | | Project Implementation | | | 29 |
|  | | | 6.1 | Overview of Project Modules | | | 29 |
|  | | | 6.2 | Tools and Technologies Used | | | 29 |
|  | | | 6.3 | Algorithm Details | | | 30 |
|  | | |  | 6.3.1 | Algorithm | | 30 |
| **07** | | | | Software Testing | | | 31 |
|  | | | 7.1 | Type of Testing | | | 31 |
|  | | | 7.2 | Test cases & Test Results | | | 34 |
| **08** | | | | Results | | | 35 |
|  | | | 8.1 | Outcomes | | | 35 |
|  | | | 8.2 | Screen Shots | | | 35 |
| **09** | | | | Conclusions | | | 38 |
|  | | 9.1 | | Conclusions | | | 38 |
|  | | 9.2 | | Future Work | | | 38 |
|  | | 9.3 | | Applications | | | 38 |
|  | | | | **Appendix A**: Problem statement feasibility assessment using, satisfiability analysis and NP Hard, NP-Complete or P type using modern algebra and relevant mathematical models.  **Appendix B**: Details of paper publication: name of the conference/journal, comments of reviewers, certificate, paper.  **Appendix C**: Plagiarism Report of project report. | | | 39  40  41 |
|  | | | | **References** | | | 42 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure** | **Illustration** | **Page No.** |
| 3.7 | SDLC Diagram | 10 |
| 4.1 | System Architecture | 12 |
| 4.2 | Data Flow Diagram level 0 | 14 |
| 4.3 | Data Flow Diagram level 1 | 14 |
| 4.4 | Data Flow Diagram level 2 | 15 |
| 4.7 | ER Diagram | 16 |
| 4.8 | Use Case Diagram | 17 |
| 4.9 | Class Diagram | 18 |
| 4.10 | Sequence Diagram | 18 |
| 8.1 | User Interface | 35 |
| 8.2 | System without data processing | 36 |
| 8.3 | Example inputs, predictions and targets | 36 |
| 8.4 | Folder view | 37 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table** | **Illustration** | **Page No.** |
|  |  |  |
| 5.1 | Risk Table | 22 |
| 5.2 | Risk Probability definitions | 22 |
| 5.3 | Risk Analysis 1 | 23 |
| 5.4 | Risk Analysis 2 | 23 |
| 5.5 | Risk Analysis 3 | 24 |
| 5.6 | Risk Analysis 4 | 24 |
| 5.7 | Risk Mitigation | 25 |
| 5.8 | Timeline Chart | 26 |
| 5.9 | Management Reporting and Communication | 28 |
| 7.1 | Test cases | 28 |

**CHAPTER 01**

**INTRODUCTION**

* 1. **OVERVIEW**

As is obvious from the name, a “computer” is basically something that computes. The entire motivation behind making computers in the first place was to make computation easy. The earliest known calculating device is probably the abacus. It dates back at least to 1100 BCE and is still in use today. Charles Babbage, also known as the “Father of Computers” conceived the first computer ever in the form of the Difference Engine. Charles Babbage began working on the difference engine in 1822. It was designed to calculate the values of polynomial functions by using the method of finite differences. Although the machine was never actually finished because of cost issues, this machine remains one of the first conceptions of computers. In 1930 an engineer named Vannevar Bush at the Massachusetts Institute of Technology (MIT) developed the first modern analog computer. The Differential Analyzer, as he called it, was an [analog](https://www.merriam-webster.com/dictionary/analog) calculator that could be used to solve certain classes of differential equations, a type of problem common in physics and engineering applications that is often very tedious to solve. As more and more innovations took place in the world of computing and calculations, more and more sophisticated technologies were invented for use in many real world applications like accounting, taxation, census, etc. Computations which took humans minutes, hours and sometimes even days to finish by hand, pen and paper, could be finished by a computer in a very short period of time.

Calculation has an application in every aspect of the functioning of technology in modern society. These include the domains of mathematics, research, chemistry, physics, aviation, architecture, construction, computing, artificial intelligence and many more. One example of such calculation is functional integration. It is a very complex problem for both humans and computers to solve. The application of deep learning, more specifically NLP will help to perform these types of problems more efficiently

* 1. **MOTIVATION**

In 1978, one of the oldest general purpose Computer Algebra System still in use today called Macsyma was released to the public. Since then, CAS’ have been heavily used for various different applications in fields like Computer Science, Chemistry, Architecture, Aviation and many more where complex algebraic calculations are required to be performed. Many different applications in such areas are solved accurately and efficiently based on a computer’s power of heavy and complex computing harnessed through these CAS’. However, an issue with these systems lies specifically in the problems of calculus. Firstly, sometimes the algorithms used by these CAS’ can be too hard and tedious for humans to understand. For instance, the Risch Algorithm used to solve integration questions, is over a hundred pages long. More importantly, for very complex problems, these systems can be slow and might time out before coming up with the correct answer. To solve these problems, system will use deep learning, more specifically Natural Language Processing, to process the equations and perform integration and differentiation as required by the user. This deep learning model can be used by anybody who wants to perform calculus calculations on their equations.

* 1. **PROBLEM DEFINITION AND OBJECTIVE**

Neural networks have a reputation for being better at solving statistical or approximate problems than at performing calculations or working with symbolic data. However, they can be surprisingly good at more elaborated tasks in mathematics, such as symbolic integration and solving differential equations.

Hence, the system will use deep learning for performing accurate integration and differentiation.

* 1. **PROBLEM SCOPE AND LIMITATIONS**

The scope of this system is to build a deep learning model that attempts to learn to solve calculus problems, namely integration and differentiation. A project with a similar problem statement, used over 252 Million tuples or roughly 70 G.B. of data. Due to resource and hardware constraints, the system cannot be trained on this amount of data required for such a model, hence the system will be implemented in its entirety and will be able to take input and give output without being actually capable of solving the problem right now. Other algebraic operations such as addition, subtraction, multiplication, division were not implemented in this calculator because they are already performed by basic algebraic calculation systems with sufficient speed. Implementing redundant operations would increase the time, resources and complexity required in this project for no added benefit. Also, this system does not intend to replace the currently used computer algebra systems since the accuracy is not high enough to be completely reliable, but maybe complement them in the future.

* 1. **METHODOLOGY OF PROBLEM SOLVING**

Problem solving is an important part of any project as it enables to assert control over the environment and solution to ever changing and emerging changes and possibilities. The foremost concern comes in understanding and accepting that exist a problem and only then the solution can be worked out. How to understand problem:

Finding the root cause: Calculus problems are solved using fixed rules in computer algorithms. Although accurate, it is a worthwhile cause to attempt to find different ways that this can be done and improved upon.

Starting at the bottom: Firstly, all the data about the different ways to solve this problem is gathered. This data is compiled and the limitations in the current system are identified.

Starting at the top: A deep learning approach can be applied to this problem to try to solve it. This will be a completely fresh approach and the technology can help in identifying patterns in an equation which will in turn help to solve the problem.

**CHAPTER 02**

**LITERATURE SURVEY**

In [1], the authors show that AI models can perform symbolic maths better than algebraic computer systems. This paper describes a comprehensive method to build AI models to perform symbolic maths. The disadvantage is that even though it has better performance than algebra computer systems, the authors still recommend that the results from this system be verified.

In [2], the authors describe a new architecture for the problem of learning continuous algebraic and logical expressions, called neural equivalence networks. Neural equivalence networks, can effectively represent semantic equivalence, even of expressions that are very different. However, changes in syntax can lead to very large changes in semantics, which can be difficult for continuous neural architectures.

In [3], the authors involve neural programming which involves training neural networks to learn programs, mathematics, or logic from data. This framework obtains high accuracies and generalizes significantly better to expressions of higher depth and is able to fill partial equations with valid completions. However, this is only applicable to the narrow area of function generalization and more research is needed.

In [4], the goal of the authors is to take a step towards solving DEs through neural programming. This paper proposes a generalizable and scalable neural solver for differential equations. However, this paper does not talk about integration.

In [5], the system involves neural machine translation which aims at building a single neural network that can be jointly tuned to maximize the translation performance. This paper achieves a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. However, it might be hard for this model to handle rare or unknown words.

In [6], the authors propose efficient inference procedures that allow applications to perform both parsing and language modelling. This paper provides better parsing in English than any previously published supervised generative model and better language modeling than state-of-the-art sequential RNNs in English and Chinese. However, these systems may have difficulty in translating words which are not used often.

In [7], the authors propose a hybrid model, called NMT+RNNG, that learns to parse and translate by combining the recurrent neural network grammar into the attention-based neural machine translation. This model learns to parse and translate simultaneously, and training it encourages both the encoder and decoder to better incorporate language. However, this is designed for natural languages and will be hard to apply on mathematical equations.

In [8], the authors propose a recurrent network architecture in conjunction with an appropriate gradient based learning algorithm called LSTM. LSTM solves complex, artificial long time lag tasks that have never been solved by previous recurrent network algorithms much faster. However, LSTM take longer to train, take more memory and easy to overfit. It is also hard to apply dropouts to LSTM’s.

In [9], the authors propose an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. Adam is the default optimization algorithm currently and helps in faster gradient descent and it is more accurate than SGD. However, choosing a proper learning rate can be difficult. Also, it is easy to get trapped in local minima for highly non-convex error functions.

In [10], the author outline a system where elementary functions are built up from the rational functions using only exponentiation, trigonometric, inverse trigonometric and algebraic operations. This paper gives the fixed algorithm to perform integration on a computer. The algorithm is very long and difficult to understand and implement.

**CHAPTER 03**

**SOFTWARE REQUIREMENT SPECIFICATION**

**3.1 Introduction**

Assumptions:

1. The available data set used is accurate and usable.
2. The equations used in the data and the system are in prefix form.
3. The set deadlines and schedule are attainable
4. The user will enter the equation in the required format.

Dependencies:

1. The implementation, planning and requirement analysis needs to be completed before we proceed with Implementation phase.
2. The chunks of code must be written first and then the checking of each module is done
3. Once all the individual chunks of code are completed and tested the complete system will be tested for the expected result.
4. For the documentation, the implementation needs to be completed.

**3.2 Functional Requirements**

* The user must be able to enter a mathematical expression into the system to perform the operation required by the user.
* The system must be able to perform the required operations on the input given by the user.
* The solution generated must be displayed to the user with the shortest delay possible.

**3.3 External Interface Requirements**

**3.3.1 User Interfaces**

A window interface where the user can enter their equation and obtain the output from the model

**3.3.3 Communication interface**

Communicating the result to the user through the UI window

**3.4 Non-functional Requirements**

**3.4.1 Performance Requirement**

System needs to ensure that the loading time is as short as possible. Also, the delays caused during taking data from the user and delivering results back to the user must also be minimized. System must ensure that the data storage and access delays are not too high. A GPU, such as RTX 2070 or above, is used in deep learning since it is faster, and better suited for the task than a CPU. System can use a GPU like this to increase the performance of the deep learning operation.

**3.4.3 Software Quality Attributes**

The system must have good performance so that user queries can be processed with greater speed. The software must also be easily usable by the user. The software must be reliable and available for whenever a user wants to access it. Also, all the industry standards and guidelines regarding security must be followed to protect the user. The software must be designed in such a way that it is easy to maintain and it is easy to implement changes and scale in the future. The various modules and the components designed must be reusable and in case of error, the system must be supportable to solve the errors and easily testable.

**3.5 System Requirements**

**3.5.1 Database Requirements**

File system to store the equations in the dataset

**3.5.2 Software Requirements**

* Windows/Mac/Linux OS
* Python 3
* Anaconda
* Jupyter Notebook

**3.5.3 Hardware Requirements**

* A computer with a Windows/Mac/Linux OS installed
* 8GB RAM and 64GB ROM
* Professional-level GPU for training
  1. **ANALYSIS MODEL: SDLC MODEL TO BE APPLIED**

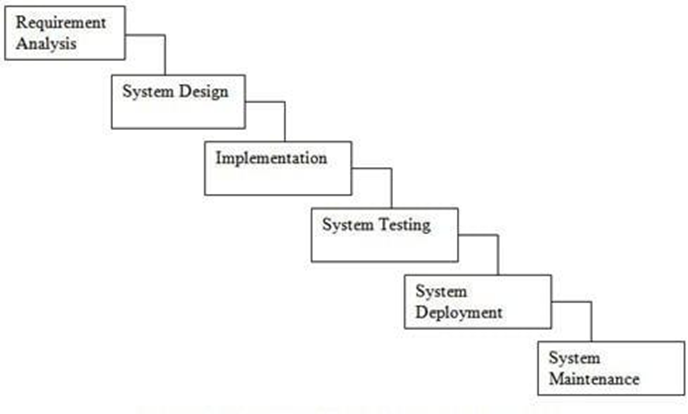


Fig 3.7 SDLC diagram

Requirement Analysis: The existing computer algebra systems were surveyed and problems were analyzed. Also, various papers were surveyed and all the various approaches taken by various researchers were taken into consideration. Based on this, the requirements were selected.

System Design: In this phase, System Architecture was designed according to the scope and the requirement gathered in previous phases. In this phase, various UML Diagrams like Sequence Diagram, Activity Diagram, and Use case Diagram, Class Diagram to visualize the working of the system.

Implementation: Python programming language will be used to develop Machine Learning models based on all the information and requirements gathered in previous phases to develop a system which attempts to perform differentiation or integration. The dataset will be taken from GitHub, where it has been made available.

Testing: Once the system is implemented, it will be tested against equations to check whether the system works as intended. If they are not met, then either the implementation is changed or the requirements are changed.

Deployment Maintenance: The completed and tested system will be deployed as a software to perform operations on user’s input equations and improvement of the system will be carried out based on the continuous feedback received.

.

**CHAPTER** **04**

**SYSTEM DESIGN**

**4.1 SYSTEM ARCHITECTURE**

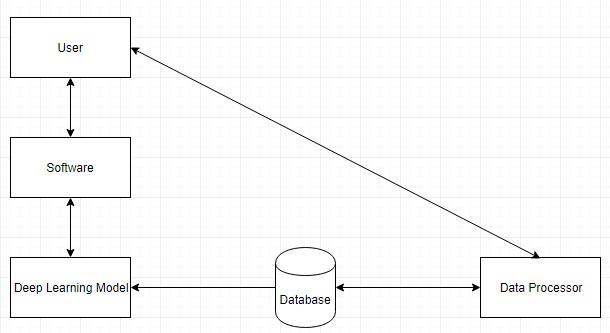


Fig 4.1 System Architecture

The figure 4.1 describes the architecture of system. The user is the person who will be using the system. In system, the user interacts using his software with the deep learning model. The software can be something that the user can use to interact with the model like Jupyter notebook, command line, etc. The model is trained using data stored in the database. The data in the database can be downloaded from the internet or can be generated by using one or more of the 3 methods of generating data – forward, backward and Integration by Parts. In the system, the model uses downloaded data to train. The user enters data into the system using the I/O software. The data is then reprocessed using the functions defined to process the data. This data is then used to train the model.

The model is an encoder-decoder model which is capable of mapping sequences of tokens to other sequences of tokens. The user can interact with this model through a UI software like a python library or command prompt by inputting equation and getting an output with the required operation performed on the input. The user’s input is first processed to make it compatible with the model. Then, it is passed to the model. The model gives some output which is processed and then displayed to the user.

**4.2 MATHEMATICAL MODEL**

System Description:

Input: Equation from the user and the operation to be performed

Output: The solution equation

Mathematical Model:

S = (I, O, F)

Where,

F = f1, f2, f3–set of functions

I = i1, i2, i3-set of inputs

O = o1, o2---set of outputs

(a) S: System.

I = F i1, i2, i3 – Inputs

(b) I1: Distance.

F = F f1, f2, f3 -Functions

Where,

(a) f1: Integration

(b) f2: Differentiation

O = F o1 - Outputs

Where,

(a) O1: The solution equation

Success Conditions: To perform the operation required by the user on the equation entered by the user

**4.3 DATA FLOW DIAGRAMS**

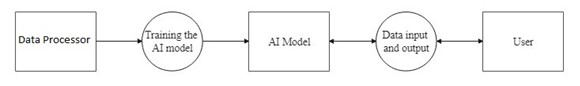


Fig 4.2 Data Flow Diagram level 0

The figure 4.2 shows the data flow diagram level 0. The data processor processes raw data. The AI model uses this data for training. The user is the user who will use the system. The model is a deep learning model. This trained model is then used by the user to perform calculations. The user can input their required equations, they will be processed and the required operation will be performed. The output will then be delivered to the user.



Fig 4.3 Data Flow Diagram level 1

The figure 4.3 shows the data flow diagram level 1. The data processor processes raw data. The AI model uses this data for training. The model is a deep learning model. The user is the user who will use the system. The user needs to interact with this model. So, the user needs some UI capabilities and for that the user requires some software like command line, Jupyter notebook, window etc. Using this, the user can enter equation, it will be processed and output will be given to the user from the system.

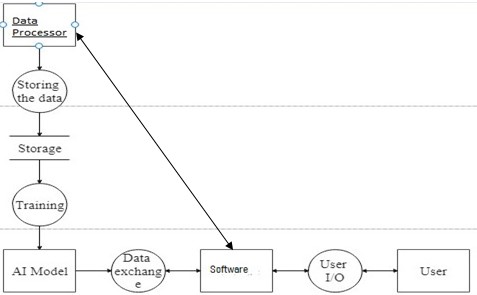


Fig 4.4 Data Flow Diagram level 2

The figure 4.4 shows the data flow diagram level 2. The data processor processes raw data. The model is a deep learning model. The processed data from raw data will be stored in a storage system such as file system. The AI model uses this data for training. The user will interact with this model. So, the user needs some UI capabilities and for that the user requires some software like command line, Jupyter notebook, etc. Using this, the user can enter equation, it will be processed and output will be given to the user from the system.

**4.4 ENTITY-RELATIONSHIP DIAGRAM**

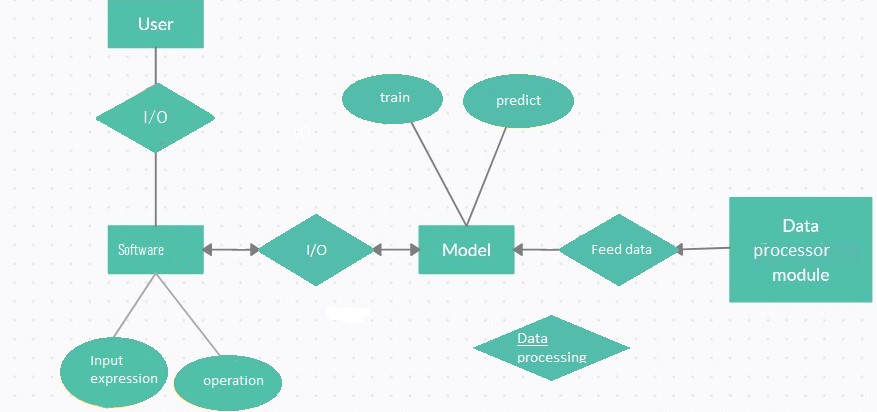


Fig 4.5 ER Diagram

The figure 4.5 shows the ER Diagram. The ER diagram shows how the various entities in this system are related. The user is the user who will use the system. The user interacts with the model through the I/O software. The I/O software can be something like Jupyter Notebook, command line, etc. The user can enter the equation and the operation to be performed into the I/O software. The model is a deep learning model which will be trained using the data processed from the raw data. The model will then take the user’s data in processed form and give a solution.

**4.5 UML DIAGRAMS**

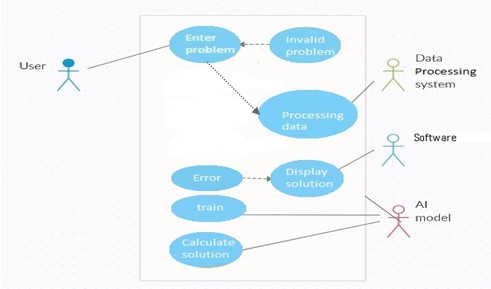


Fig 4.6 Use Case Diagram

The figure 4.6 shows the Use Case Diagram. In the use case diagram, the various operations available to various stakeholders are shown. The user is the user who will use the system. The AI model is a deep learning model. The data processing system will process raw data to train the model. The user enters the problem equation. The AI model shows either the solution or an error to the user through their software. The data processing system will be used to convert the data in the system into the required form.

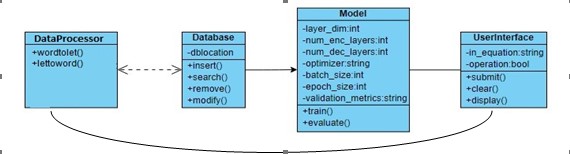


Fig 4.7 Class Diagram

The figure 4.7 shows the class diagram. The class diagram shows the potential classes that can be made in the system. The data processor is used to process the data and convert it to the required form. Database will be used for database operations. Model will be used to implement the model. It will also contain the various methods and variables associated with the model such as optimizer, train, evaluate, etc. The user is the user who will use the system. To do that, the user needs to be able to interact with the system. UserInterface will be used for that purpose.

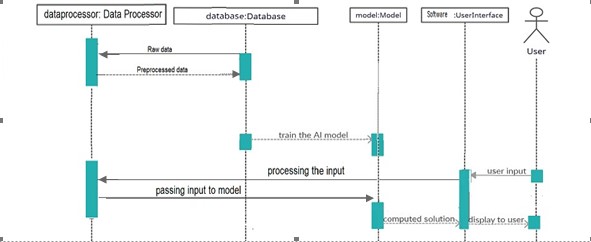


Fig 4.8 Sequence Diagram

The figure 4.8 shows the sequence diagram. The sequence diagram will show the sequence of operations in the system. The data processor is used to process the raw data and convert it to the required form. The processed data is stored in the database. The model is a deep learning model. The model will then be trained on this data. The user can then interact with this model to get the solutions to problems. The user will interact with the system using a software such as command line or Jupyter Notebook. The user’s input will be processed and entered into the model and the model’s output will be delivered to the user through the UI software.

**CHAPTER 05**

**PROJECT PLAN**

**5.1 PROJECT ESTIMATE**

**5.1.1 Reconciled Estimates**

Cost Estimate

Line Of code(LOC)is 500(Appr.)

LOC in KLOC(Kiloes LOC)is 0.5 KLOC

Efforts

The Effort is Calculated by formula.

E=2.4\*(KLOC)1.05

E=2.4\*(0.5)1.05

E=1.26 Appr.

Time Estimates

The Development Time is Calculated by formula.

D=2.5\*(E)0.38

D=2.5\*(1.26)0.38

D=1.52Appr.

This does not take into account the time and effort it takes to perform data preprocessing and training.

**5.1.2 Project Resources**

1. Hardware Resources Required

Processor-Intel i5 core

Speed-1.6 GHz

RAM-8GB

Hard Disk-1TB

GPU

Key Board-Standard Windows Keyboard

Mouse-Two or Three Button Mouse

Monitor-SVGA

2. Software Resources Required

Operating system: Windows.

Coding Language: Python3

Database: File System

IDE: Jupyter Notebook

**5.2 RISK MANAGEMENT**

The identification of risk is central to the success and failure of the project, hence concentrated effort has been made to minimize and even eliminate certain risk related.

Software risk could be classified into categories. Internal and External risk, those risks

which arise from the risk factor within the organization can be defined as internal risk and the risk coming from outside is called external risk. Internal risk avoidance can be done by clear picturing of the process, product risk.

**5.2.1 Risk Identification**

For risks identification, review of scope document, requirements specifications and schedule is done. Answers to questionnaire revealed some risks. Please refer table for all the risks. You can refereed following risk identification questionnaire. Top soft-ware and customer managers formally committed to support the project. End- users enthusiastically committed to the project and the system/product to be built. All requirements fully understood by the software engineering team and its customers. The customers been involved fully in the definition of requirements. Do end-users have realistic expectations? The software engineering team have the right mix of skills. All project requirements are stable. All customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built.

**5.2.2 Risk Analysis**

The risks for the Project can be analyzed within the constraints of time and quality.

Table 5.1: Risk Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Risk description | Probability | Impact | | |
| Schedule | Quality | Overall |
| 1 | Increase in complexity of modules | Low | Medium | Low | Low |
| 2 | Inadequacy of hardware/monetary resources | High | Low | Medium | High |
| 3 | Errors related to dependencies | Medium | High | Low | Low |
| 4 | Wrong operation being performed on the input equation | Low | Low | Medium | Low |
| 5 | Input equation or output equation length might be out of bounds | Medium | Low | Medium | Low |

Table 5.2: Risk Probability definitions

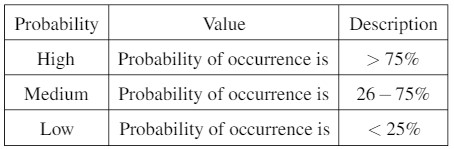


Table 5.3: Risk Analysis 1

|  |  |
| --- | --- |
| Risk Id | 2 |
| Risk Description | Inadequacy of hardware/monetary resources |
| Category | Requirement |
| Source | This was identified during the implementation and testing phase |
| Probability | High |
| Impact | High |
| Response | Accept |
| Strategy | Build the system and train the model on less data and then increase the training data in the future when more hardware is made available |
| Risk Status | Occurred |

Table 5.4: Risk Analysis 2

|  |  |
| --- | --- |
| Risk Id | 3 |
| Risk Description | Errors related to dependencies |
| Category | Implementation |
| Source | This was identified during the implementation and testing phase |
| Probability | Medium |
| Impact | Low |
| Response | Accept |
| Strategy | Perform proper testing on the different modules and analyze whether they interact with each other |
| Risk Status | Mitigated |

Table 5.5: Risk Analysis 3

|  |  |
| --- | --- |
| Risk Id | 4 |
| Risk Description | Wrong operation being performed on the input equation |
| Category | Implementation |
| Source | This was identified during the implementation and testing phase |
| Probability | Low |
| Impact | Low |
| Response | Accept |
| Strategy | Perform proper testing on how the system processes the user input and output equations |
| Risk Status | Mitigated |

Table 5.6: Risk Analysis 4

|  |  |
| --- | --- |
| Risk Id | 5 |
| Risk Description | Input equation or output equation length might be out of bounds |
| Category | Requirement |
| Source | This was identified during early development and testing |
| Probability | Medium |
| Impact | Low |
| Response | Accept |
| Strategy | Properly gather information on how long equations should be and apply proper error handling |
| Risk Status | Mitigated |

**5.2.3 Overview of Risk Mitigation, Monitoring, Management**

The identification of Risk is central to the success and failure of the project, hence concentrated effort has been made to minimize and even eliminate certain risks. Software risk could be classified into categories. Internal and External risk, those risk which arise from the risk factor within the organization can be defined internal risk and the risk coming from outside is called external risk. Internal risk avoidance can be done by clear picturing the process.

|  |  |
| --- | --- |
| Risk Id | 1 |
| Risk Description | Increase in complexity of modules |
| Category | Requirement |
| Source | This was identified during early development and testing |
| Probability | Low |
| Impact | Low |
| Response | Mitigated |
| Strategy | Proper planning of modules can avoid this |
| Risk Status | Mitigated |

Table 5.7 Risk Mitigation

**5.3 PROJECT SCHEDULE**

**5.3.1 Project task set**

Major Tasks in the Project stages are:

Task 1: Requirement Analysis (Base Paper Explanation).

Task 2: Project Specification (Paper Work).

Task 3: Technology Study and Design.

Task 4: Coding and Implementation (Module Development).

**5.3.2 Task Network**

Individual tasks and sub-tasks have inter-dependencies based on their sequence. A task-network is a graphic representation of the task overview for a project. Project tasks and their dependencies are noted.

**5.3.3 Timeline Chart**

Table 5.8: Timeline chart

|  |  |  |
| --- | --- | --- |
| **Sr No** | **Task** | **Scheduled Date** |
| **1** | Domain selection | 2 July 2020 |
| 2 | Topic selection | 18 July 2020 |
| 3 | Literature survey, Requirement Analysis | 24 July 2020 |
| 4 | Design modules | 15 August 2020 |
| 5 | Refined Project Scope | 5 August 2020 |
| 6 | UML Diagram | 2 September 2020 |
| 7 | Report submission | 15 September 2020 |
| 8 | Basic ML, dataset selection | 5 October 2020 |
| 9 | Data cleaning | 15 January 2021 |
| 10 | Modelling | 23 April 2021 |
| 11 | Integration and testing | 2 May 2021 |
| 12 | Final Report Submission | 12 June 2021 |

**5.4 TEAM ORGANIZATION:**

**5.4.1 Team structure**

Member 1:

1. Communication and Literature survey for other math related systems

2. Develop SRS

3. Planning project schedules

4. Plan the project

Member 2:

1. Design Mathematical model.

2. Design machine learning model

3. Feasibility Analysis

4. Design System Architecture

Member 3:

1. Plan the project

2. Design machine learning model

3. Write Deployment code

4. Build and test basic functional unit

Member 4:

1. Design risk management plan

2. Build and test security features

3. Design UML and other diagrams

4. Design test plan

**5.4.2 Management reporting and communication**

Table 5.9 Management reporting and communication

|  |  |  |
| --- | --- | --- |
| Sr No. | Work Description | Month |
| 1 | Discussion with guide regarding domain selection, selection of base paper for domain | July |
| 2 | Literature review regarding selected base paper domain | August |
| 3 | Requirement gathering and analysis, designing models | September |
| 4 | Report Preparation, stage 1 report submission | November |
| 5 | Searching for datasets and selecting the appropriate one | December |
| 6 | Searching for appropriate ML responses for found dataset | January |
| 7 | Coding and implementation | April |
| 8 | Validation and testing | May |
| 9 | Report preparation, stage 2 report submission | June |

**CHAPTER 06**

**PROJECT IMPLEMENTATION**

**6.1 OVERVIEW OF PROJECT MODULES**

1. Data Processing Module.

2. Machine learning Module.

3. Database module.

4. User Interface module

**6.2 TOOLS AND TECHNOLOGIES USED**

**6.2.1 Technology Description**

In the Python programming language for this system, all source code is written either Jupyter notebooks that end with .ipynb extension or in a text file saved with .py extension. Python is a robust Programming that is used when the time to market of a product is to be minimized. Python has a vast collection of libraries that can be used for a broad spectrum of applications like machine learning, image processing.

**6.2.2 Hardware Specifications**

Processor: I5

Speed: 3.8 GHz

RAM: 8GB

Hard Disk: 1 T.B.

Keyboard: Standard Windows Keyboard

**6.2.3 Software specifications**

Operating system: Windows 10

Programming Language: Python 3

IDE: Jupyter Notebook, Sublime Text

**6.3 ALGORITHM DETAILS**

**6.3.1 Encoder-Decoder**

Encoder decoder is an algorithm which is usually used in language translation but has been used for mathematical application in this system.

1. In this algorithm, there are 2 different models called encoder and decoder.
2. Input is taken from the user and processed so that it can be entered into the system.
3. The encoder takes the entire input and encodes it into a matrix.
4. The decoder takes the output of this encoder and a starting token and generates the first token of the output. This token is then passed to the decoder in the next step along with the encoder output and so on.
5. This way, the output is generated token-by-token.
6. When the ending token is generated or maximum length is reached, the output is processed back to readable form and displayed to user.

**CHAPTER 07**

**SOFTWARE TESTING**

**7.1 TYPES OF TESTING**

**7.1.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code ow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a speci c business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**7.1.2 Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or elds. Integration tests demonstrate that al-though the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**7.1.3 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process ows; data elds, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**7.1.4 System Test**

System testing ensures that the entire integrated software system meets require-ments. It tests a configuration to ensure known and predictable results. An exam-ple of system testing is the configuration oriented system integration test. System testing is based on process descriptions, emphasizing pre-driven process links and integration points.

**7.1.5 White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**7.1.6 Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. You cannot “see" into it. The test provides inputs and responds to outputs without considering how the software works.

**7.1.7 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**7.1.8 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or one step up software applications at the company level interact without error.

**7.1.9 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires signi cant participation by the end user. Test Results: All the test cases mentioned above passed successfully. No defects encountered.

**7.2 TEST CASES AND TEST RESULTS**

Table 7.1: Test case 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Purpose |  | Description | Expected results |  |  |
|  |  | |  |  |  |  |
| 1 | Check if all the training data is being appropriately preprocessed | | Compare the number of lines in raw data to the number of lines in preprocessed data | All the data is appropriately preprocessed | |  |
|  |  |  |  |  |  |  |

Table 7.2: Test case 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| ID | Purpose |  | Description | Expected results |  |  |
|  |  | |  |  |  |  |
| 2 | Check if all the modules are properly implemented | | Debug the code | All modules are implemented properly |  |  |
|  |  |  |  |  |  |  |

Table 7.3: Test case 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| ID | Purpose |  | Description | Expected results |  |  |
|  |  | |  |  |  |  |
| 3 | Check if input and output of equations works without error | | Test the system by giving input and output | System can take input and print output | |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |

Table 7.4: Test case 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| ID | Purpose |  | Description | Expected results |  |
|  |  | |  |  |  |
| 4 | Check if the proper operation is being performed on the equation | | Test the system by giving input and output | System can take input and print output | |
|  |  |  |  |  |  |

**CHAPTER 08**

**RESULTS**

**8.1 OUTCOMES**

The system was implemented as per requirements and even though it is incapable of actually giving correct solutions due to being undertrained, it is a complete system capable of taking input and giving output. All the modules are designed and developed successfully and the entire system runs without error.

**8.2 SCREENSHOTS:**

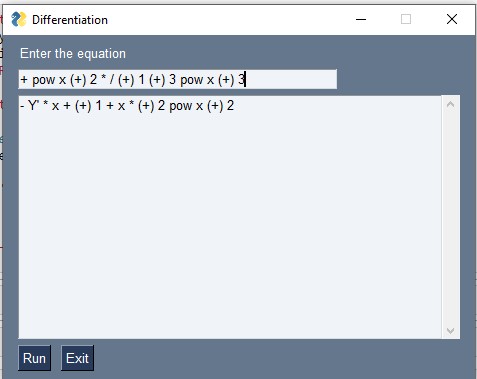


Fig 8.1: User Interface

In figure 8.1, it is a screenshot of the user interface of the system. It shows how the user can enter input and get output.

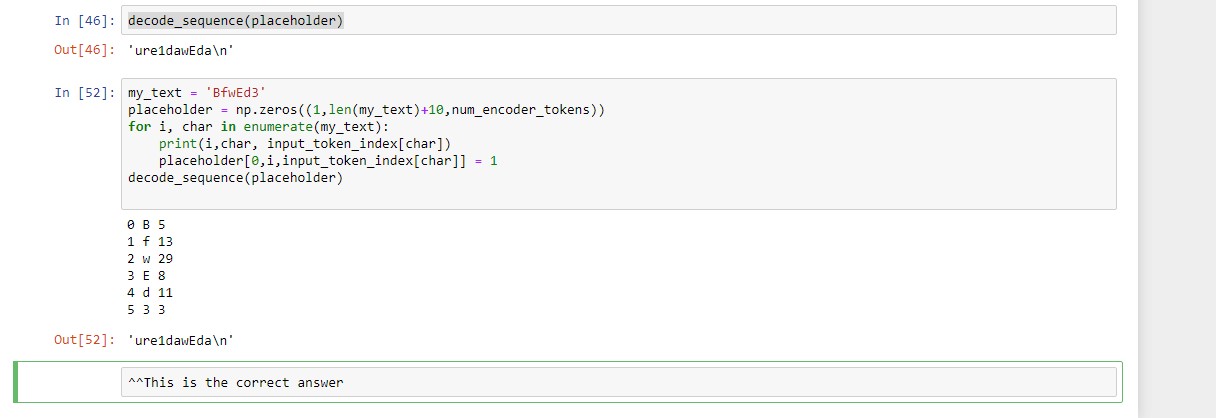


Fig 8.2 System without data processing

In fig 8.2, the screenshot shows an example of what kind of data the system works on and what that data will roughly look like without any data processing. The user should not have to handle data in this form.

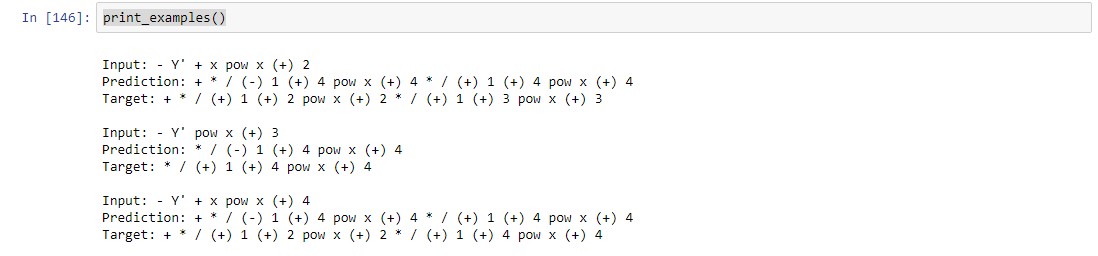


Fig 8.3 Example inputs, predictions and targets

In fig 8.3, the screenshot shows some examples of inputs, the prediction of a model trained previously on those inputs and the target output. This model provided proper output and different outputs on different inputs, however was deleted due to a hardware failure.

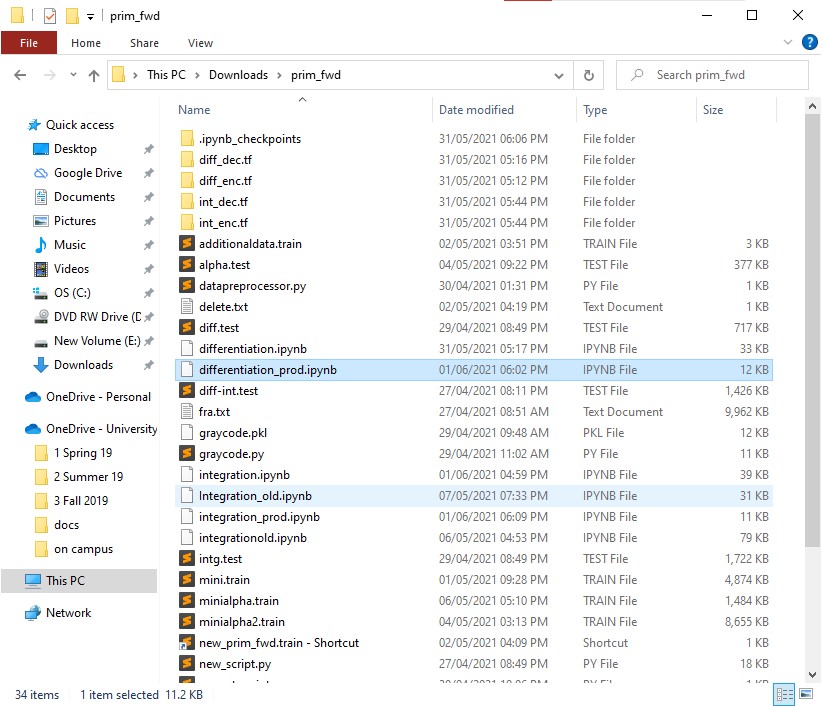


Fig 8.4 Folder view

Figure 8.3 shows how the folder of the system files looks like. Many of these files were created during the development of the system and are not necessary in use. However, they are important artifacts of what the development process of this system looked like.

**CHAPTER 09**

**CONCLUSIONS**

* 1. **CONCLUSION**

Functional differentiation and integration are challenging problems and new solutions should be applied to such problems. One such solution was developed for this problem with a new approach. This solution faced challenges due to lack of proper hardware, however the system and all modules were implemented in their entirety.

**9.2 FUTURE WORK**

In the future, this system can be trained on a lot of data using professional level hardware and resources to make it usable. For this, outside funding and support would be very beneficial, if not essential. When professional hardware is made available, this system will be trained on the entire dataset that is required and the model’s accuracy will rise drastically. Major computer algebra systems can implement this in their own systems to increase their speed and performance. Maybe, if the accuracy of this system increases high enough, it might even replace computer algebra systems as a standalone product.

**9.3 APPLICATIONS**

The applications for this project range from a wide variety of domains which include mathematics, research, chemistry, physics, aviation, architecture, construction, education, computing, artificial intelligence and many more.

**APPENDIX A**

NP is a complexity class used to classify decision problems. NP is the set of decision problems for which the problem instances, where the answer is "yes", have proofs verifiable in polynomial time. In computational complexity theory, an NP-complete decision problem is one belonging to both the NP and the NP-hard complexity classes. NP stands for "nondeterministic polynomial time". NP is a complexity class used to classify decision problems. NP is the set of decision problems for which the problem instances, where the answer is "yes", have proofs verifiable in polynomial time. In computational complexity theory, an NP-complete decision problem is one belonging to both the NP and the NP-hard complexity classes. NP stands for "nondeterministic polynomial time". NP-hardness (non-deterministic polynomial time hardness), is the defining property of a class of problems that are, informally, "at least as hard as the hardest problems in NP". The complexity class P is the set of all decision problems that can be solved with worst-case polynomial time-complexity. "N" in "NP" refers to the fact that you are not bound by the normal way a computer works, which is step-by-step. This means that you are dealing with an amazing kind of computer that can run things simultaneously or could somehow guess the right way to do things, or something like that. So this "N" computer can solve lot more problems in "P" time - for example it can just clone copies of itself when needed. So, programs that takes dramatically longer as the problem gets harder (i.e. not in "P") could be solved quickly on this amazing "N" computer and so are in "NP".

Thus "NP" means "we can solve it in polynomial time if we can break the normal rules of step-by-step computing".

**APPENDIX B**

Details of Paper Publication:

Name of the journal/conference: ICINC-2021 - 6th International Conference on Internet of Things, Next Generation Networks and Cloud Computing.

Comments of reviewers: Received the award for “Best Paper”, reviewer said the concept is fresh and creative, advised to train on more data if/when sufficient hardware becomes available.

Paper Abstract:

Mathematics is used in almost every walk of life and in every field, including aviation, chemistry, education, artificial intelligence and many more. Computer algebra systems have been used to solve complex mathematical computations ever since the first computer algebra system Macsyma was released in 1978. Future researchers used several options to solve these problems by trying different representations of problems and by using modern technologies like Artificial Intelligence. This is not an easy problem to solve, for both humans and computers. This paper provides a brief overview of these techniques and methods for tackling various such mathematical problems.

**APPENDIX C**

****

**REFERENCES**

1. Guillaume Lample, François Charton, “Deep Learning for Symbolic Mathematics”, arXiv:1912.01412, 2 December 2019
2. Miltiadis Allamanis, Pankajan Chanthirasegaran, Pushmeet Kohli, and Charles Sutton. Learning continuous semantic representations of symbolic expressions. In Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML’17, pp. 80–88. JMLR.org, 2017.
3. Forough Arabshahi, Sameer Singh, and Animashree Anandkumar. Combining symbolic expressions and black-box function evaluations for training neural programs. In International Conference on Learning Representations, 2018a.
4. Forough Arabshahi, Sameer Singh, and Animashree Anandkumar. Towards solving differential equations through neural programming. 2018b.
5. D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations (ICLR), 2015.
6. M. Bronstein. Symbolic Integration I: Transcendental Functions. Algorithms and combinatorics. Springer, 2005. ISBN 978-3-540-21493-9.
7. Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. Recurrent neural network grammars. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 199–209, 2016.
8. Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. Learning to parse and translate improves neural machine translation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 72–78, 2017.
9. Philippe Flajolet and Andrew M. Odlyzko. Singularity analysis of generating functions. SIAM J. Discrete Math., 3(2):216–240, 1990.
10. Philippe Flajolet and Robert Sedgewick. Analytic Combinatorics. Cambridge University Press, New York, NY, USA, 1 edition, 2009. ISBN 0521898064, 9780521898065.