KANTIPUR ENGINEERING COLLEGE

(Affiliated to Tribhuvan University)

Dhapakhel, Lalitpur

[Subject Code: CT755]

A MAJOR PROJECT PROPOSAL ON

NETWORK INTRUSION DETECTION SYSTEM USING DEEP NEURAL NETWORK

Submitted by:

Aman Devkota [KAN076BCT010]

Ankur Karmacharya [KAN076BCT013]

Prashad Adhikary [KAN076BCT056]

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

June, 2023

NETWORK INTRUSION DETECTION SYSTEM USING DEEP NEURAL NETWORK

Submitted by:

Aman Devkota [KAN076BCT010]

Ankur Karmacharya [KAN076BCT013]

Prashad Adhikary [KAN076BCT056]

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

Kantipur Engineering College

Dhapakhel, Lalitpur

June, 2023

TABLE OF CONTENTS

List of Figures iii

List of Abbreviations iv

1 Introduction 1

1.1 Background . . . 1

1.2 Problem Statement . . . 2

1.3 Objectives . . . 2

1.4 Project Features . . . 3

1.5 Application Scope . . . 3

1.6 System Requirement . . . 3

1.6.1 Development Requirements . . . 3

1.6.1.1 Software Requirements . . . 3

1.6.1.2 Hardware Requirements . . . 3

1.6.2 Deployment Requirements . . . 4

1.6.2.1 Software Requirements . . . 4

1.6.2.2 Hardware Requirements . . . 4

1.7 Project Feasibility . . . 4

1.7.1 Technical Feasibility . . . 4

1.7.2 Operational Feasibility . . . 4

1.7.3 Economic Feasibility . . . 4

1.7.4 Schedule Feasibility . . . 4

2 Literature Review 5

2.1 Related Projects . . . 5

2.1.1 Suricata . . . 5

2.1.2 Snort . . . 5

2.2 Related Works . . . 5

3 Methodology 6

3.1 Working Mechanism . . . 6

3.1.1 Data set . . . 7

3.1.2 Data Preprocessing . . . 8

3.1.3 Feature Selection . . . 8

3.1.4 Train and test the model . . . 8

3.1.4.1 Long Short term Memory (LSTM) 8
3.1.5 Evaluation Metrics 9
3.2 System Diagram 10
3.2.1 Use case diagram 10
3.2.2 Software Development Model 11
4 Epilogue **13**
4.1 Expected Output 13
References **13**

LIST OF FIGURES

1.1 Gantt Chart 4
3.1 Working mechanism of Network Intrusion Detection System 6
3.2 Features of NSL-KDD dataset 7
3.3 LSTM 9
3.4 Confusion Matrix 10
3.5 Use case Diagram of Stock Price Prediction System 11
3.6 Incremental Model 12

LIST OF ABBREVIATIONS

ATR: Average Time Range

CBIR: Commercial Bank Interest Rate

EMA: Exponential Moving Average

ER: Exchange Rate

GRU: Gated Recurrent Unit

IR: Inflation Rate

LSTM: Long Short Term Memory

MACD: Moving Average Convergence Divergence

MAPE: Mean Absolute Percentage Error

MFI: Money Flow Index

NEPSE: Nepal Stock Exchange

RSI: Relative Strength Index

TRB: Treasury Bills

VRNN: Vanilla Recurrent Neural Network

CHAPTER 1
INTRODUCTION

1.1 Background

With the increasingly deep integration of the Internet and society, the Internet is changing the way in which people live, study and work, but the various security threats that we face are becoming more and more serious. How to identify various network attacks, especially unforeseen attacks, is an unavoidable key technical issue. An Intrusion Detection System (IDS), a significant research achievement in the information security field, can identify an invasion, which could be an ongoing invasion or an intrusion that has already occurred. In fact, intrusion detection is usually equivalent to a classification problem, such as a binary or a multiclass classification problem, i.e., identifying whether network traffic behavior is normal or anomalous, or a five-category classification problem, i.e., identifying whether it is normal or any one of the other four attack types: Denial of Service (DOS), User to Root (U2R), Probe (Probing) and Root to Local (R2L). In short, the main motivation of intrusion detection is to improve the accuracy of classifiers in effectively identifying the intrusive behavior. Machine learning methodologies have been widely used in identifying various types of attacks, and a machine learning approach can help the network administrator take the corresponding measures for preventing intrusions. However, most of the traditional machine learning methodologies belong to shallow learning and often emphasize feature engineering and selection; they cannot effectively solve the massive intrusion data classification problem that arises in the face of a real network application environment. With the dynamic growth of data sets, multiple classification tasks will lead to decreased accuracy. In addition, shallow learning is unsuited to intelligent analysis and the forecasting requirements of high-dimensional learning with massive data. In contrast, deep learners have the potential to extract better representations from the data to create much better models. As a result, intrusion detection technology has experienced rapid development after falling into a relatively slow period [1]. A well-known method of securing the network is through implementing an intrusion detection system (IDS). IDS was originally implemented in 1980 . The main aim of their work was to introduce a mechanism which differentiates between benign activities from malicious ones. Further research was carried out to optimizing this methodology to aid monitoring the network traffic in case

of attacks, this system is now known as Network Intrusion Detection System (NIDS) . In NIDS, the detection system is inspecting the incoming and outgoing network traffic from all hosts in real time and based on certain criteria, it can detect and identify the attack, then, take the suitable security measures to stop or block it, which significantly reduces the risk of damage to the network. However, due to the rapid increase in the complexity of the cyber-security attacks, the current methods used in NIDS are failing to sufficiently address this issue[2]. IDSs can be divided into two categories according to the main detection technology: misuse detection and anomaly detection. Misuse detection is a knowledge-based detection technology. A misuse detection system needs to clearly define the features of the intrusion, then identify the intrusion by matching the rules. Misuse detection can achieve a high accuracy and low false alarm rate. However, it needs to build a feature library and cannot detect unknown attacks. In contrast, anomaly detection is a behavior-based detection technology. First, it needs to define the normal activities of a network, and then check whether the actual behavior has deviated from the normal activities. Anomaly detection needs only to define a normal state of a specific network, without prior knowledge of intrusion. Thus, it can detect unknown attacks, although there may be a high false alarm rate. At present, network structure is becoming more and more complicated, and intrusion methods are following the trend of diversification and complication, creating more challenges for IDSs. The recurrent neural network (RNN) has failed to become a mainstream network model in the past few years due to difficulties in training and computational complexity. In recent years, with the development of deep learning theory, RNN began to enter a rapid development period. Currently, RNN has already been applied successfully to handwriting [5] and speech recognition [6]. The main feature of RNN is that it circulates information in a hidden layer which can remember information processed previously, leading to a structural advantage for the processing of time series information. Correspondingly, many intrusion behaviors can be abstracted as specific time series of events from the underlying network. So, RNN is considered suitable for building an IDS[3]

1.2 Problem Statement

1.3 Objectives

The primary objective of this project:

i. To develop a robust and accurate system that can effectively detect and notify network intrusions or malicious activities within a computer network.

1.4 Project Features

The project will be able to accomplish following:

- Anomaly Detection
- Real-time Monitoring
- Traffic Preprocessing

1.5 Application Scope

Network Intrusion Detection System has various applications in areas such as Enterprise networks, Internet Service Providers(ISPs), Cloud Computing environment, Government networks and so on. The application scope of network intrusion detection systems using deep neural networks extends beyond these areas, as network security is crucial in nearly every sector that relies on secure and reliable communication. By deploying these systems, organizations can proactively detect and respond to network intrusions, minimize the impact of attacks, and protect their assets, data, and operations from potential threats.

1.6 System Requirement

1.6.1 Development Requirements

1.6.1.1 Software Requirements

- Windows/Linux/Mac
- HTML/CSS/JS
- Jupyter Notebook
- Python IDE

1.6.1.2 Hardware Requirements

- PC with at least 4-8 GB RAM
- Higher graphics of at least 2 GB

1.6.2 Deployment Requirements

1.6.2.1 Software Requirements

- Web browser
- Visual studio code
- Pycharm

1.6.2.2 Hardware Requirements

- More than 1.5 GHz clock speed
- Minimum 4 GB RAM

1.7 Project Feasibility

1.7.1 Technical Feasibility

The technical feasibility assessment is focused on gaining in understanding of the present technical resources required by the system and their applicability to the expected needs of the proposed system. Regarding the proposed system, the technical requirement includes a PC.

1.7.2 Operational Feasibility

The user will not need any formal knowledge about programming so our project is operationally feasible.

1.7.3 Economic Feasibility

The purpose of the economic feasibility assessment is to determine the positive economic benefits to the user that the proposed system will provide. Most of the software used for the development is free. Thus, the project is economically feasible.

1.7.4 Schedule Feasibility

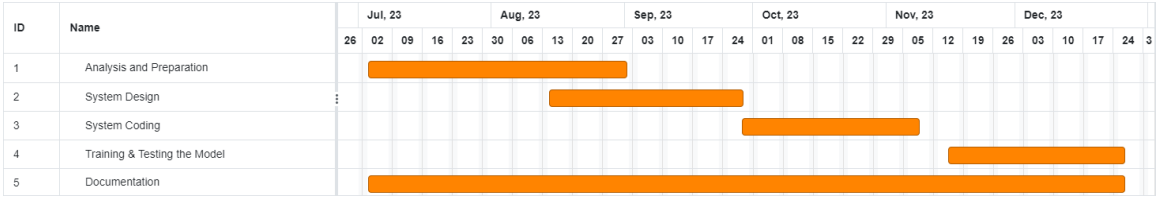

Figure 1.1: Gantt Chart

CHAPTER 2
LITERATURE REVIEW

2.1 Related Projects

2.1.1 Suricata

Suricata is a high performance, open source network analysis and threat detection software used by most private and public organizations, and embedded by major vendors to protect their assets. It was developed by the Open Information Security Foundation (OSIF) and is a free tool used by enterprises, small and large. The system uses a rule set and signature language to detect and prevent threats. Suricata can run on Windows, Mac, Unix and Linux. As we know intrusion detection "detects" and "alerts" a threat. In contrast, an intrusion prevention system also takes action on the event and attempts to block the traffic. Suricata can do both and also does well with deep packet inspection, making it perfect for pretty much any kind of standard security monitoring initiatives for a company.

2.1.2 Snort

Snort is the foremost Open Source Intrusion Prevention System (IPS) in the world. Snort IPS uses a series of rules that help define malicious network activity and uses those rules to find packets that match against them and generates alerts for users. Snort can be deployed inline to stop these packets, as well. Snort has three primary uses: As a packet sniffer like tcpdump, as a packet logger- which is useful for network traffic debugging, or it can be used as a full-blown network intrusion prevention system. Snort can be downloaded and configured for personal and business use alike.

2.2 Related Works

CHAPTER 3
METHODOLOGY

3.1 Working Mechanism

The development of Network Intrusion Detection System involves major steps which is depicted in the diagram given below:

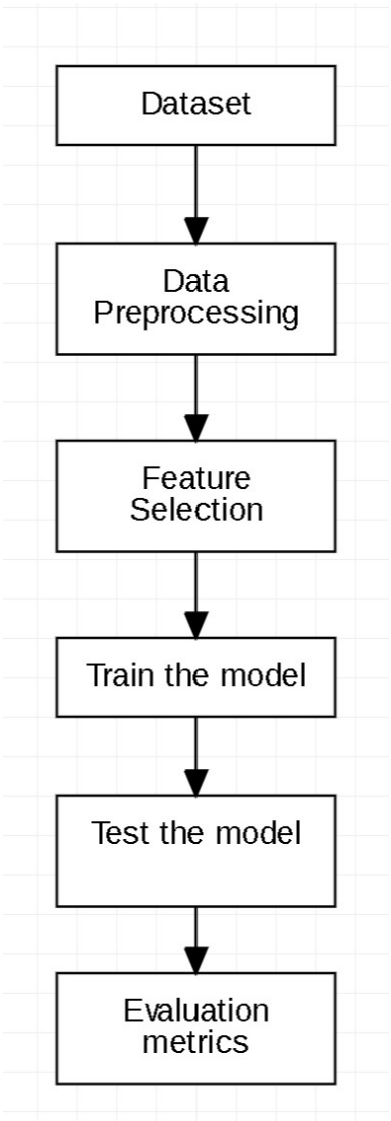

Figure 3.1: Working mechanism of Network Intrusion Detection System

3.1.1 Data set

The NSL-KDD dataset generated in 2009 is widely used in intrusion detection experiments. In the latest literature , all the researchers use the NSL-KDD as the benchmark dataset, which not only effectively solves the inherent redundant records problems of the KDD Cup 1999 dataset but also makes the number of records reasonable in the training set and testing set, in such a way that the classifier does not favour more frequent records. There are 41 features and 1 class label for every traffic record, and the features include basic features (No.1- No.10), content features (No.11 - No.22), and traffic features (No.23 - No.41). According to their characteristics, attacks in the dataset are categorized into four attack types: DoS (Denial of Service attacks), R2L (Root to Local attacks), U2R (User to Root attack), and Probe (Probing attacks). The testing set has some specific attack types that disappear in the training set, which allows it to provide a more realistic theoretical basis for intrusion detection.

No.	Features	Types	No.	Features	Types
1	duration	Continuous	22	is_guest_login	Symbolic
2	protocol_type	Symbolic	23	count	Continuous
3	service	Symbolic	24	srv_count	Continuous
4	flag	Symbolic	25	serror_rate	Continuous
5	src_bytes	Continuous	26	srv_serror_rate	Continuous
6	dst_bytes	Continuous	27	rerror_rate	Continuous
7	land	Symbolic	28	srv_rerror_rate	Continuous
8	wrong_fragment	Continuous	29	same_srv_rate	Continuous
9	urgent	Continuous	30	diff_srv_rate	Continuous
10	hot	Continuous	31	srv_diff_host_rate	Continuous
11	num_failed_logins	Continuous	32	dst_host_count	Continuous
12	logged_in	Symbolic	33	dst_host_srv_count	Continuous
13	num_compromised	Continuous	34	dst_host_same_srv_rate	Continuous
14	root_shell	Continuous	35	dst_host_diff_srv_rate	Continuous
15	su_attempted	Continuous	36	dst_host_same_src_port_ra	Continuous
16	num_root	Continuous	37	dst_host_srv_diff_host_rat	Continuous
17	num_file_creations	Continuous	38	dst_host_serror_rate	Continuous
18	num_shells	Continuous	39	dst_host_srv_serror_rate	Continuous
19	num_access_files	Continuous	40	dst_host_rerror_rate	Continuous
20	num_outbound_cmds	Continuous	41	dst_host_srv_rerror_rate	Continuous
21	is_host_login	Symbolic			

Figure 3.2: Features of NSL-KDD dataset

3.1.2 Data Preprocessing

There are 38 numeric features and 3 non-numeric features in the NSL-KDD dataset.Because the input should be a numeric matrix, we must convert some non-numeric features, such as 'protocol-type', 'service' and 'flag' features, into numeric form using one hot encoding. For the efficient training of neural networks, input data should be transformed by performing some pre-processing known as data normalization. It is used where inputs are widely divergent. Without such a process, networks would take a long time to train. Different schemes can be used to normalise the input data before it is fed to the input layer of neural network. We have used Min-Max normalization to normalize the attributes of our dataset, because of the fact that it preserves relationships between features. Mathematically,

$$x_i = \frac{x_i - Min}{Max - Min} \tag{3.1}$$

3.1.3 Feature Selection

As mentioned before, the benchmark dataset contains 41 features and 1 output class attribute. After applying Information Gain, Gain Ratio and Correlation based feature reduction algorithms, it has been found that some of the features such as num file creations, num outbound cmds, dst host count, is host login, dst host rerror rate, su attempted and num access files do not play significant role in detection of malicious traffic from normal patterns. More over some other features in NSL-KDD datasets have been zeroed which can also reduce the performance of IDS.

3.1.4 Train and test the model

The model used in this project for NIDS is: Long Shortterm Memory (LSTM)

3.1.4.1 Long Short term Memory (LSTM)

LSTM is a popular deep learning technique in RNN for time series prediction. While standard RNNs outperform traditional networks in preserving information, they are not very effective in learning long term dependencies due to the vanishing gradient problem. An LSTM is well-suited to classify and/or predict time-series data. There are several

architectures of LSTM units. A common architecture is composed of a memory cell, an input gate, an output gate and a forget gate. The mathematical formulation of the LSTM cell is given below:

$$f_t = \sigma(x_t W_f + H_{t-1} U_f) \tag{3.2}$$

$$o_t = \sigma(x_t W_o + H_{t-1} U_o) \tag{3.3}$$

$$S_t = \sigma(S_{t-1} * f_t + i_t * H_t^{'}) \tag{3.4}$$

$$i_t = \sigma(x_t W_i + H_{t-1} U_i) \tag{3.5}$$

$$H_t^{'} = tanh(x_t W_g + H_{t-1} U_g) \tag{3.6}$$

$$H_t = tanh(S_t) * o_t \tag{3.7}$$

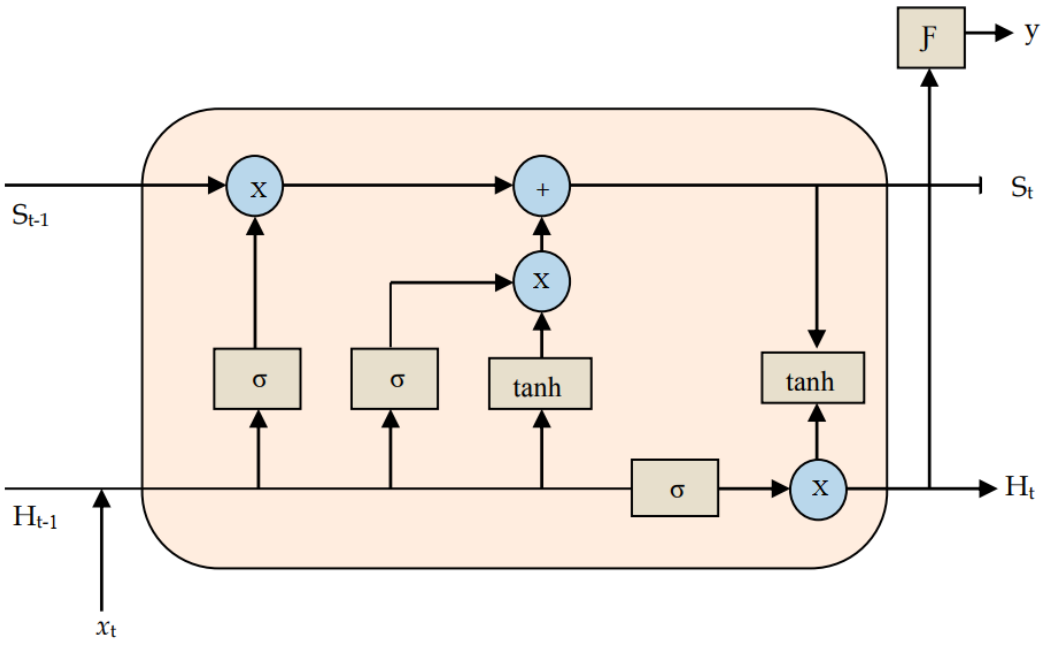

Figure 3.3: LSTM

3.1.5 Evaluation Metrics

The most important performance indicator (Accuracy, AC) of intrusion detection is used to measure the performance of the model. In addition to the accuracy, we introduce the detection rate and false positive rate. The True Positive (TP) is equivalent to those correctly rejected, and it denotes the number of anomaly records that are identified as

anomaly. The False Positive (FP) is the equivalent of incorrectly rejected, and it denotes the number of normal records that are identified as anomaly. The True Negative (TN) is equivalent to those correctly admitted, and it denotes the number of normal records that are identified as normal. The False Negative (FN) is equivalent to those incorrectly admitted, and it denotes the number of anomaly records that are identified as normal.

Actual	Predicted	
	Attack	Normal
Attack	TP	FN
Normal	FP	TN

Figure 3.4: Confusion Matrix

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.8}$$

Precision is the number of actual attacks as a proportion of the number classified as attacks.

$$Precision = \frac{TP}{TP + FP} \tag{3.9}$$

True Positive Rate shows the percentage of the number of records identified correctly over the total number of anomaly records.

$$TruePositiveRate = \frac{TP}{FN + TP} \tag{3.10}$$

False Positive Rate is the percentage of the number of records rejected incorrectly is divided by the total number of normal records.

$$FalsePositiveRate = \frac{FP}{FP + TN} \tag{3.11}$$

3.2 System Diagram

3.2.1 Use case diagram

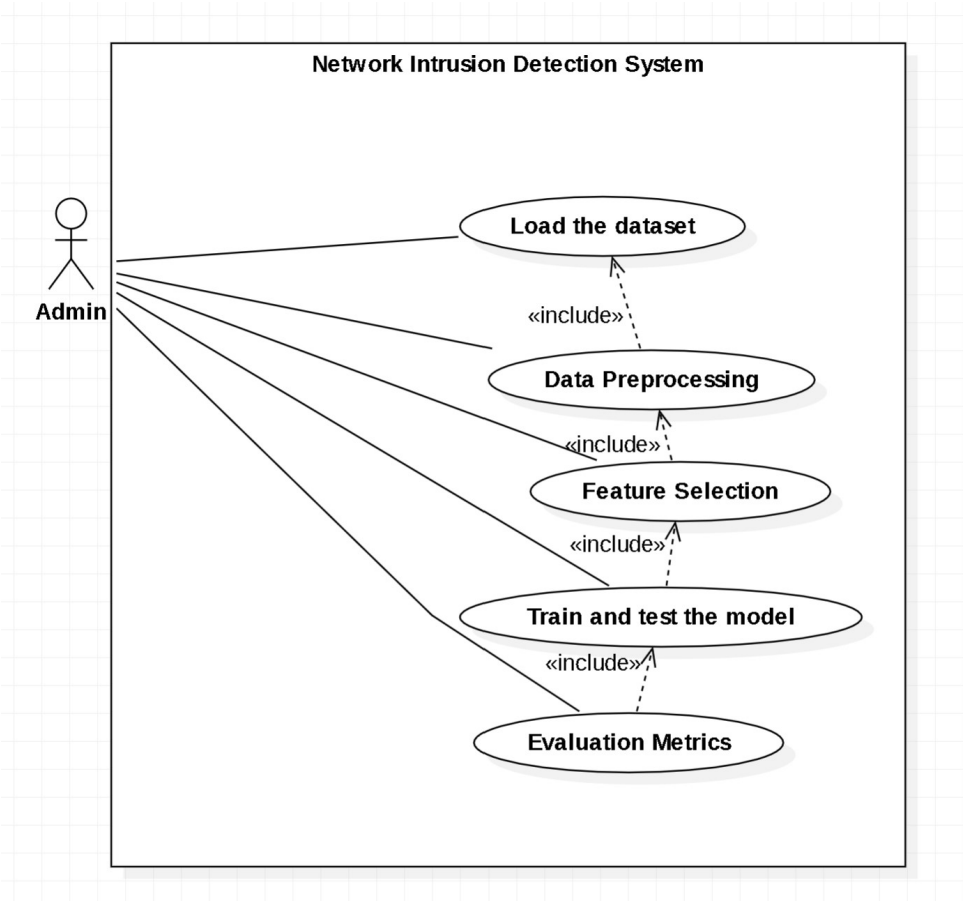

Figure 3.5: Use case Diagram of Network Intrusion Detection System

3.2.2 Software Development Model

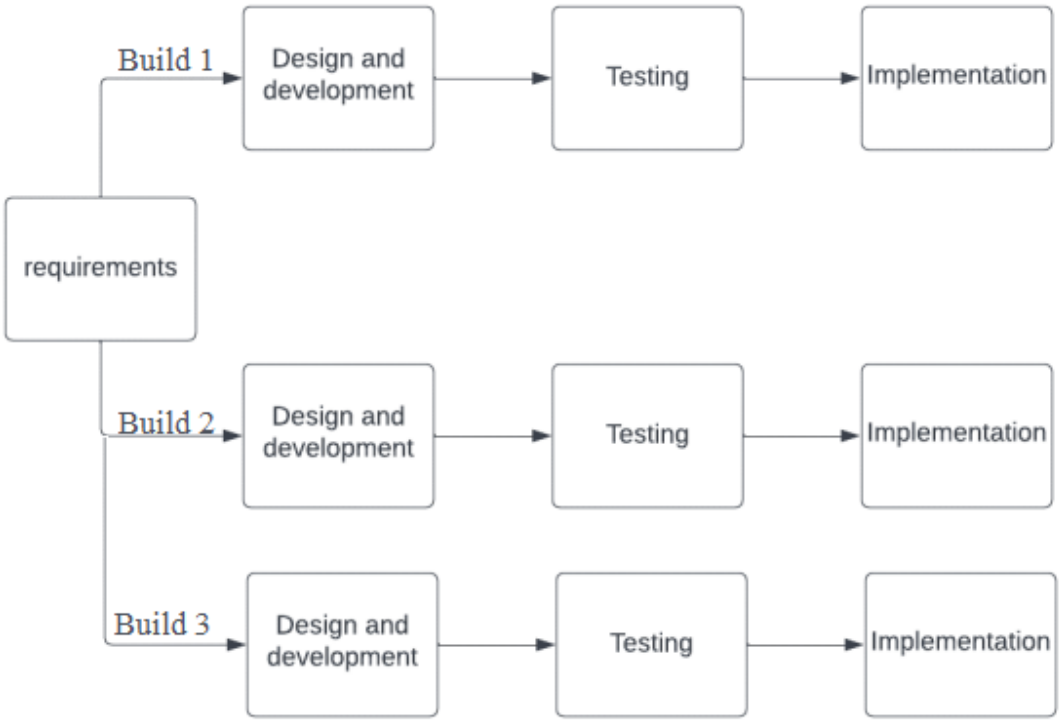

Figure 3.6: Incremental Model

Incremental model is a method of software engineering that combines the elements of waterfall model in iterative manner. It involves both development and maintenance. In this model requirements are broken down into multiple modules. Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance. Each iteration passes through the requirements, design, coding and testing phases. The first increment is often a core product where the necessary requirements are addressed, and the extra features are added in the next increments. The core product is delivered to the client. Once the core product is analyzed by the client, there is plan development for the next increment.

CHAPTER 4
EPILOGUE

4.1 Expected Output

Upon the completion of the project, the system will be able to predict the price of the stock the user selects.

REFERENCES

[1] A. S. Saud and S. Shakya, "Analysis of look back period for stock price prediction with rnn variants: A case study on banking sector of nepse," *Procedia Computer Science*, vol. 167, pp. 788–798, 2020.

[2] ——, "Analysis of gradient descent optimization techniques with gated recurrent unit for stock price prediction: A case study on banking sector of nepal stock exchange," *Journal of Institute of Science and Technology*, vol. 24, no. 2, pp. 17–21, 2019.

[3] N. R. Pokhrel, K. R. Dahal, R. Rimal, H. N. Bhandari, R. K. Khatri, B. Rimal, and W. E. Hahn, "Predicting nepse index price using deep learning models," *Machine Learning with Applications*, vol. 9, p. 100385, 2022.