DETECTING SLQ INJECTION

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT FOR THE REQUIREMENT OF THE AWARD OF DEGREE OF

BACHELOR OF SCIENCE
(INFORMATION TECHNOLOGY)

Submitted by

AYESHA YUSUFALI SIDDIQUI
A-119

Under the esteemed guidance of
Prof. Minal Prashant Suryavanshi
Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY
Malini Kishor Sanghvi College of Commerce and Economics
(Affiliated to University of Mumbai)
MUMBAI, 400049
MAHARASHTRA
2022-2023

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PNR No : D20CQ0017408 Roll No : A- 119

1. Name of the Student : AYESHA SIDDIQUI
2. Title of the Project : **Detecting SQL Injection**
3. Name of the Guide : Ms. MINAL SURYAVANSHI
4. Teaching experience of the Guide :
5. Is this your first submission ? Yes ☑ No ☐

Signature of the Student Signature of the Guide

Date: _____ Date:_____

Signature of the Coordinator

Date: _____

MALINI KISHOR SANGHVI COLLEGE OF COMMERCE AND ECONOMIC

(Affiliated to University of Mumbai)
CITY-MAHARASHTRA-400049

DEPARTMENT OF INFORMATION TECHNOLOGY

College Logo

CERTIFICATE

This is to certify that the project entitled, **Detecting SQL Injection** , is boneheaded work of **AYESHA SIDDIQUI** bearing Seat.No: **A-119** submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide **Coordinator**

External Examiner

Date: **College Seal**

Abstract

All of us are surrounded by technology. So much information and millions of files are being shared all across the Internet over web applications. Online payments and Internet banking have also become so common recently. Web-based applications store crucial information from users in databases. The database in the backend is integrated with web front ends, which allows injection attacks to be performed. SQL injection means placing harmful code in the original code by inputting malicious SQL statements. Therefore, testing SQL vulnerabilities is important, but at the same time, it is practically impossible to check everything without using a proper algorithm. This paper attempts to detect SQL attacks using basic Machine Learning algorithms and to improve the performance stacking technique was used in which one model was chosen as meta model - Logistic Regression and different combination of basic algorithms (Logistic regression, k-nearest-neighbor, formed the base models. The reason for using these basic models is to highlight that to improve the performance matrix we don't necessarily need deep learning models which require large datasets and high computational power.

In our day-to-day, the usage of web applications has become the most common thing as a part of daily activities. Now-a-days people are very interested in online shopping, social networking, financial transactions, etc...in which they prefer online services rather than in-person services. Making the web application available to everyone makes it more vulnerable. These vulnerabilities enable hackers to steal confidential information from the web applications. Out of those vulnerabilities one of them is SQL Injection(Structured Query Language).So with this type of attack, the hackers can obtain or change the data present in the database by injecting malicious code. This paper mainly focuses on the types of attacks that can be done and the prevention of SQL using Message Digest 5 (MD5) cryptographic hash algorithm. The user's passwords have to be stored in hashed format in the database.

ACKNOWLEDGEMENT

No project is ever complete without the guidance of those experts who have already traded this past before and hence become master of it and as a result, our leader. So I would like to take this opportunity to take all those individuals who have helped us in visualizing this project.

We express our deep gratitude to our project guide Mrs.Minal Suryavanshi for providing timely assistance to my query and guidance that she gave owing to her experience in this field for the past many years. She had indeed been a lighthouse for us on this journey.

We extend our sincere appreciation to our entire Professor from Malini Kishor Sanghvi College for their valuable inside and tip during the designing of the project. Their contributions have been valuable in so many ways that we find it difficult to acknowledge them individually.

Thanking You,

Ayesha Siddiqui

DECLARATION

I here by declare that the project entitled, **Detecting SQL Injection** done at **Malini Kishor Sanghvi College**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Ayesha Siddiqui

Detecting SQL Injection

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATIONS

1. SQL- Structured Query Language
2. DBMS - Database Management System
3. AES- Advance Encryption Standard
4. SQLIA- SQL Injection Attack
5. XML- Extensive Markup Language
6. HTTP- Hypertext Transfer Protocol
7. GUI- Graphical User Interface
8. SQLX- SQL Extension
9. NLTK- Natural Language Toolkit
10. POS- Part of Speech
11. SVM- Support Vector Machine
12. SVC- Support Vector Classifier
13. TF-IDF - Term frequency-Inverse document frequency
14. MD5 Algorithm - Message-digest algorithm
15. PERT -Project (or Program) Evaluation and Review Technique
16. SDLC - The spiral model is a systems development lifecycle
17. DFD- Data Flow Diagram
18. UML- Unified Modelling Language
19. EPD-Event driven Procedural Diagram
20. DSD- Data Structure Diagram
21. ERD- Entity Relation Diagram
22. TCP/IP- Transmission Control Protocol /Internet Protocol
23. DAST- Dynamic Application Security Testing
24. Cryptography :is the practice and study of techniques for secure communication in the presence of adversarial behavior. More generally, cryptography is about constructing and analyzing protocols that prevent third parties or the public from reading private messages

25. Buffer Overflow- a buffer overflow, or buffer overrun, is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory locations.
26. True Positive Rate (TPR): True Positive/positive
27. False Positive Rate(FPR): False Positive /Negative
28. False Negative Rate(FNR): False Negative/Positive
29. True Negative Rate(TNR): True Negative/Negative
30. DAST : Dynamic Application Security Testing
31. SAST: Static Application Security Testing
32. SDLC : Software Development Lifecycle
33. JSA : Java String Analyzer
34. NDFA : Non-deterministic Finite Automaton
35. HADOOP : Hadoop is not a type of database, but rather a software ecosystem that allows for massively parallel computing. It is an enabler of certain types NoSQL distributed databases (such as HBase), which can allow for data to be spread across thousands of servers with little reduction in performance.

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

In recent years, the inclusion of internet, in almost every form of commercial enterprise transaction resulted in exponential growth within the dependency in the direction of information technology. The data generation, used by the overall hundreds, for reasons inclusive of commercial transactions, educational transactions and other endless activities related to finance. The use of the net to perform vital ordinary jobs such as transactions of balance, transactions of stability from bank bills, constantly stays below safety chances. Many software program structures have advanced to include an internet-primarily based component, one of them is SQL injection, that may give attackers unrestricted ingress to the databases that underlie web applications and has grown to be more recurrent and serious. SQL injection is an attack that arises within the database layer of an application. The vulnerability is present when user detail is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input isn't strongly typed. SQL Injection is one of the ubiquitous application layer attacks practiced these days. Many surveys have addressed this hassle, also some researchers have proposed specific procedures to come across and prevent this vulnerability however they may be no longer a hit completely. Furthermore, some of these approaches have not applied but users might be pressured in selecting the best device.

Enterprise-scale applications over the Internet often have a need to authenticate a large number of users, to retrieve specific information from a large warehouse of data, or to save information for later retrieval by an application or an analyst. The amount of data is so large, that it cannot be accomplished in-memory.

Therefore, these Internet applications must use a database in the form of a database management system (DBMS). Internet applications often take data obtained from web pages and then combine them directly into an SQL statement to be directly and dynamically parsed, and 10 then processed by the DBMS. This allows powerful and effective queries to be written quickly and easily. In turn, these incorporated queries do jobs such as log in authentication,

and data storage and retrieval. Flexibility is another strength of SQL's.

Similar to a true programming language, SQL allows the inclusion of inline comments within the code, including between statements. SQL allows pattern and regular expression matching of strings. SQL also enables users to concatenate and combine separate characters or values (such as from a column or field of a record) to form a complete string. As it will be shown, it is the flexibility of SQL syntax which primarily makes signature-based detection challenging. SQL also has a UNION construct which allows the rows from different tables to be selected simultaneously, as long as the columns are compatible types.This is a powerful feature that opens up a new dimension in table selection and enables more complicated aggregation. As with all power in the wrong hands, the UNION can be used to do things beyond the intention of the database application designers and engineers. A person may make use of all of these features and functionality in a way that is unauthorized and dangerous by using SQL-Injection attacks.

Researchers have proposed a different technique to provide a solution for SQL, but many of these solutions have limitations that affect their effectiveness and practically. Researchers have indicated that the solution to these types of attacks may be based on defense coding practice. But it's not efficient because of three reasons.

- First, it is difficult to bring out a rigorous defensive coding discipline.
- Second, many solutions based on defensive coding address only a subset of the possible attacks.
- Third, legacy software poses a particularly difficult problem because of the cost and complexity of retrofitting existing code so that it is compliant with defensive coding practices. In this work, an attempt has been made to increase the efficiency of the above techniques by a combinational approach for protecting web applications against SQL Injection attacks.

Even the SQL Injection attacks can bypass the security mechanisms such as Firewall, cryptography and traditional Intrusion detection systems. If the trend of providing web-based services continues, the prevalence of SQL Vs is likely to increase access to the database that underlie Web applications and the most worrying aspect of SQL Injection attack are; it is very easy to perform, even if the developers of the application are well known about this type of attack. The basic idea behind this attack is that the malicious user counterfeits the data that a web application sends to the database aiming at the modification of the SQL Query that will be executed by the DBMS software.

Why was this topic chosen?

Different types of SQL injection attacks exist today which enables the attacker to inject malicious data into an SQL statement. Through this, they perform malicious tasks such as retrieving or modifying application data, subverting application logic, fingerprinting a database, etc. Over the years, organizations have implemented several stringent security measures for SQL injection mitigation, the attackers have still been one step ahead.

The impact of SQL injection attacks can be far-reaching and may vary depending on the target. Data integrity and confidentiality breach are the most common consequences though. An SQL injection can be used to reveal confidential information or cause modification to the database. It can also permanently delete data, access the OS Shell, and execute commands on the system.

However, the potential cost of an SQL injection attack is even higher. It may gravely impact the credibility of an organization. It can also result in the loss of customers' trust as their personal information such as credit card details, phone number, passwords, etc. can be stolen. Not to mention, the further damage that customers can face due to the information theft.

An SQL injection is an input validation vulnerability. Although quite common, SQL injection attacks can be contained. SQL injection vulnerabilities are often a consequence of developers creating dynamic database queries with user-supplied inputs, without validating. By choosing to stop writing queries that are dynamic in nature or using validated user-supplied input that may contain malicious data, developers can avoid an injection attack.

Finally, using prepared statements with parameterized queries to write database queries will force the developer to define the SQL code and then pass each of the parameters to the query. This helps the database in differentiating between code and data. It also ensures that an attacker cannot change the intent of a query by the insertion of SQL commands.

Similarly, measures such as whitelisting the input validations, frequent scanning and audits, authentication mechanisms, use of latest development platforms, and enforcing least privileges can further prevent the attacks.

1.2 OBJECTIVES

The objective of this project is to develop a secure path for transactions done by the user. Using AES (Advanced Encryption Standard) encryption technique, the transaction and user account details can be made secured. This system is online so no need for implementation. It can be accessed through the net from anywhere. The system uses AES encryption to encrypt the user's card and password information while transaction.

AES encryption is a cybersecurity technology refers to the process of concealing electronic data using an approved 128-bit, 192-bit, or 256-bit symmetric encryption algorithm from the Advanced Encryption Standard, also known as FIPS 197. The AES is a computer security standard for cryptographic securing electronic information, usually secret and top-secret government information

1.3 PURPOSE, SCOPE AND APPLICABILITY

1.3.1 Purpose

This Technique is used to detect and prevent SQLIA's with runtime monitoring. The solution insights behind the technique are that for each application, when the login page is redirected to our checking page, it was to detect and prevent SQL Injection attacks without stopping legitimate accesses. It is a hacking technique in which the attacker adds SQL statements through a web application's input fields or hidden parameters to gain access to resources or make changes to data. The fear of SQL injection attacks has become increasingly frequent and serious. This proposed technique is Random4 and Hirschberg Algorithm is used.

Today's modern web era expects the organization to concentrate more on web application security. This is the major challenge faced by all the organizations to protect their precious data against malicious access or corruption. Generally, the program developers show keen interest in developing the application with usability rather than incorporating security policy rules. Input validation issue is a security issue if an attacker finds that an application makes unfounded assumptions about the type, length, format, or range of input data. The attacker can then supply a malicious input that compromises the application. When a network and host level entry points are fully secured; the public interfaces exposed by an application become the only source of attack the cross-site scripting attacks, SQL Injections attacks and Buffer Overflow are the major threat in the web application Security through this input validation security issues.

Our main aim is to provide increased security by developing a tool which prevents illegal access to the database. The project aims to prevent SQL injection while performing a query. It does so by implementing a secure and online method to store and protect all the sensitive data stored in the database.

The purpose of this project is to provide a safe transaction for the users. Both the transaction and the user data can be encrypted using the AES encryption technique. This system encrypts the user's login details to preserve privacy of the website's clients. identifying trusted data sources and marking data coming from these sources as trusted, using dynamic tainting to track trusted data at runtime, and allowing only trusted data to form the relevant parts of queries such as SQL keywords and operators.

1.3.2 Scope

This system can be used in shops. This system can be used to sell like chains of clothing shops from a single site Secured transaction while doing card payment. Less risk of data getting hacked. Use of Standardized AES algorithm for data security. The system is very secure and robust in nature. SQL Injection prevention mechanism is used. The proposed technique is used to detect and prevent SQLA's during runtime.

The solution insights behind the technique are that for each application, when the login page is redirected to the next page, it was to detect and prevent SQL Injection attacks by only allowing authorized users. So, the proposed algorithm is Message Digest Algorithm in which the passwords are stored in hashed format in the database. So that no one knows what the exact password is and it is safe for users also as data is securely stored. SQL injection attacks are older attacks but they are still increasing these days and they are dangerous. It provides the Security Analyst with all the necessary security issues and its solution to prevent hackers.

It provides the users with all the necessary privileges to access and modify the data intended for them. It doesn't entirely replace the existing system but it mostly automates the Scanning process and all the data used. This system is online so there is no need for implementation. It can be accessed through the internet from anywhere. The system uses AES encryption to encrypt the user's card and password information while transaction. The system can be modified for detection and prevention of XML attacks and attacks on web services. Can be extended for detection and prevention of more attacks.

We depend on database-driven web applications for an ever-increasing number of activities, such as banking and shopping. When performing such activities, we entrust our personal information to these web applications and their underlying databases. The confidentiality and integrity of this information is far from guaranteed; web applications are often vulnerable to attacks, which can give an attacker complete access to the application's underlying database. SQL injection is a type of code-injection attack in which an attacker uses specially crafted inputs to trick the database into executing attacker-specified database commands.

1.3.3 Applicability

We depend on database-driven web applications for an ever-increasing number of activities, such as banking and shopping. When performing such activities, we entrust our personal information to these web applications and their underlying databases. The confidentiality and integrity of this information is far from guaranteed; web applications are often vulnerable to attacks, which can give an attacker complete access to the application's underlying database. SQL injection is a type of code-injection attack in which an attacker uses specially crafted inputs to trick the database into executing attacker-specified database commands.

1.4 ACHIEVEMENTS

As future work, we want to evaluate methods using different web-based application scripts with public domain to achieve great accuracy in SQL injection prevention approaches. Integrate SQLiX with nikto HTTP scanner, HTTP scanning proxies, and with Metasploit will help to detect other web vulnerabilities. Also add features to dump vulnerable databases and database schema.

It is used for Safe Authentication. Reduce the time and space complexity. More Secure online transactions are possible. The major issue of web application security is the SQL Injection, which can give the attackers unrestricted access to the database that underlie Web applications. Highly automated approach for protecting Web applications from SQLAs. Provides DBMS auditing methods to find out the transactions. Does not require customized and complex runtime environments. Requires no modification of the runtime system.

Three major challenging enhancements that I have completed successfully in this project

1. Enhanced the crawler to handle HTTP post methods and fill forms automatically.
2. Created Graphical User Interface (GUI) for SQLX.
3. Added a Module to Detect Cross Site Scripting (XSS) attacks.

1.5 ORGANISATION OF REPORT

In Chapter 1, I have described about our project and the background and Objectives in 1.1 and 1.2. 1.3 Includes three parts in which I have discussed the purpose scope and applicability from 1.3.1 through 1.3.3.

In Chapter 2, I have discussed about the survey of technology and tools used and its working which is compatible with this project. I have also discussed the various papers that we have referred to for our project. This section includes the title of the papers, along with their description and the pros and cons of those projects. Here, I have also mentioned the ways by which propose to overcome all the disadvantages of the projects that have been described in the paper. This chapter also includes the technological review of this project.

In Chapter 3, the requirement analysis of my project has been discussed. This includes the operating system that we are working on, the hardware, software, front end and the back-end requirement of our project to execute successfully. A complete planning and scheduling of this project with problem definition, preliminary product description and conceptual models have been discussed.

Chapter 4, is based on system design. This includes all the design approaches that include the front-end design, component diagram, deployment diagram, E-R diagram and the flow graph of this project. This will also include the basic modules and designs used. This is related to the implementation details of this project. This includes the assumptions that I have taken into consideration while designing our project and also the dependencies, describing the modular of the project. The use case report and the class-diagram report has been explained respectively.

Chapter 5, includes coding details, test cases of the way the project has been tested, detailed information of the types of testing in a software management life cycle including the testing approach used for this particular project with its future modification and improvement.

Chapter 6, contains the results of the working assembled project after testing it and discussions of end- user usage manual and documentation along with test reports of each component's expected results after testing. The purpose of this documentation is to guide the users on how to properly install, use, and/or troubleshoot a product and prevent the user directly reaching the customer care.

Chapter 7, has the conclusion, significance of the project, why and how this project system will benefit the future, limitations of the systems and future scope of the system about when can the changes take place adapting the future requirements and how can the system be compatible with improved features considering the current limitations.

CHAPTER 2

SURVEY OF TECHNOLOGIES

2.1 EXISTING SYSTEM

2.1.1 System working flow:

The process of designing a functional classifier for sentiment analysis can be broken down into four basic categories. They are as follows:

1. Data Acquisition 2. Human Labeling 3. Feature Extraction 4. Classification

1. Data Acquisition: The datasets used in this project consist of simulated normal and malicious traffic originating from the traffic generation server and collected and correlated as mentioned above. For our results, the dataset is downloaded from the kaggle website and has 2 columns such as SQL query and classification label. This dataset is unique because it captures features typical of log analysis, such as the SQL statement that results from a web application request, but also captures and correlates features such as response length and result that are returned from the database server to the web. application. To our knowledge, this is the first project using such a dataset. This dataset, as a combination of the previous two datasets, combines features from both, and in our experiments, machine learning algorithms are able to use features from both of these datasets.

2. Human Labeling: We labeled the SQL queries in two classes according to SQL syntax : Sql Injection, Not Sql Injection.

- Sql Injection : If the user entered value belongs to the sql query then the model is treated as a sql injection because by using that query a third person can retrieve all data which is harmful.
- Not Sql Injection : If the user entered a normal value then the model is treated as a non-sql injection.

3. Feature Extraction:

Now that we have arrived at our training set we need to extract useful features from it which can be used in the process of classification. But first we will discuss some text formatting techniques which will aid us in feature extraction:

• Tokenization: It is the process of breaking a stream of text up into words, symbols and other meaningful elements called “tokens”. Tokens can be separated by whitespace characters and/or punctuation characters. It is done so that we can look at tokens as individual components that make up a sql query.

• URLs and user references (identified by tokens “http” and “@”) are removed if we are interested in only analyzing the text of the sql query.

• Punctuation marks and digits/numerals may be removed if for example we wish to compare the sql query to a list of English words.

• Lowercase Conversion: sql query may be normalized by converting it to lowercase which makes its comparison with an English dictionary easier.

• Stemming: It is the text normalizing process of reducing a derived word to its root or stem. For example, a stemmer would reduce the phrases “stemmer”, “stemmed”, “stemming” to the root word “stem”. Advantage of stemming is that it makes comparison between words simpler, as we do not need to deal with complex grammatical transformations of the word. In our case we employed the algorithm of “porter stemming” on both the sql query and the dictionary, whenever there was a need for comparison.

• Stop-words removal: Stop words are a class of some extremely common words which hold no additional information when used in a text and are thus claimed to be useless. Examples include “a”, “an”, “the”, “he”, “she”, “by”, “on”, etc. It is sometimes convenient to remove these words because they hold no additional information since they are used almost equally in all classes of text, for example when important words in a sql query according to their frequency of occurrence in different classes and using this polarity to calculate the sql query over the set of words used in that sql query.

2.1.2 Environment

The whole program was written in Machine learning. We've twisted it to utilize the NLTK AI library joined with WordNet 3.0 for text examination and extraction. NLTK was utilized to separate a neighborhood of discourse labels, and WordNet was utilized for determining the lemma of a given word. We've twisted the utilization, straightforwardly from the AI code, for running the classifiers on the training data. We've adapted to require a look at totally changed choices settings with the three classifiers and in this way the outcomes are reportable and broken down inside the ensuing segment.

4. Classification:

The classification approach generally followed in this domain is a two-step approach. First Objectivity Classification is done which deals with classifying a sql query or a phrase as either objective or subjective. After this we perform Polarity Classification (only on a sql query classified as subjective by the objectivity classification) to determine whether the sql query is Sql Injection or Not Sql Injection. This was presented by Wilson et al. and reports enhanced accuracy over a simple one-step approach.

The following Machine Learning algorithms for this second classification to arrive at the best result:

• Support Vector Machine

• Naive Bayes

The characterization approach regularly followed during this area could likewise be a two-venture approach. First perspicacity Classification is finished that manages characterizing a sql query or an expression as one or the other unbiased or emotional. When this happens, we have adapted to perform Polarity Classification to find out whether the sql query is Sql Injection or Not Sql Injection. This was given by Wilson et al. furthermore, reports have expanded precision than a simple one-venture approach.

2.2 PROPOSED SYSTEMS

Normal Language Toolkit:

The NLTK is a stage for building python projects to work with text information. It gives a choice of corpora and assets and different libraries for text arrangement, labeling, stemming, tokenization and parsing. During this task, NLTK was utilized broadly for (tokenizing the sql query), POS labeling, the tagger model being maxent treebank pos tagger, stemming (as outlined on top of, it utilized the Porter Stemmer of NLTK), and grouping. The NLTK classifiers utilized were the Naïve Bayes Classifier and the Maxent Classifier.

Pandas:

Pandas may be a code library composed for Python and is utilized for information investigation and manipulation6. Pandas is an open stock library and it conjointly interoperates with the Numpy and elective Python libraries.

Sci-unit Learn:

Scikit-Learn is an open inventory AI library for the Python programming language .It choices various grouping, relapse and pack calculations likewise as help vector machines, arrangement relapse, credulous mathematician, arbitrary backwoods, inclination boosting and k-implies, and should interoperate with the Python mathematical and logical libraries NumPy and SciPy. During this undertaking, Sci-unit Learn was mainly utilized for the SVM classifier, uncommonly, the Linear SVC.

Numeric features:

The numeric features that were employed during this experiment embrace a range sql injected words, emoticons, length of sql query and range of special characters like exclamations, hashtags then on. tons of details on a few of these options is given within the knowledge section. The numeric options didn't yield sensible accuracy and gave around sixty-three percent accuracy.

Data Analysis:

Data Analysis is the process of systematically applying statistical and/or logical techniques to explain and illustrate, condense and recap, and evaluate data. An important component of ensuring data integrity is the accurate and appropriate analysis of research findings. Information investigation is critical in business to comprehend issues confronting an organization and to investigate data altogether information in itself is simply raw numbers

information investigation puts together deciphers designs and presents the data into supportive data that has setting for the information investigation assists organizations with getting constant bits of knowledge concerning deals advertising money advancement and that's just the beginning it licenses bunches inside organizations to work together and are accessible through higher outcomes its useful for organizations to investigate past business execution and enhance future business measures data picture helps key chiefs in an incredibly business generally non-tech senior executives see investigation introduced outwardly in diagrams graphs and so forth so as that they will set up patterns and designs and see progressed information why learn it if you're innovative this could be the legitimate capacity to search out.

TF-IDF:

Term frequency-Inverse document frequency (tf-idf) might be a numeric value that tells how necessary a word is during a document. It is an awfully well-liked feature generation technique employed in data retrieval.

Term frequency: In document classification every term is allotted a weight, depending on the number of times it happens within the document. This will be mentioned because the term frequency is denoted by tfidf. The term weight doesn't believe the order of its incidence, however solely the frequency.

Inverse Document frequency: Let the document frequency , outlined because the range of documents within the assortment that contain the term t, be dft. If the general range of documents is N, then the inverse document frequency of the term. Tf-idf gets a high score once the term t seems to be in a smaller range of documents, however tons of oft. Its score is low once the term appears in most documents which could mean that the term cannot be used to distinguish the document. Tf-idf is incredibly well-liked in text classification. During this read, it was used as part of the choices during this project. However, tf-idf gave occasional accuracy. This could be attributed to the terribly incontrovertible fact that sql queries are of very short length and considering these as documents doesn't add any worth.

2.3 JUSTIFICATION OF PLATFORM

Naive Bayes:

The Scikit Learn library provides the naive_bayes module to perform classification using the naive bayes algorithm. Basically, there are two main types of Naive Bayes classifiers in this module:

MultinomialNB: This classifier is used when dealing with data sets whose properties are essentially count data (randomly discrete). variables), as in the case of the text classification.

GaussianNB: This classifier is used when dealing with datasets whose characteristics are continuous random variables (e.g. the characteristics of the breast cancer dataset). GaussianNB has its own version of probability smoothing called variance smoothing, which uses the same underlying logic as Laplace smoothing, But it is modified appropriately for continuous random variables. Naive Bayes models are probabilistic generative models. The generative modes are based on: The joint probabilities P(xi,cj) (where xi is a data point and cj is a class j corresponding to xi). Make predictions using Bayes' rule to calculate P(cj | xi). In essence, discriminative classifiers learn a mapping from the inputs X to the class labels y.Generative models learn the distribution of the training data and can thus: Predict the probabilities for each class given a data point (i.e. a set of features). It will also be able to simulate/generate the data from scratch. The Naive Bayes model uses the chain rule of probability to model the distribution of each class with respect to the properties of the data set. For this purpose, each feature is examined individually and the respective statistics are recorded.

Naive bayes is wide utilized ai classifier and probabilistic calculation essential uses of naive bayes region unit to channel spam order archives and so on the component feed into the model is independent of each unique that is renascent the value of any of the other component utilized inside the calculation naive bayes enjoys vital benefit is that we’ve an adapted to confront measure prepared to coded up to foresee the yield ongoing speedy it’s only climbable and old calculation is also a most reasonable alternative for planet applications that region unit needed to answer to client as by and as feasible. Lets take an example: you have a set of reviews and classifications

Figure 1 Naive Bayes

Sr. No	Text	Class
1	I loved the movie	+
2	I hated the movie	-
3	A great movie. Good movie	+
4	Poor acting	-
5	Great acting. A good movie	+

Above table defines movie review with sentiment data. On top of the table there's a text column that is input and there are 10 unique words that are: - "I, loved, the, movie, hated, a, great, poor, acting, good". Categories contain the sentiment information that's negative and positive. which define that movie review is negative or positive based on 10 unique words. Then we've got to convert the above data table into features and based on that we tend to get sentiment output. 1st we have to convert it into matrix type and also have to find how many times has that word come back.

Figure 2 Naive Bayes

Sr. No.	I	Loved	the	movie	hated	a	great	poor	acting	good	Class
1	1	1	1	1							+
2	1		1	1	1						-
3				2		1	1			1	+
4				1				1	1		-
5						1	1		1	1	+

In Figure 2 unique words are comeback that is "I" word is continual in initial and second review, then the word "Loved" word is repeated exactly in the initial review and so on. In the class column there are 5 categories that's mixture of positive and negative classes. In this there are 3 positive categories and a couple of negative categories. Then we've to count all the positive unique words. Then we've to calculate the likelihood against the positive category therefore the probability is 3/5. Then we compute p(I) before that there's a formula that we've to refer that's We add 1 in each probability thus the chance, like P(class | text) can never be zero. we are trying to determine if a data row should be classified as negative or positive. due to these we are able to ignore the divisor.

Therefore we have to calculate the probabilities of every classification and therefore the

probabilities of every feature falling into each classification. now we have to place values in our equation In P(I | +) nk 1, that is I occurred in the initial document just once. This method is comparable for negative text also. However, vocabulary count is constant in both cases. Now we have to train our classifier, for that there's one formula that is Vnb = argmax(summation of all the words occur in sentence) That is, suppose there's a sentence like "I hated the poor acting" then 1st we've to classify all words that is therein sentence and then we have to get the likelihood to get the class that is positive or negative. Thus we grab all the distinctive words and then place them in the above equation.

If Vj = +; p(+)P(I|+) P(I|+) P(I|+) P(I|+) P(I|+) = 6.03x10-7

If Vj = -; p(-)P(I|-) P(I|-) P(I|-) P(I|-) P(I|-) = 1.22x10-5

So get the probability of having this sentence in positive and negative. For positive classification we get 6.03x10-7 and for negative classification we get 1.22x10-5.now we have to determine whether or not a sentence is classed as positive or negative. so, the answer is negative. because of 10-5. In negative values the min negative range is greater than the most negative number so that sentence gets classified into the negative class. Suppose there's a maximum value that is bigger than -20 then we need to take a log of that number and then we have to compare both the values. The focal point of the undertaking is on the peril investigation of high danger citizens. The dataset utilized for our task is downloaded from a US web webpage that contains genuine character identified with the risk evaluation. then we deduct every one of the invalid qualities that may not be significant assuming we utilize those invalid qualities, It can anticipate wrong answers. In this manner we've to drop those invalid qualities. At that point distinctive commotion even needs to be deterred and feed unadulterated data to the calculations. Exploitation AI methodologies attempt to get a model that foresees if the client is in hazard or not, assuming that client is in hazard, that approach has the chance to be applied subsequently to the client. assuming that the client can't pass every one of the methodologies, which recommends he/she is in the high danger citizens list.

Advantages and Limitations of Naive Bayes:

The main advantage of the Naive Bayes model is its interpretability since it is a probabilistic model. Naive Bayes classifiers are quick to train because they basically just examine each feature in the dataset individually and compute class label statistics. They are not overly sensitive to parameter settings and generally perform well. They are suitable for large, high-dimensional datasets and work equally well with continuous and categorical data.Naive Bayesian models have slightly lower generalization performance than linear parametric models. Laplace smoothing turns out to be a general and necessary requirement when the probability of a feature in a class turns out to be zero. Naive Bayes models are not suitable for imbalanced data. Therefore, the training data must always be balanced before being used in the Naive Bayes model

SVM:

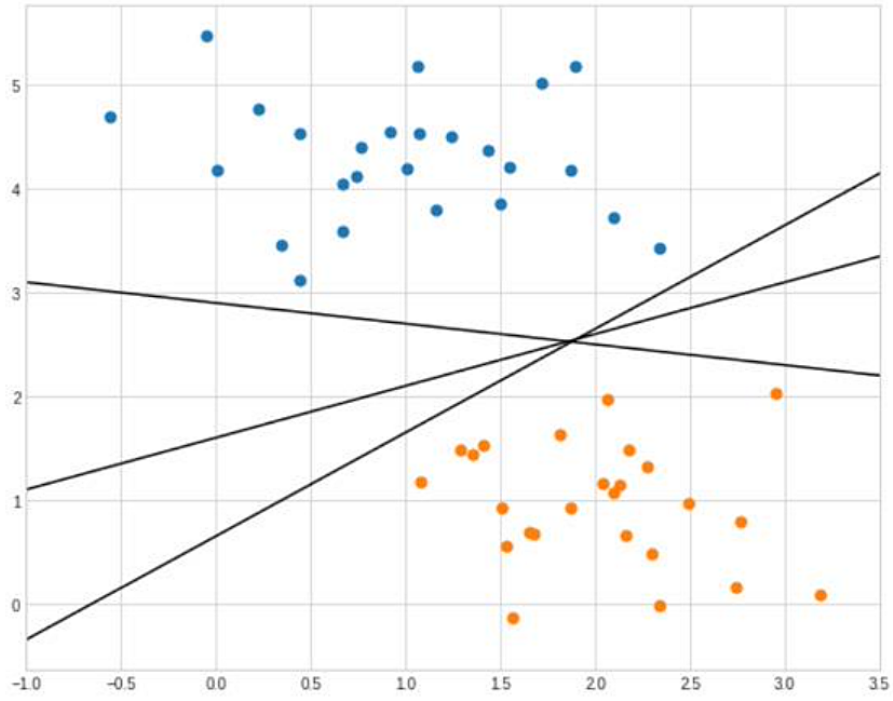
The Scikit Learn library provides three SVM implementations which we could use

1. The svm.LinearSVC class
2. The svm.SVC class and
3. The linear_model.SGDClassifier class.

The second implementation uses Dual Form optimization and has the capacity to perform the Kernel trick, whereas the other two are linear classifiers. The first classifier implements the Primal form using slack variables, whereas the third classifier implements the hinge loss form, using stochastic gradient descent for optimization.

Logistic Regression is the technique of classification using linear hyperplanes. Consider the two-dimensional dataset shown in the image below. As can be seen, there are multiple hyperplanes that can successfully classify the data points. Logistic regression chooses the hyperplane whose weight vector (w) produces the least logistic loss.

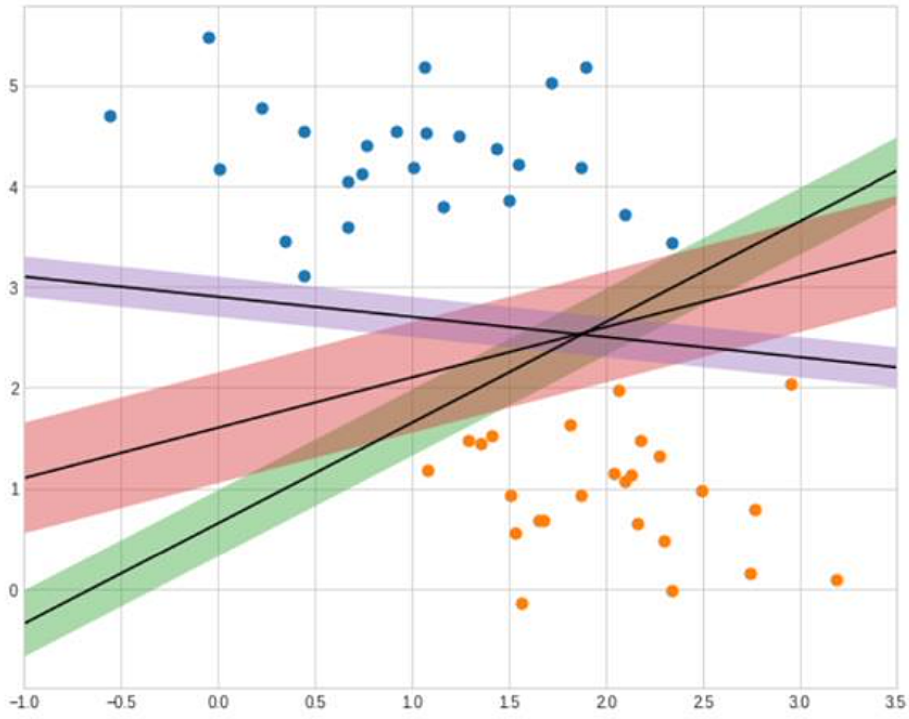
Figure 3



Another strategy for choosing the best hyperplane, is choosing that hyperplane that has the **maximum margin**. Consider the image shown below of the same dataset as earlier. Each of the hyperplanes now has an equi-spaced margin around them, the width of which is constrained by the positions of the closest data points to the hyperplane from the opposite classes. Now, the hyperplane with the largest margin can be considered as the best

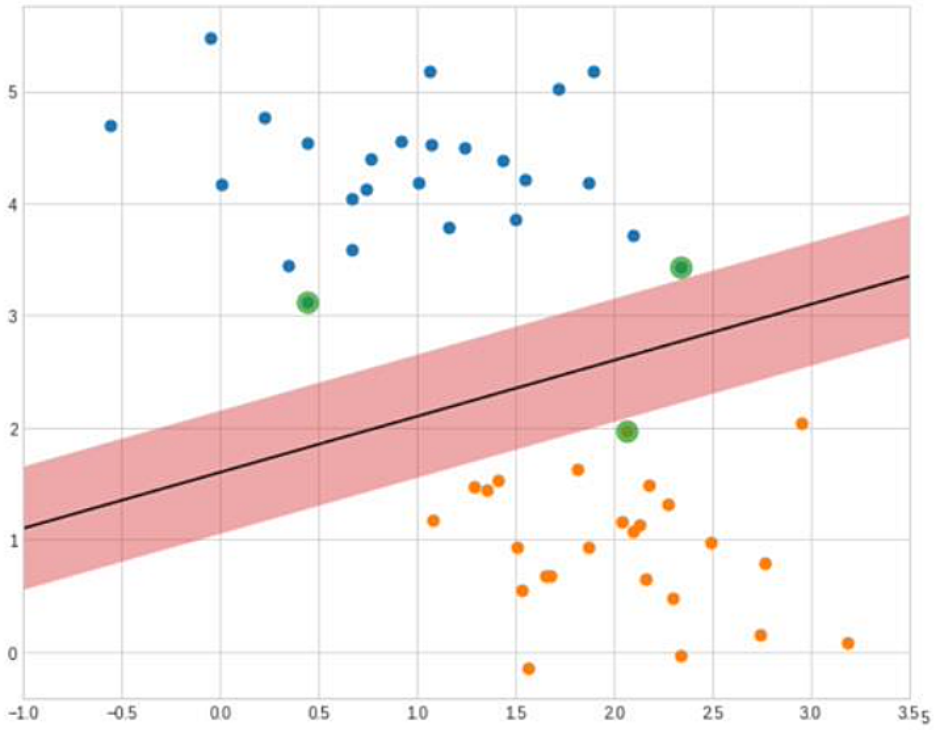
hyperplane. Support vector machine (SVM) is a classification algorithm that tries to find that weight vector (w) that corresponds to the hyperplane with the largest margin.

Figure 4



For a given hyperplane support vector are those data points that define the hyperplane margin. In other words, support vectors are those points from opposite classes that are nearest to the hyperplane being considered. In the image shown below, the points colored green are the closest points from opposite classes and they represent potential support vectors. The ones that produce the hyperplane with the largest margin are then chosen as the final support vectors. Once the support vectors are fixed we need not consider the rest of the data points for any classification purposes.

Figure 5



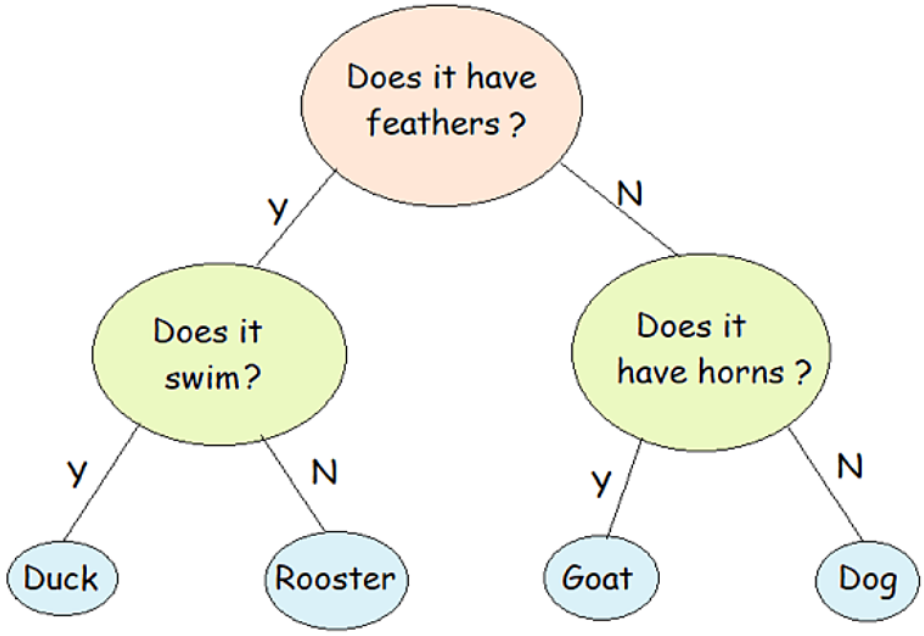
Advantages and Limitations of SVM:

Support vector machines have the capacity to work on non-linearly separable data due to the usage of the kernel trick. They work well on datasets that are high dimensional and even in situations where the number of data points is smaller than its dimension. The computational requirement of the dual form kernelized implementation is quadratic in nature, since it has to compute the dot product of all the data points with each other. Thus kernelized SVMs do not scale well and become infeasible for datasets containing more than 30 - 40 thousand samples. In case of large datasets it is more practical to use the stochastic gradient descent based linear SVMs.

Decision Tree:

Decision Trees are machine learning models that learn a hierarchy of if-else questions that eventually lead to a decision. Consider the hierarchy of questions shown in the image below. This hierarchy can be used to classify any group containing four types of animals: Dog, goat, rooster & duck.

Figure 6



Each question leads to a yes/no (binary) split. Each of the two groups formed is then recursively subjected to such critical if-else questions, until we eventually reach splits that completely classify the data. Such an hierarchical structure is called a Decision Tree. The questions/conditions used to split the data are called splitting conditions. The top most node representing all the data is called the root node. The subgroups formed after each split are called branch nodes of the decision tree and the last (or terminal) branches of the decision tree are called leaf nodes. The total number of "levels" in the tree's hierarchy is called its depth. In the example above, the data consists of three categorical features: Feathers, Swim and Horns (3D). Each data point (x_i) corresponds to a value/class (y_i) in the target variable/ (labels column), in this case: Duck, Rooster, Goat and Dog. Decision trees work by basically dividing the feature space using axes parallel separation boundaries as previously mentioned while discussing KD Trees in the chapter on KNNs. "Learning" in the case of Decision Trees essentially involves discovering the right splitting conditions for each branch at every level

(depth) of the tree's hierarchy. The strategies used for classification and regression to find the best splitting conditions are slightly different but are similar.

Decision Trees use the concept of Entropy for choosing the best splitting conditions.eEntropy as discussed in the chapter on Multi class Logistic regression, is basically a measure of the amount of randomness (lack of information/predictability) present within a random variable. Higher the randomness, the greater the entropy. Given a random variable y containing k classes/states. The process for finding the splitting condition for multidimensional data remains the same except for one minor modification. First we find the best splitting condition with respect to each feature. Then we choose that feature that produces the best splitting condition with respect to all other features. This is performed recursively as before for all resulting branches until we reach a depth where all branches are pure (homogeneous).

Advantages and Limitations of Decision Tree:

Decision Trees are simple to understand and interpret. They are completely unaffected by the scale of the individual features and so do not require feature scaling. On the down side, Decision Trees cannot extrapolate data outside the feature ranges it was trained on. Even with pruning techniques discussed, they tend to over fit, resulting in poor generalization performance when compared to other models.

CHAPTER 3

REQUIREMENTS AND ANALYSIS

3.1 PROBLEM DEFINITION

This project is to stop SQL injection and firing queries to info and to make the info secure. This method is online, therefore there is no want of implementation. It is accessed through the web from anyplace. The system uses the SQL Injection mechanism to keep the info safe and secured. The highlighted half here is the secret writing of card knowledge victimization AES (Advanced secret writing Standard) technique. The net search secures the cardboard payment and won’t let the cardboard knowledge be hacked. whereas a user doing a card payment, all the cardboard knowledge is encrypted so hold on to the info.

System conjointly keeps user details in Associate in Nursing secret writing kind victimization AES secret writing. The system is made of handling SQL Injection capabilities that double the protection of info and prevents from injection hacking codes into the info. Here, the project files and an info file are going to be held on into the cloud, which can be a kind of an association between application and cloud server via web browser.

Many of the antecedent designed ways don't seem to be providing the correct results there is a technique like hindrance of SQL injection exploitation encoding in hold on procedures that square measure unsafe against SQL injection attacks and additionally preventing sql injection attacks supported question optimization method also are unsafe as they cannot be enhanced by indexes alone and should want schema modifications that square measure troublesome to integrate into existing applications and the hindrance of sql injection exploitation rc4 and blowfish techniques unsuccessful because the vulnerabilities found in rc4 square measure extraordinarily insecure thus only a few applications use it currently and it can't be performed on smaller streams of data thanks to these the present methodologies does not able to stop the attacks from the assaulter.

3.2 REQUIREMENT SPECIFICATION

First, whenever anyone wants to access the site, the user has to login to see the details of the items. So, the user has to give the respective login credentials like the username and password, and the password is converted into its hashed format using the MD5 algorithm, and it checks with the hashed password stored in that username record in the database. If both the hashes match, the user logins to the system.

If the hashes do not match or if the username is not present, the user cannot login to the system. If an attacker wants to access the system, he/she will not have a username and password so they will write some malicious code to get into the database. So if any hacker inserts SQL injection queries in the input field like a password, they can access the system and read all data from the database. Here, the MD5 algorithm is used so that if any malicious injection code is written the hacker cannot access the system. Here the hacker tries multiple SQL injection queries, but he cannot login to the system. So here the steps are:

- Create a web application
- Conduct SQLIA before implementing a cryptography algorithm.
- Implementation of cryptographic algorithm on the web application
- Conduct SQLIA after implementing, observe the changes

3.3 PLANNING AND SCHEDULING

1. Prepare

- Working on various encryption algorithms for secure transfer of data.
- Work on Web Development
- Use-case diagram, Activity diagram
- Worked on consent management form.

2. Research

- Working on various encryption algorithms for secure transfer of data.
- Work on Web Development
- Worked on research papers.
- Deploy a web app for the model.

3. Chapter 1 & 2

- Working over various digital watermarking algorithms including both visible & invisible watermarking.
- Work on Web Development
- Class diagram, DFD diagrams.
- Worked on consent management form.

4. Chapter 3 & 4

- Prepare the Gantt chart as per the work plan.
- Worked on research papers.
- Set the outcomes for the project.
- Work on web development.

5. Chapter 5 & 6

- Debug, implement the code for the project and prepare test cases based on observation
- Segregate the testing approaches and prepare documentation accordingly
- Modify the code if necessary and note the changes

6. Chapter 6 & 7

- According to the tests conducted on implemented final project prepare a documentation on the results of each test case and note down user guide.
- Conclude the project with not more than 2 pages for the user and manager to brief through the objective and guide of the project.

3.3.1 GANTT CHART:

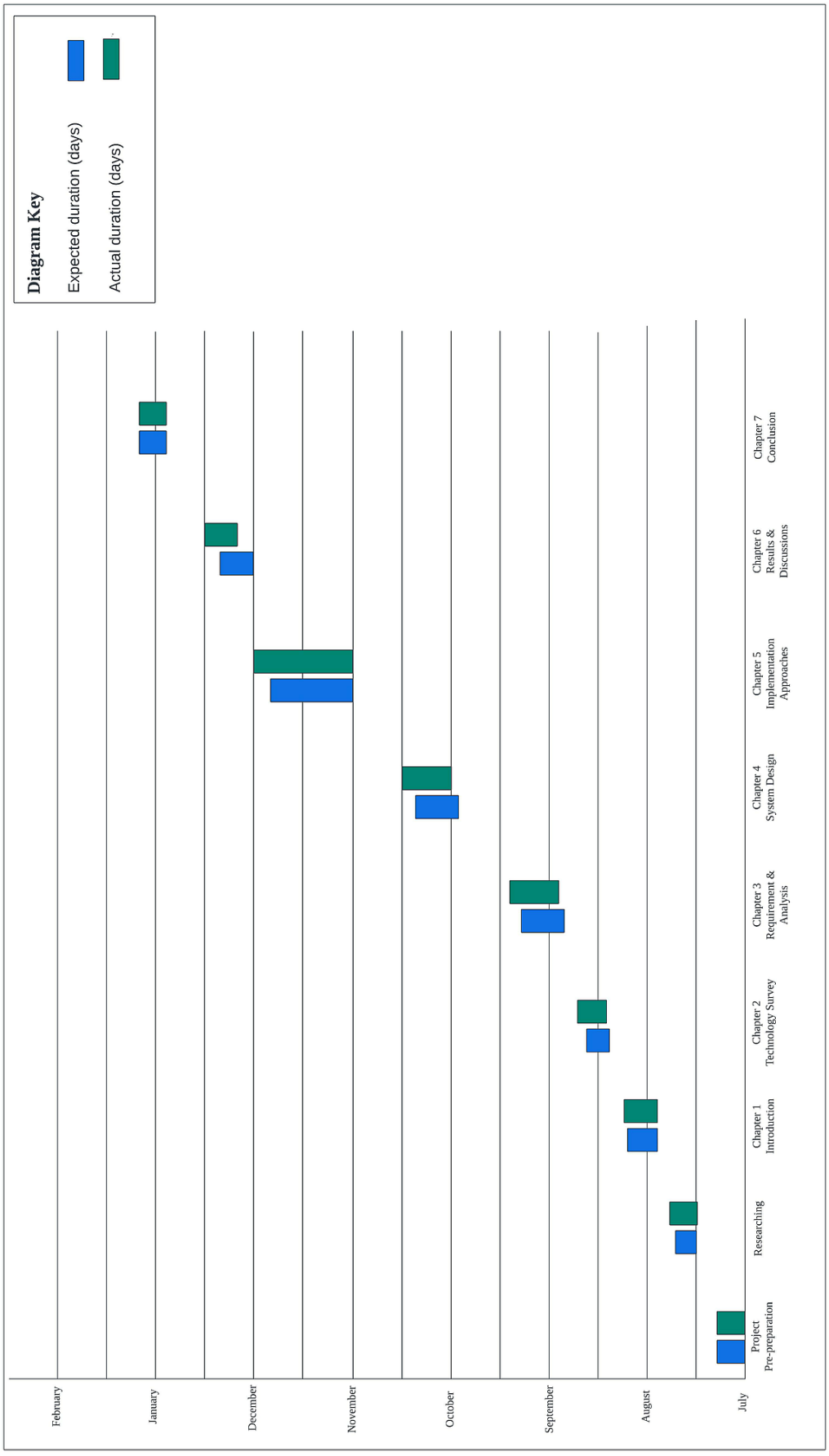
A Gantt chart is a visualization that helps in scheduling, managing, and monitoring specific tasks and resources in a project. It consists of a list of tasks and bars depicting each task's progress. The Gantt chart shows all of the tasks that need to be done, the amount of time each task is expected to take, the time frames in which individual tasks are to be completed, and the relationship between various tasks. This way, everything gets done on schedule, and you never waste time waiting for a task to be completed that should have been done already.

Here, Gantt Chart is made of 4 key features:

- Months: One of the main components of a Gantt chart, the month axis allows the project managers to see not only when the entire project will begin and end, but also when each task will take place. These are displayed along the top of the chart.
- Bars. Once the sub-tasks have been listed, bars are used to show the time frame in which each task should be completed. This helps ensure that every sub-task is done on schedule so the entire project will be completed on time.
- Diagram Key: for the viewer to be familiar of what the bars depict
- Expected time vs Actual time taken to complete the project: usually the actual time taken to complete the project is more than the planned expectations.

3.3.1.1 GANTT CHART

Figure 7



3.3.2 PERT CHART

PERT stands for Project (or Program) Evaluation and Review Technique. PERT chart definition derives from this abbreviation: it is a PM tool that graphically represents a project's tasks, terms, and dependencies between them.

A PERT chart looks like a network diagram. Every separate box (or nod) is a project task. All of them are connected with arrows that show dependencies between tasks. They help to plan and determine tasks and set a realistic timeframe for project completion. In general, this project management tool aims to complete plans on time and accurately estimate the amount of work at the project planning stage.

The PERT chart allows project managers to learn and track important information about the workflow: task dependencies, estimated task time, and minimum project delivery time. So, it suits mostly every project on a planning stage, except for the really small ones, where checklists are enough to see the big picture.

Pert chart will facilitate your workflow when:

- Your plan has multiple tasks in progress simultaneously. The diagram will ease the prioritization process, help you understand which task depends on another.
- A project has a strict time frame. As you already know, a PERT diagram is not only a visual representation of your project's structure but also a helper to evaluate its duration. Using PERT charts allows you to set up terms and always stick to them, track which tasks completion affects the project realization on time.

The PERT chart allows project managers to learn and track important information about the workflow: task dependencies, estimated task time, and minimum project delivery time. So, it suits mostly every project on a planning stage, except for the really small ones, where checklists are enough to see the big picture.

PERT chart will facilitate your workflow when:

- Your plan has multiple tasks in progress simultaneously. The diagram will ease the prioritization process, help you understand which task depends on another.
- A project has a strict time frame. As you already know, a PERT diagram is not only a visual representation of your project's structure but also a helper to evaluate its duration. Using PERT charts allows you to set up terms and always stick to them, track which tasks completion affects the project realization on time.

3.3.2.1 PERT TABLE

	Process	Duration (days)
A	Start project preparation	12
B	Specs finalized	11
C	Detailed information	7
D	Technology finalized	10
E	Analysis layout	20
F	System designing	15
G	Project coding	22
H	Debugging and change implementation	7
I	Project Completion	3

Figure 8

3.3.2.2 PERT CHART

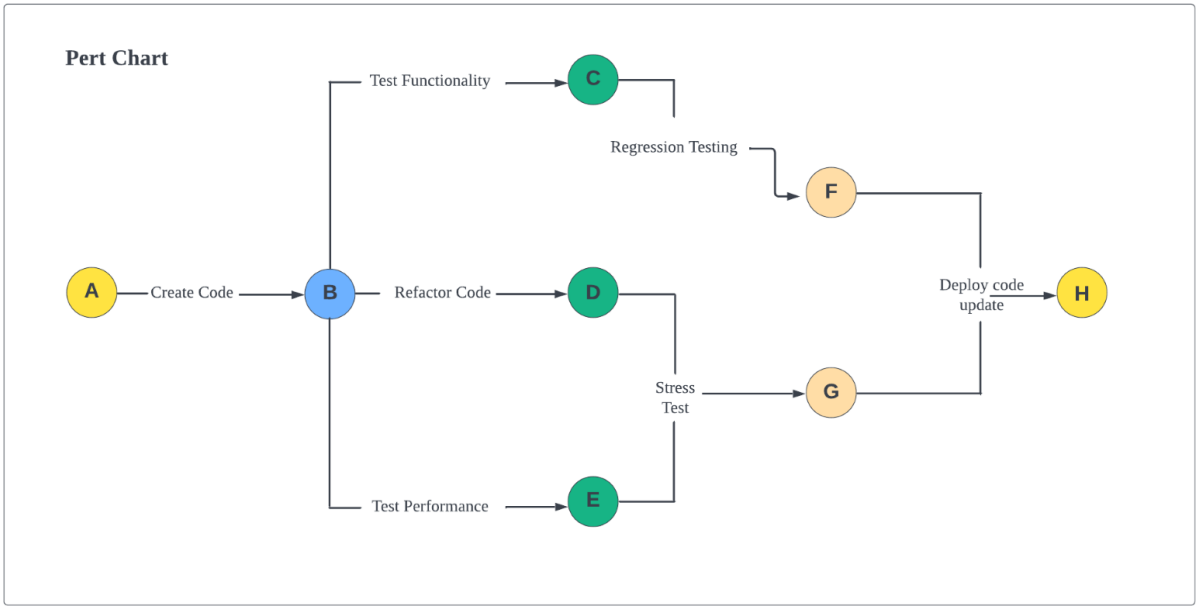


Figure 9

3.3.3 THE SPIRAL MODEL

The spiral model is a systems development lifecycle (SDLC) method used for risk management that combines the iterative development process model with elements of the Waterfall model. The spiral model is used by software engineers and is favored for large, expensive and complicated projects. When viewed as a diagram, the spiral model looks like a coil with many loops. The number of loops varies based on each project and is often designated by the project manager. Each loop of the spiral is a phase in the software development process.

The spiral model enables gradual releases and refinement of a product through each phase of the spiral as well as the ability to build prototypes at each phase. The most important feature of the model is its ability to manage unknown risks after the project has commenced; creating a prototype makes this feasible.

Uses of the spiral model

As mentioned before, the spiral model is best used in large, expensive and complicated projects. Other uses include:

- projects in which frequent releases are necessary;
- projects in which changes may be required at any time;
- long term projects that are not feasible due to altered economic priorities;
- medium to high risk projects;
- projects in which cost and risk analysis is important;
- projects that would benefit from the creation of a prototype; and
- projects with unclear or complex requirements.

Spiral model phases

When looking at a diagram of a spiral model, the radius of the spiral represents the cost of the project and the angular degree represents the progress made in the current phase. Each phase begins with a goal for the design and ends when the developer or client reviews the progress. Every phase can be broken into four quadrants: identifying and understanding requirements, performing risk analysis, building the prototype and evaluation of the software's performance.

Phases begin in the quadrant dedicated to the identification and understanding of requirements. The overall goal of the phase should be determined and all objectives should be elaborated on and analyzed. It is important to also identify alternative solutions in case the attempted version fails to perform.

Next, risk analysis should be performed on all possible solutions in order to find any faults or vulnerabilities -- such as running over the budget or areas within the software that could be open to cyber attacks. Each risk should then be resolved using the most efficient strategy.

In the next quadrant, the prototype is built and tested. This step includes: architectural design, design of modules, physical product design and the final design. It takes the proposal that has been created in the first two quadrants and turns it into software that can be utilized.

Finally, in the fourth quadrant, the test results of the newest version are evaluated. This analysis allows programmers to stop and understand what worked and didn't work before progressing with a new build. At the end of this quadrant, planning for the next phase begins and the cycle repeats. At the end of the whole spiral, the software is finally deployed in its respective market.

Benefits of spiral model :

- Flexibility - Changes made to the requirements after development has started can be easily adopted and incorporated.
- Risk handling - The spiral model involves risk analysis and handling in every phase, improving security and the chances of avoiding attacks and breakages. The iterative development process also facilitates risk management.
- Customer satisfaction - The spiral model facilitates customer feedback. If the software is being designed for a customer, then the customer will be able to see and evaluate their product in every phase. This allows them to voice dissatisfactions or make changes before the product is fully built, saving the development team time and money.

3.3.3.1 SPIRAL MODEL

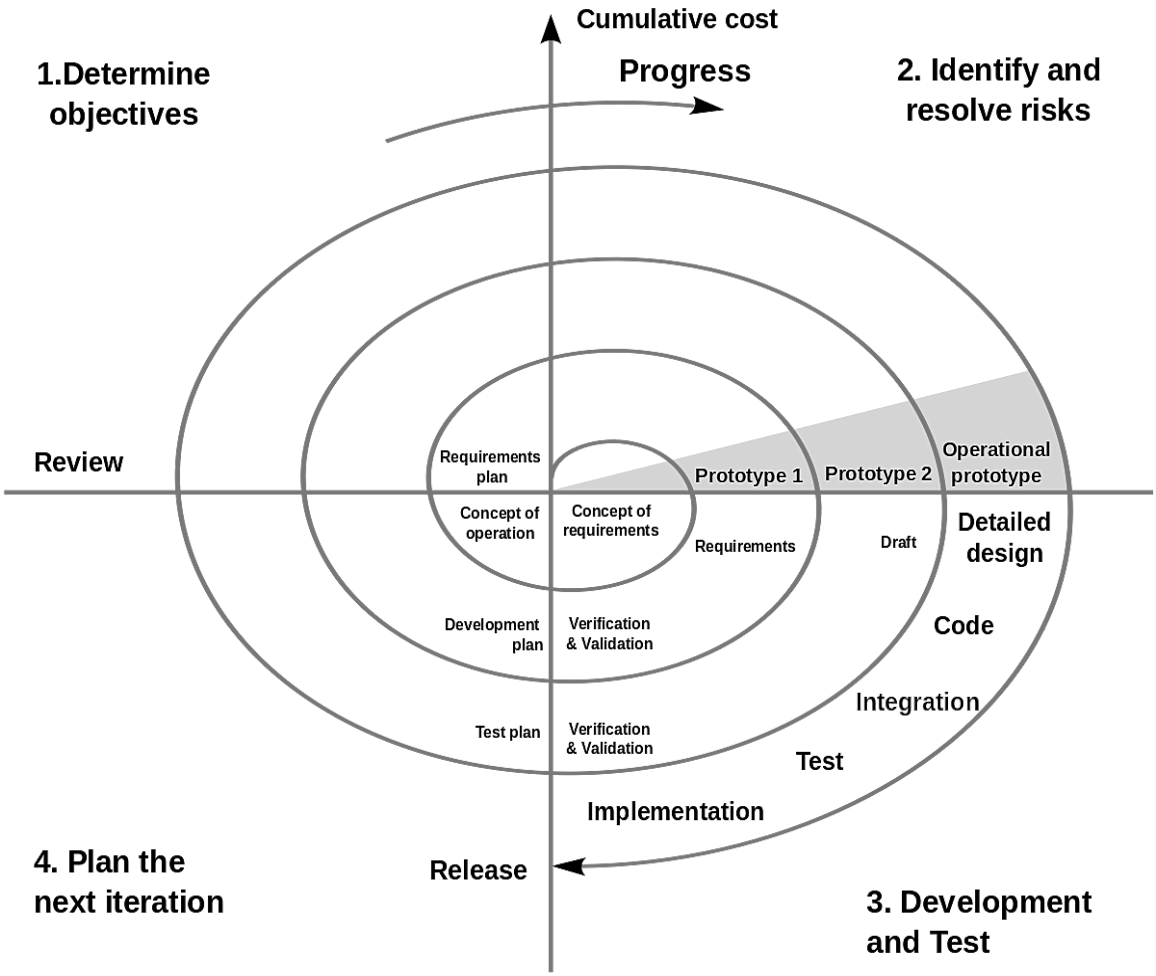


Figure 10

3.4 SOFTWARE AND HARDWARE REQUIREMENTS

Software Requirements:

Windows 7 or above

SQL Server 2008 or above

Anaconda 3

MYSQL

SQLYOG

Hardware Components:

Processor – Core i3

Hard Disk – 160 GB minimum

Memory – 2GB minimum

Internet Connection

3.5 PRELIMINARY PRODUCT DESCRIPTION

This product shortly describes the categories of SQLi attacks that we tend to propose solutions within the next section.

1. Victimization tautology: the instance quoted SQL injection describes the attack victimization tautology.

2. Victimization of illegal/incorrect queries: By providing incorrect inputs, the information could possibly contain some necessary data concerning the table and fields employed in the info. victimization of serial requests like these, the protection is compromised. Within the following example, rather than a legitimate username (xyz), associate degree incorrect input (xyz') is provided and a mistake message is returned giving clues regarding the info, creating it liable to intrusion. Error: choose username, countersign from student wherever username = xyz' Here field names like username and countersign from table student are exposed.

3. Victimization Piggy-Backed Queries: antecedence mentioned attack sorts attempt to gain unauthorized access to the appliance or fetch details regarding the info with none further queries further to the input question. If further queries are piggy-backed within the input space victimization special characters like " ", " – or `` " ; " hackers will modify or delete the info. associate degree example to the present is: choose * from User wherever id=123;drop table User; Here ";" acts as a delimiter and a further drop statement could also be dead and delete the table

4. Blind injection: just in case of incorrect queries, programmers attempt to hide info data as a live to shield the appliance from attacks victimization. Now, the hacker doesn't get any clues regarding the info. The choice to compromise security is by querying the info with tons of true/false queries and checking its result. supported the response of the appliance to the queries, the hackers tried to embezzled access. This sort of attack is most conspicuously used. It includes temporal arrangement attacks collectively of the techniques to spot the behavior of the appliance.

3.6 CONCEPTUAL MODELS

Conceptual models define the relationships between your commercial products, the services that they represent, and the resources that are required to implement the services. They define how commercial products and technical services are related, and they enable you to associate the products that you sell with the technical services and resources that are required to fulfill orders. Conceptual models comprise entities that represent components of a service but that contain no detailed application-specific information. The information that you define for conceptual model entities determines how service capabilities can be commercialized. For example, the conceptual model defines your products and services and the actions that must be performed in a run-time environment to provision a service order request. When designing conceptual models, you define the associations among the conceptual model entities. You also define the patterns that are used to fulfill service orders and technical orders. These patterns define how requests to design reusable service components are fulfilled and how delivery systems activate and manage the work to be performed on resources in the network.

3.6.1 CONCEPTUAL MODEL DIAGRAM

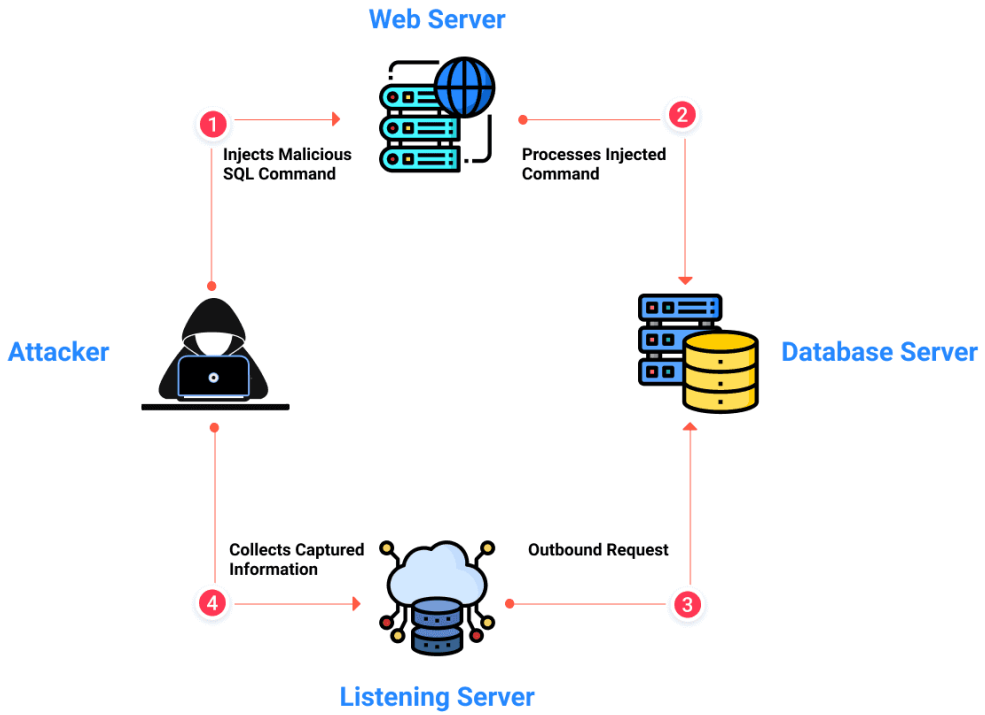


Figure 11

3.6.2 DATA FLOW DIAGRAM

Data flow diagram functions to figure division system into smaller module, making easy user that insufficiently understand computer area for understands system that will be worked. DFD constitutes tool that is utilized on systems developmental methodology which most structure. A data flow diagram (DFD) shows how data moves through an information system but does not show program logic or processing steps. DFD represent a logical model that shows what the system does, not how it does. That distinction is important because focusing on implementation issues at this point would restrict your search for the most effective system design. The DFD enables the software engineer to develop models of the information domain & functional domain at the same time. As the DFD is refined into greater levels of details, the analyst perform an implicit functional decomposition of the system. At the same time, the DFD refinement results in a corresponding refinement of the data as it moves through the process that embody the applications. A context-level DFD for the system the primary external entities produce information for use by the system and consume information generated by the system. The labeled arrow represents data objects or object hierarchy.

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often, they are a preliminary step used to create an overview of the system which can later be elaborated DFDs can also be used for the visualization of data processing (Structured design).

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

3.6.2.1 DFD COMPONENTS

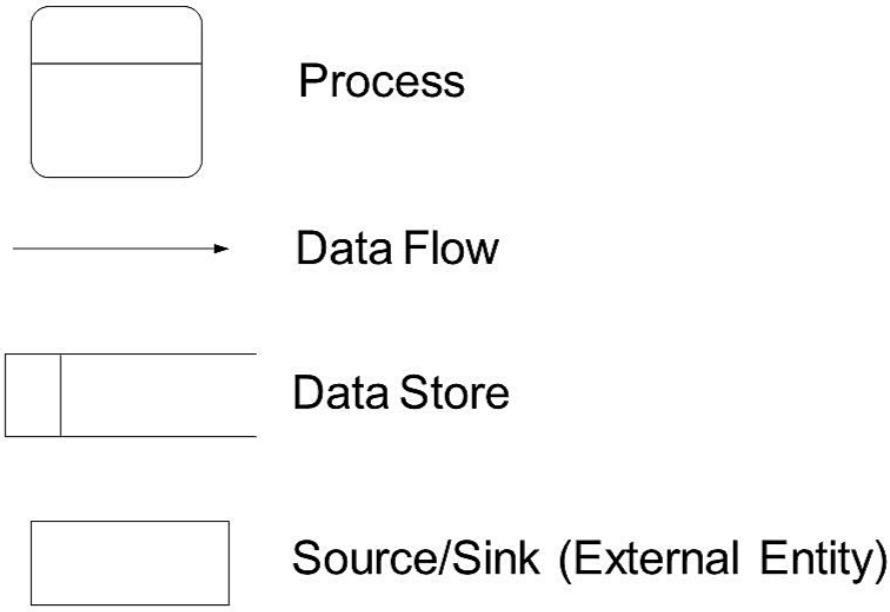
Circle: A circle (bubble) shows a process that transforms data inputs into data outputs.

Data Flow: A curved line shows the flow of data into or out of a process or data store.

Data Store: A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

Source or Sink: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

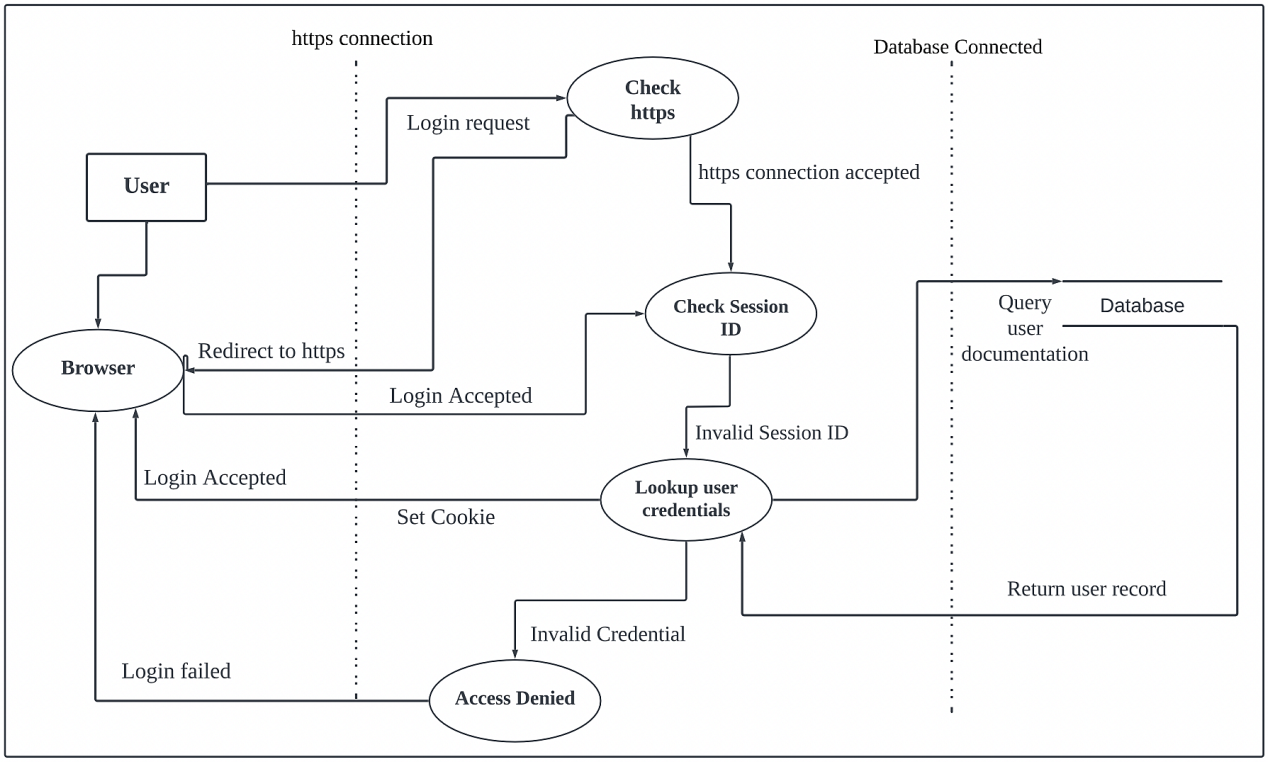
Figure 12



1. DFD LEVEL 0

It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

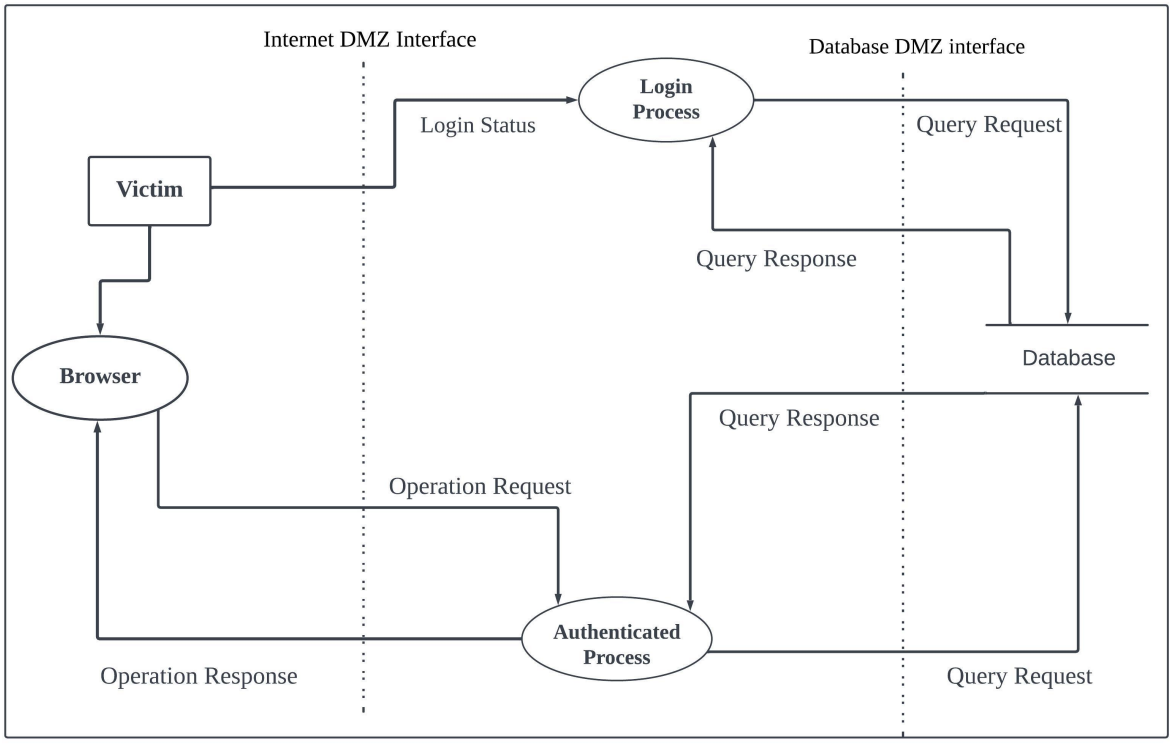
Figure 13



2. DFD LEVEL 1

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into sub processes. It provides a more detailed view of the Context Level Diagram.

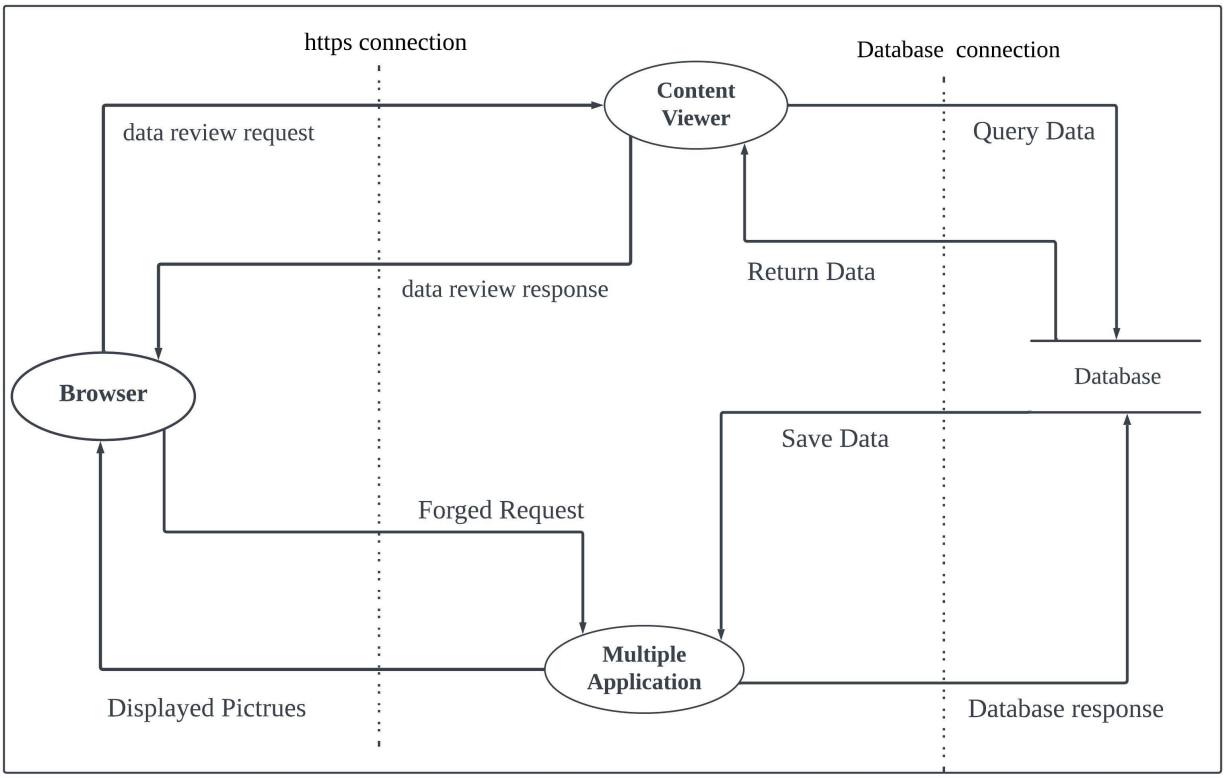
Figure 14



3. DFD LEVEL 2

This level two data flow diagram (DFD) template can map out information flow, visualize an entire system, and be shared with your stakeholders.2-level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system's functioning.

Figure 15



3.6.2.2 DFD DIAGRAM

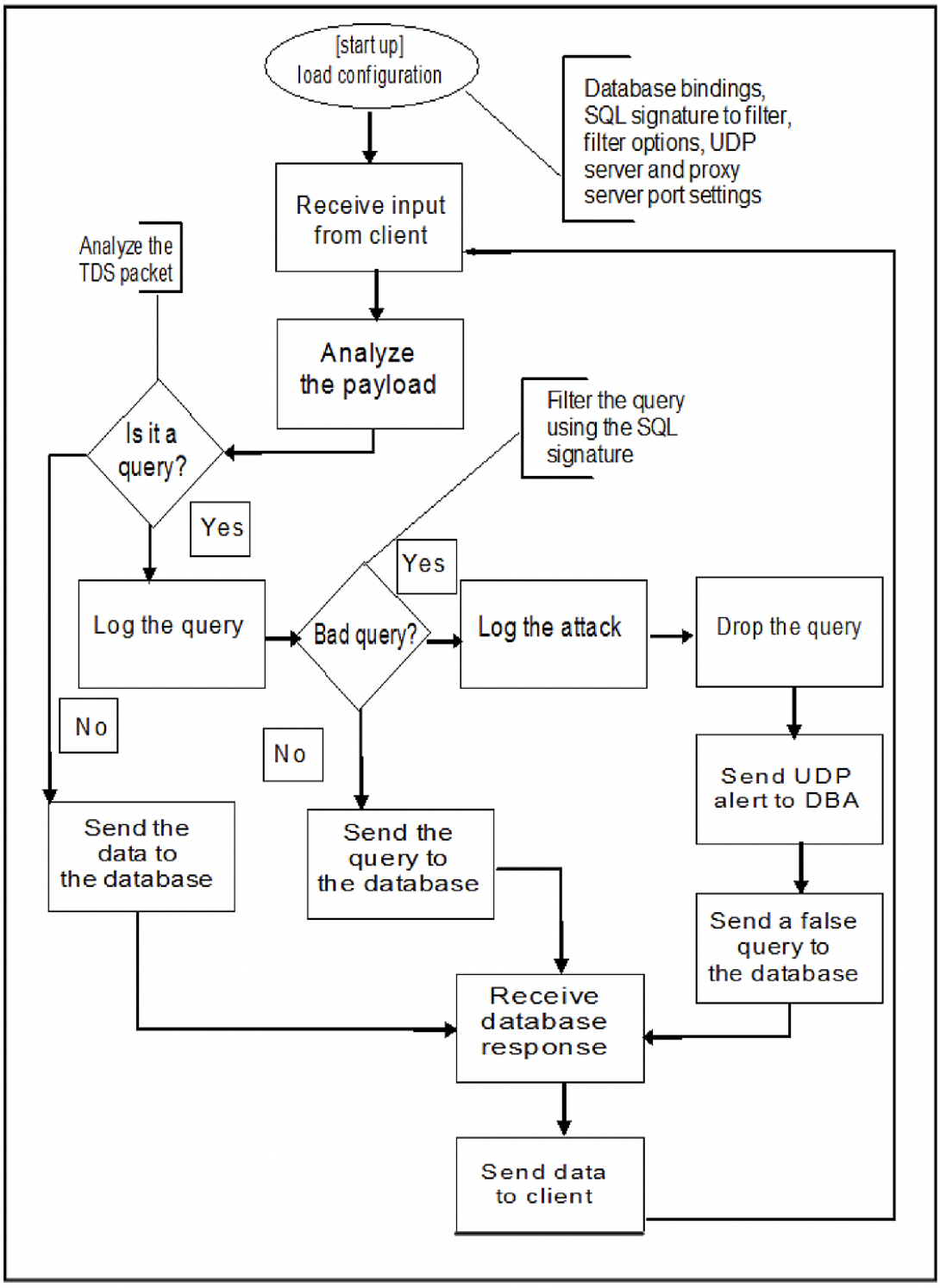


Figure 16

3.6.3 ER DIAGRAM

The ER or (Entity Relational Model) is a high-level conceptual data model diagram. Entity-Relation model is based on the notion of real-world entities and the relationship between them. An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER modeling is something regarded as a complete approach to design a logical database schema.This is incorrect because the ER diagram is just an approximate description of data, constructed through a very subjective evaluation of the information collected during requirements analysis. ER Diagrams are composed of entities, relationships and attributes.

Entity

An entity is an object or component of data. An entity is represented as a rectangle in an ER diagram. For example: Student and College and these two entities have many to one relationship as many student studies in a single college. An entity that cannot be uniquely identified by its own attributes and relies on the relationship with another entity is called a weak entity. The weak entity is represented by a double rectangle.

Attribute

An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

1.Key attribute

2.Composite attribute

3. Multivalued attribute

4. Derived attribute

Relationship

A relationship is represented by a diamond shape in the ER diagram. It shows the relationship among entities. There are four types of relationships:

1. One to One
2. One to Many
3. Many to One
4. Many to Many

3.6.3.1 ER DIAGRAM COMPONENTS

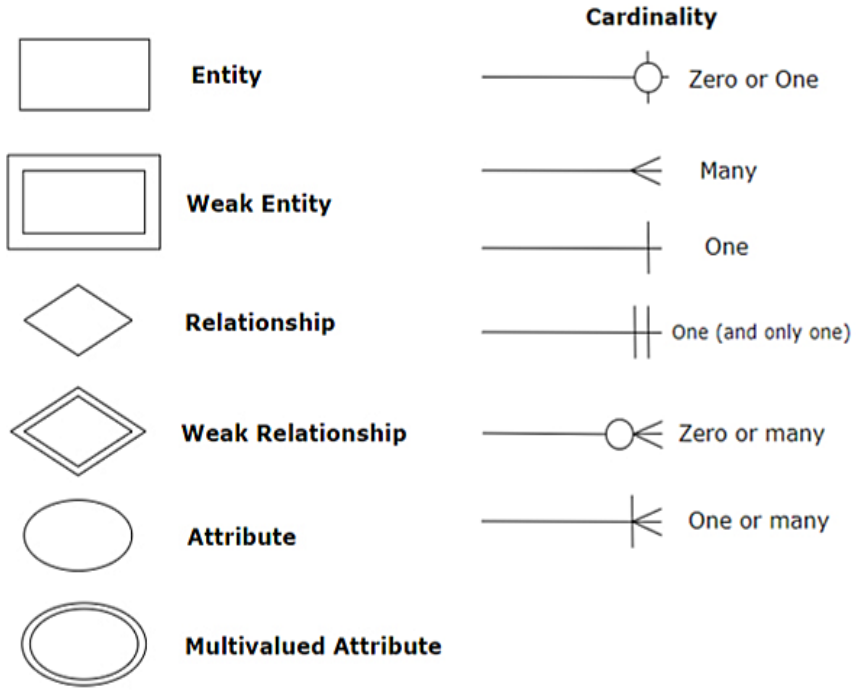


Figure 17

3.6.3.2 ER DIAGRAM

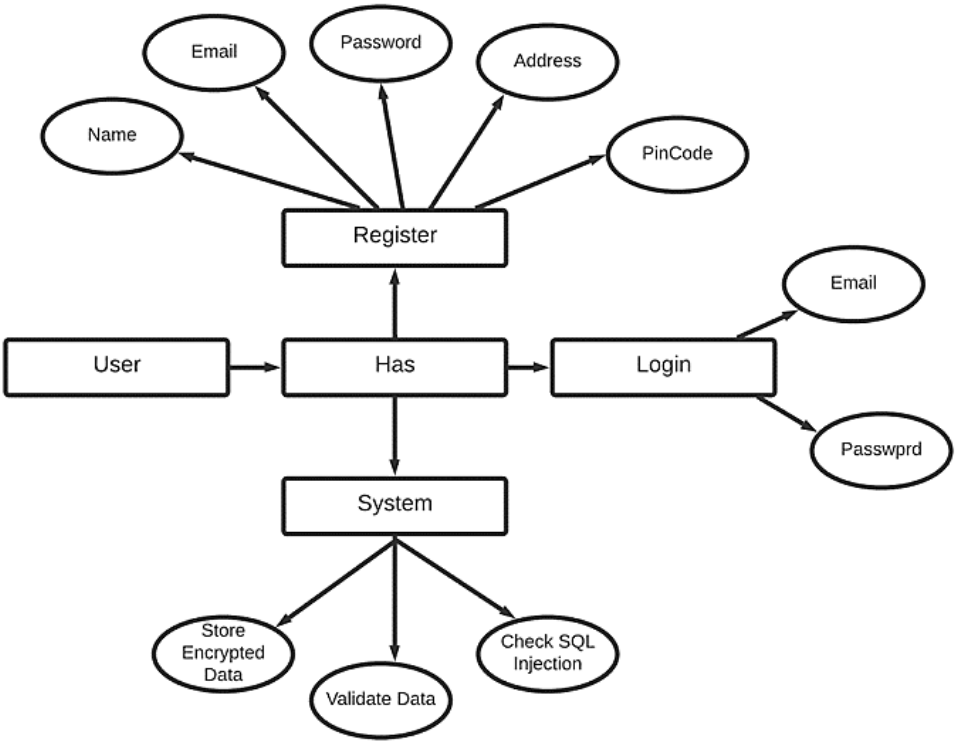


Figure 18

3.6.4 USE CASE DIAGRAM

Use case diagrams model behavior within a system and help the developers understand what the user requires. The stick man represents what's called an actor.

Use case diagram can be useful for getting an overall view of the system and clarifying who can do and more importantly what they can't do.

The use case diagram consists of use cases and actors and shows the interaction between the use case and actors.

- The purpose is to show the interactions between the use case and actor.
- To represent the system requirements from user's perspective.
- An actor could be the end-user of the system or an external system.

A Use case is a description of a set of sequence of actions. Graphically it is rendered as an ellipse with solid line including only its name. Use case diagram is a behavioral diagram that shows a set of use cases and actors and their relationship. It is an association between the use cases and actors. An actor represents a real-world object. Primary Actor – Sender, Secondary ActorReceiver.

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

3.6.4.1 USE CASE COMPONENTS

- **Actor** – anyone or anything that performs a behavior (who is using the system)
- **Stakeholder** – someone or something with vested interests in the behavior of the system under discussion (SUD)
- **Primary Actor** – stakeholder who initiates an interaction with the system to achieve a goal
- **Preconditions** – what must be true or happen before and after the use case runs.
- **Triggers** – this is the event that causes the use case to be initiated.
- **Main success scenarios** [Basic Flow] – use case in which nothing goes wrong.
- **Alternative paths [**Alternative Flow] – these paths are a variation on the main theme. These exceptions are what happen when things go wrong at the system level.

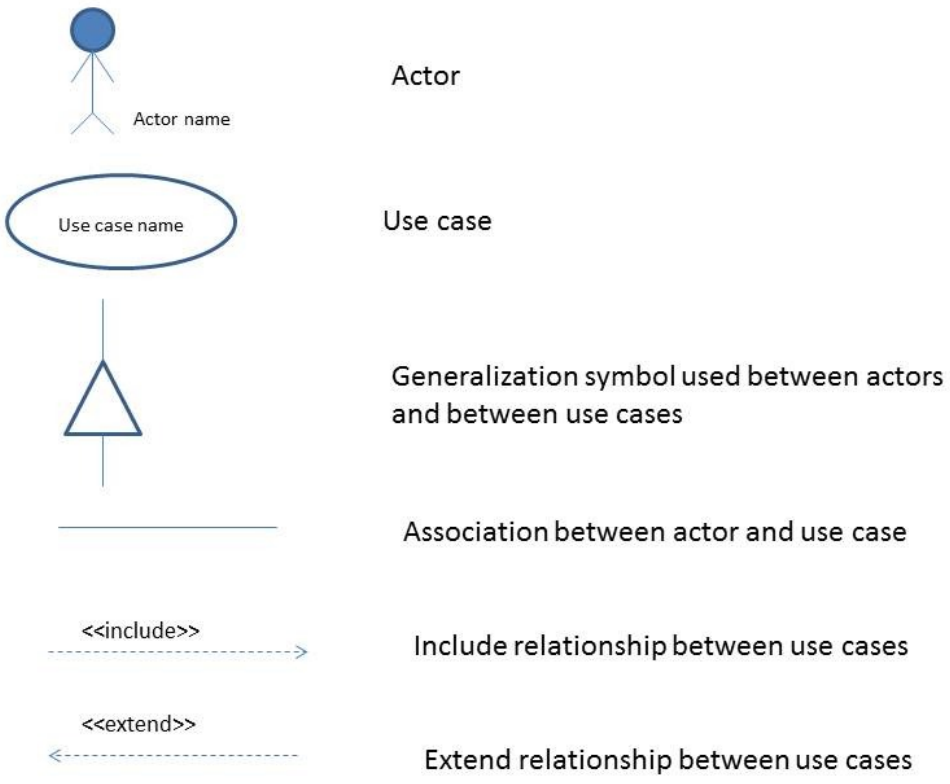
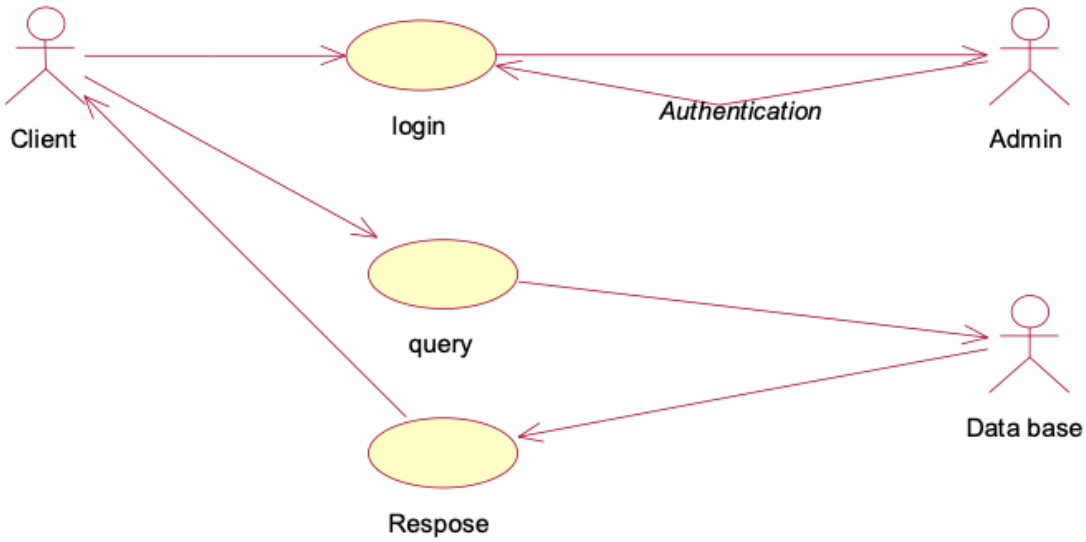


Figure 19

3.6.4.2 USE CASE DIAGRAM

Figure 20



Use Case Diagram for Combination approach

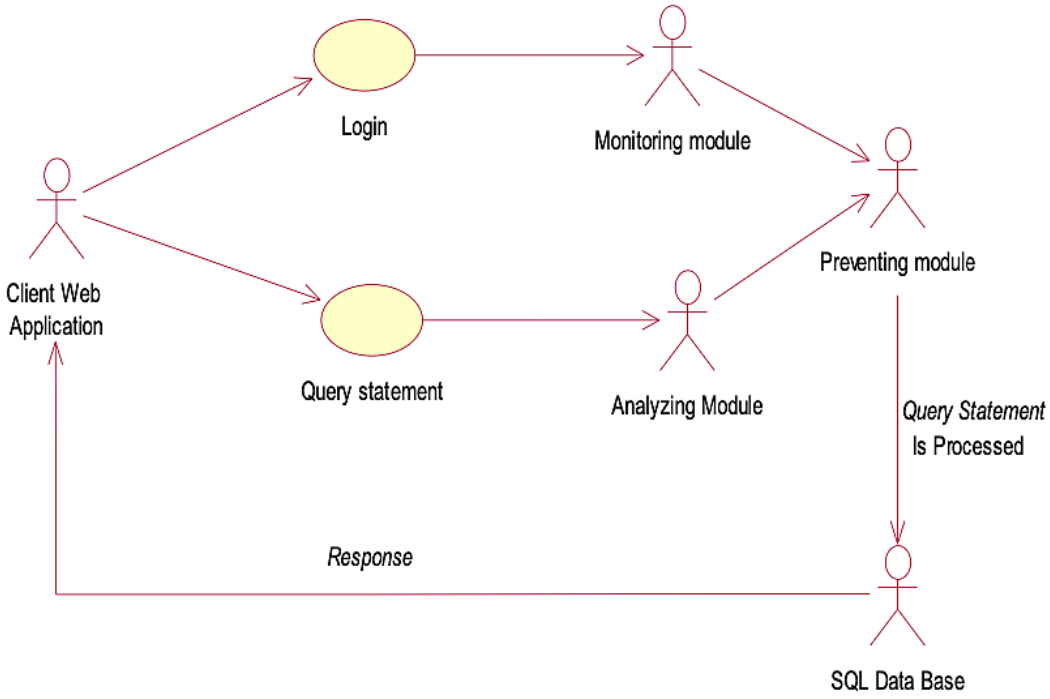


Figure 21

3.6.5 CLASS DIAGRAM

It is a model which is used to show the classes constituting a system and their interrelationship. It is based on UML. Only the important attributes and methods are shown in class diagrams. In the initial period of analysis, the important attributes of the classes, which must be captured and the functionalities provided by the class may not be obvious. As the analysis progresses, the attributes and methods may be added. If more focus is on interrelationships between classes, then the attributes and methods may not be shown in the class diagram. The class diagram is used to identify and classify the objects which constitute a system. It also includes the important attributes of the objects which must be captured.

Class is nothing but a structure that contains both variables and methods. The Class diagram shows a set of classes, interfaces, and collaborations and their relating ships. There is the most common diagram in modeling object oriented systems and are used to give the static view of a system. It shows the dependency between the classes that can be used in our system. The interactions between the modules or classes of our projects are shown below. Each block contains Class Name, Variables and Methods.

In the diagram, classes are represented with boxes which contain three parts

- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake

In the design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those objects. With detailed modeling, the classes of the conceptual design are often split into many sub classes.

3.6.5.1 CLASS DIAGRAM COMPONENTS

Class Name:

- The name of the class appears in the first partition.

Class Attributes:

- Attributes are shown in the second partition.
- The attribute type is shown after the colon.
- Attributes map onto member variables (data members) in code.

Class Operations (Methods):

- Operations are shown in the third partition. They are services the class provides.
- The return type of a method is shown after the colon at the end of the method signature.
- The return type of method parameters are shown after the colon following the parameter name. Operations map onto class methods in code

Class Visibility

The +, - and # symbols before an attribute and operation name in a class denote the visibility of the attribute and operation.

- + denotes public attributes or operations
- - denotes private attributes or operations
- # denotes protected attributes or operations

Figure 22

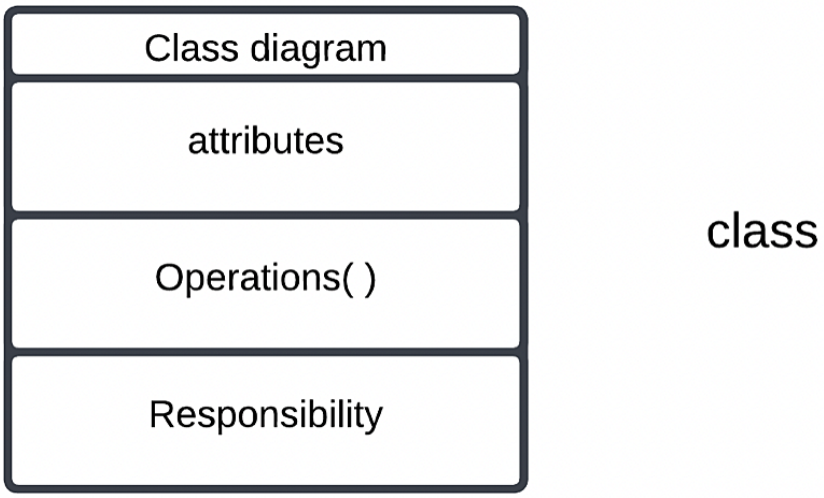
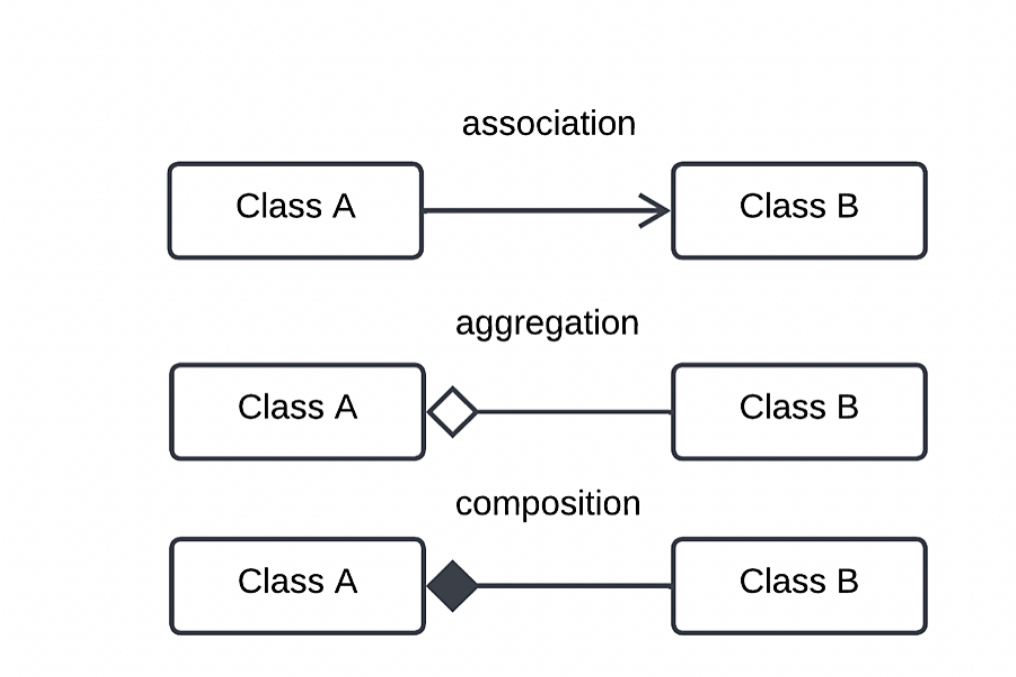


Figure 23



3.6.5.2 CLASS DIAGRAM

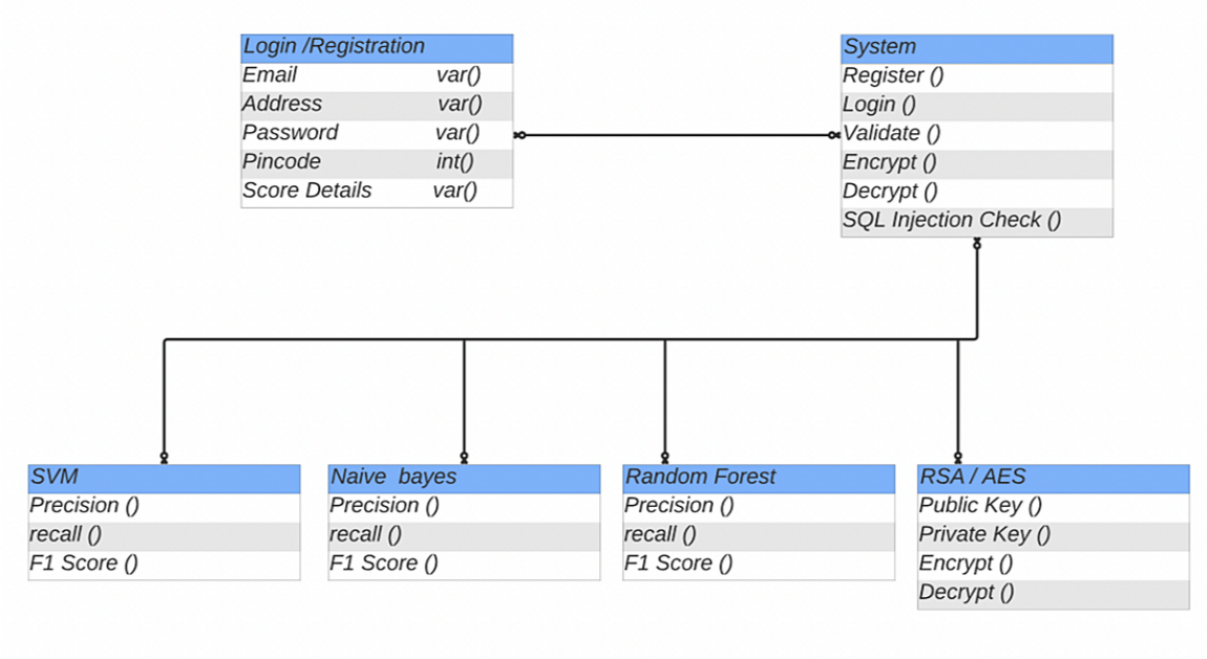


Figure 24

3.6.6 FLOW CHART

A general project management flowchart is a diagram that depicts a series of actions that begin with the commencement and progress of a project leading to a conclusion. However, if the project statement is not accepted, you'll have to amend or terminate the project. As soon as your scope statement is accepted, you can move forward with the strategic planning. A flowchart is a picture of the separate steps of a process in sequential order. It is a generic tool that can be adapted for a wide variety of purposes, and can be used to describe various processes, such as a manufacturing process, an administrative or service process, or a project plan. It's a common process analysis tool and one of the seven basic quality tools. Elements that may be included in a flowchart are a sequence of actions, materials or services entering or leaving the process (inputs and outputs), decisions that must be made, people who become involved, time involved at each step, and/or process measurements.

A project management process flowchart is a graphical aid, designed to visualize the sequence of steps to be followed throughout the project management process. Once your process flow has been developed, it will guide the primary phases of any future projects, from start to finish.

3.6.6.1 COMPONENTS OF FLOW CHART

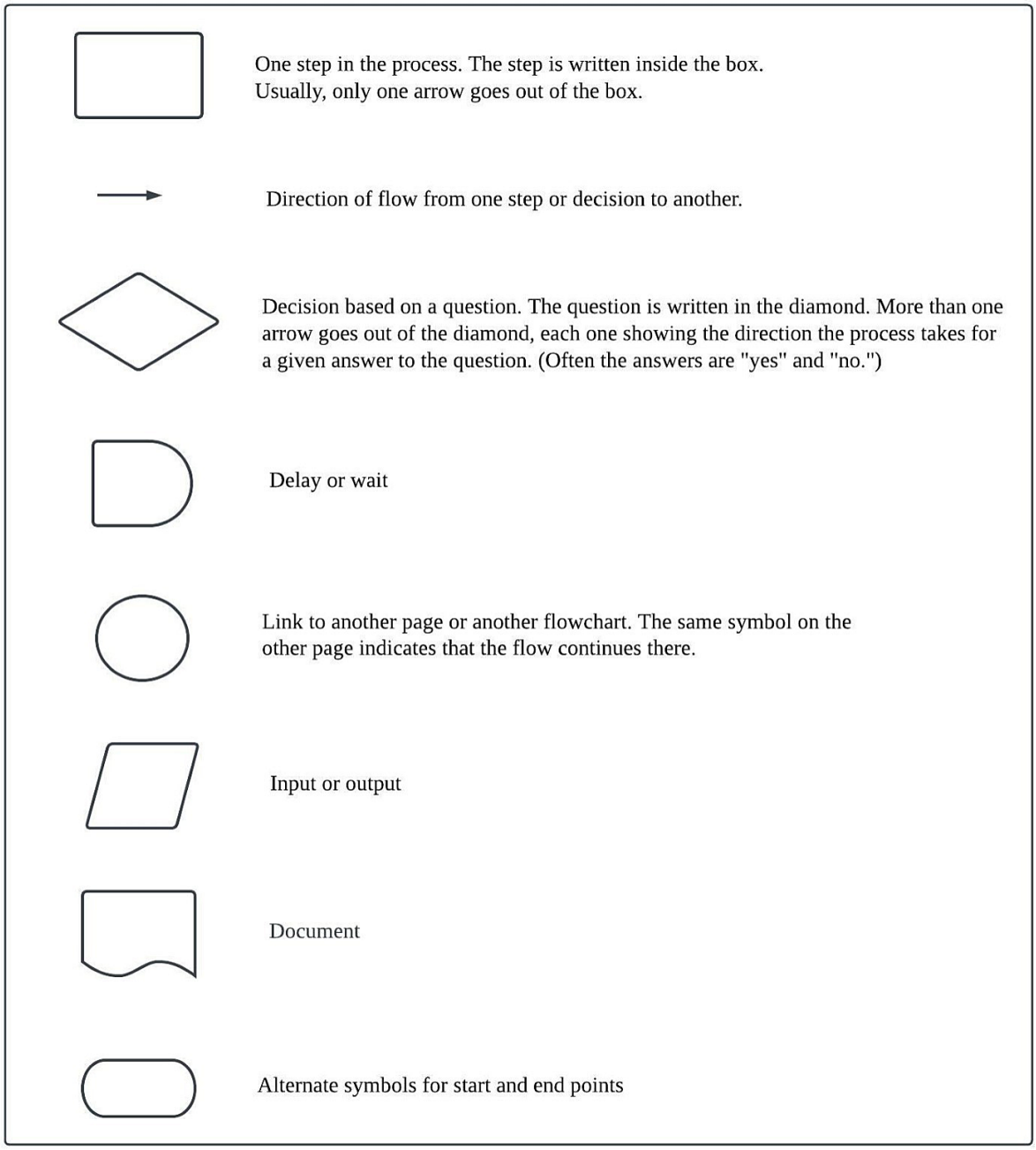
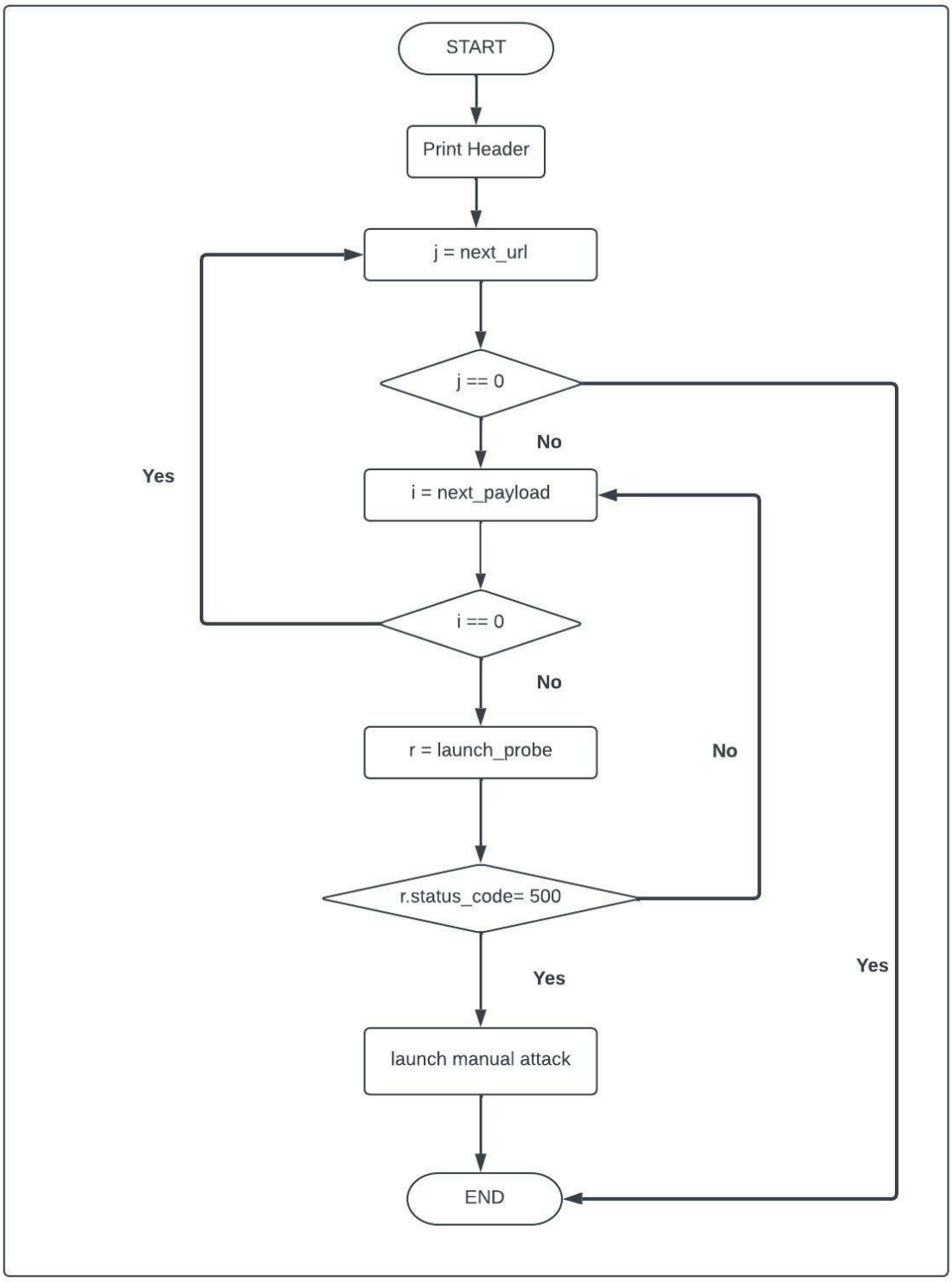


Figure 25

3.6.6.2 FLOW CHART

Figure 26



3.6.7 UML SEQUENCE DIAGRAM

To understand what a sequence diagram is, it's important to know the role of the Unified Modeling Language, better known as UML. UML is a modeling toolkit that guides the creation and notation of many types of diagrams, including behavior diagrams, interaction diagrams, and structure diagrams. A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

Benefits of sequence diagrams

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.

Use cases for sequence diagrams

The following scenarios are ideal for using a sequence diagram:

- Usage scenario: A usage scenario is a diagram of how your system could potentially be used. It's a great way to make sure that you have worked through the logic of every usage scenario for the system.
- Method logic: Just as you might use a UML sequence diagram to explore the logic of a use case, you can use it to explore the logic of any function, procedure, or complex process.
- Service logic: If you consider a service to be a high-level method used by different clients, a sequence diagram is an ideal way to map that out.
- Sequence diagram Visio - Any sequence diagram that you create with Visio can also be uploaded into Lucidchart. Lucidchart supports .vsd and .vdx file import and is a great Microsoft Visio alternative.

3.6.7.1 UML SEQUENCE DIAGRAM

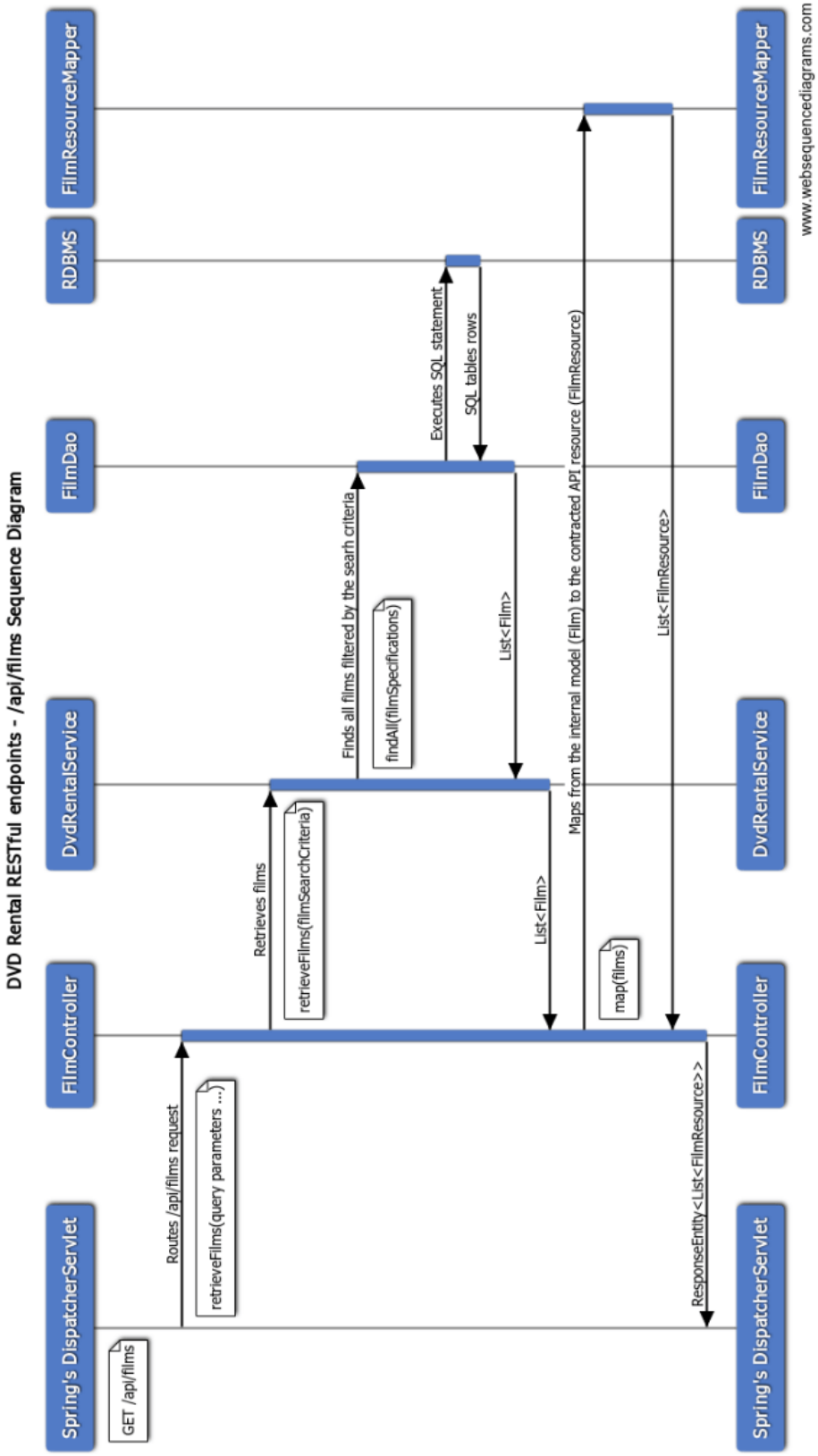


Figure 27

3.6.8 ACTIVITY DIAGRAM

Activity Diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The basic purpose of activity diagrams is to capture the dynamic behavior of the system. It is also called object-oriented flowchart. An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling. They can also describe the steps in a use case diagram. Activities modeled can be sequential and concurrent. In both cases, an activity diagram will have a beginning (an initial state) and an end (a final state).

Activity diagrams present a number of benefits to users. Consider creating an activity diagram to:

- Demonstrate the logic of an algorithm.
- Describe the steps performed in a UML use case.
- Illustrate a business process or workflow between users and the system.
- Simplify and improve any process by clarifying complicated use cases.
- Model software architecture elements, such as method, function, and operation.

3.6.8.1 COMPONENTS OF ACTIVITY DIAGRAM

Before you begin making an activity diagram, you should first understand its makeup. Some of the most common components of an activity diagram include:

- Action: A step in the activity wherein the users or software perform a given task.
- Decision node: A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.
- Control flows: Another name for the connectors that show the flow between steps in the diagram.
- Start node: Symbolizes the beginning of the activity. The start node is represented by a black circle.
- End node: Represents the final step in the activity. The end node is represented by an outlined black circle.

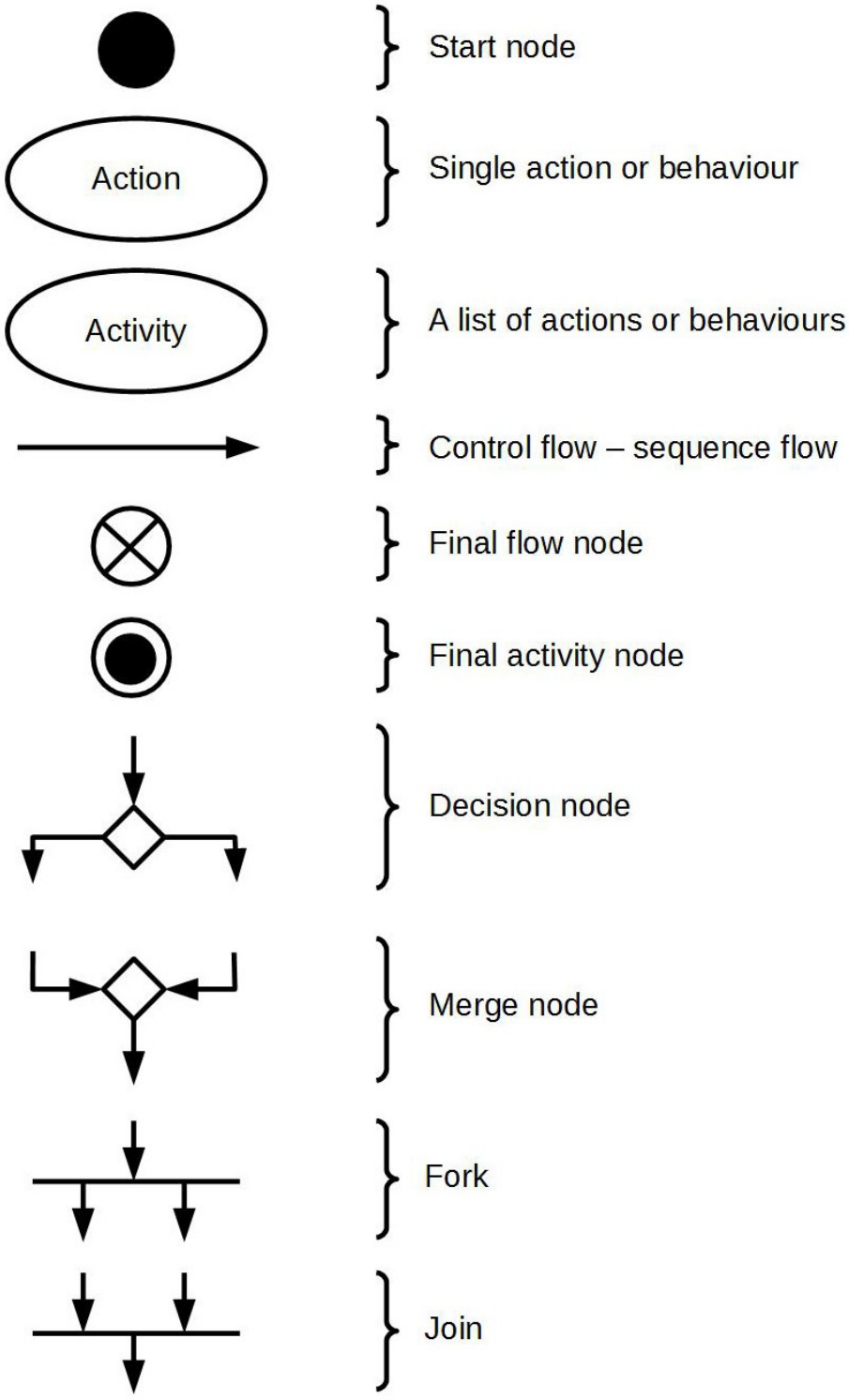

Start node
Action
Single action or behaviour
Activity
A list of actions or behaviours
Control flow – sequence flow
Final flow node
Final activity node
Decision node
Merge node
Fork
Join

Figure 28

3.6.8.2 ACTIVITY DIAGRAM

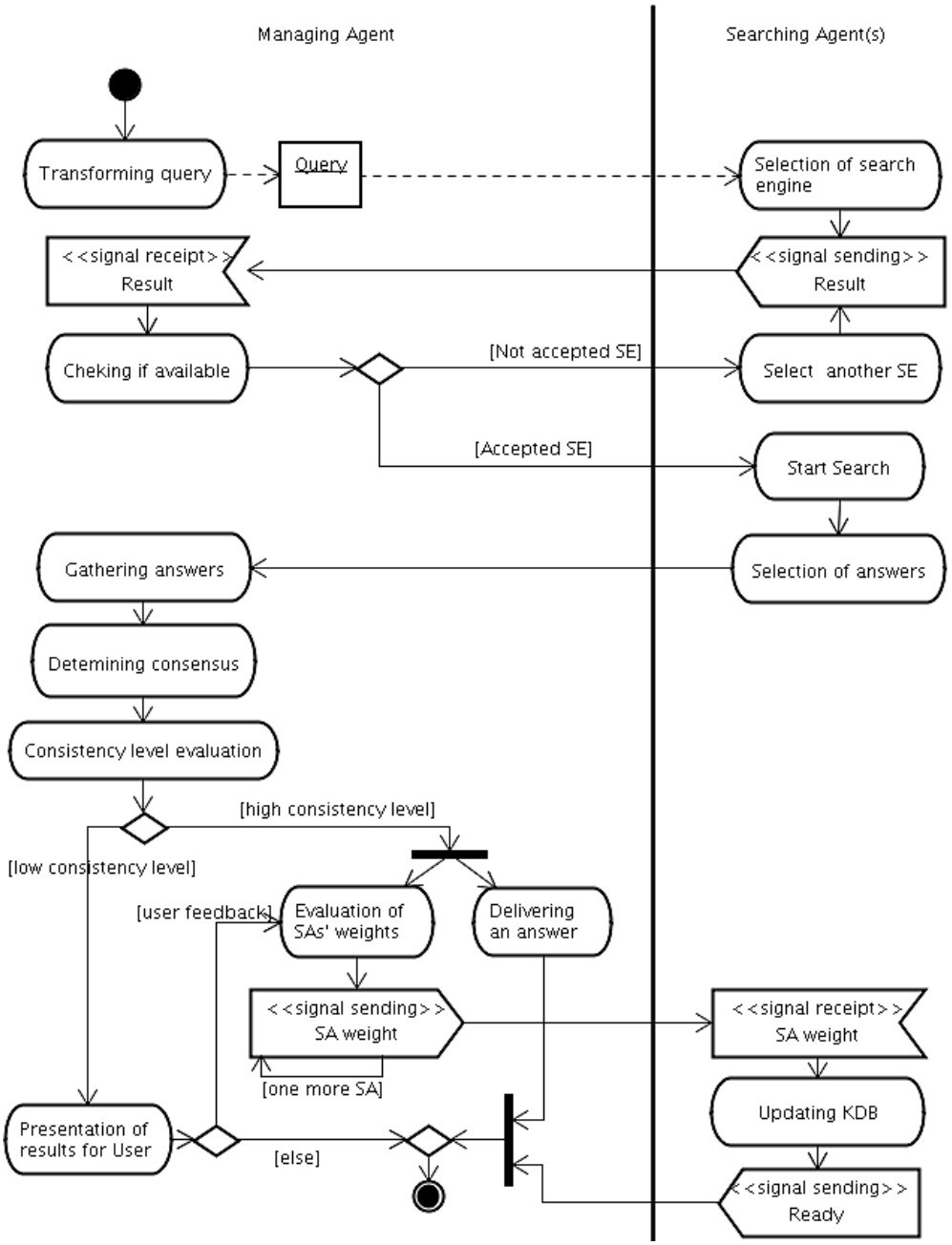
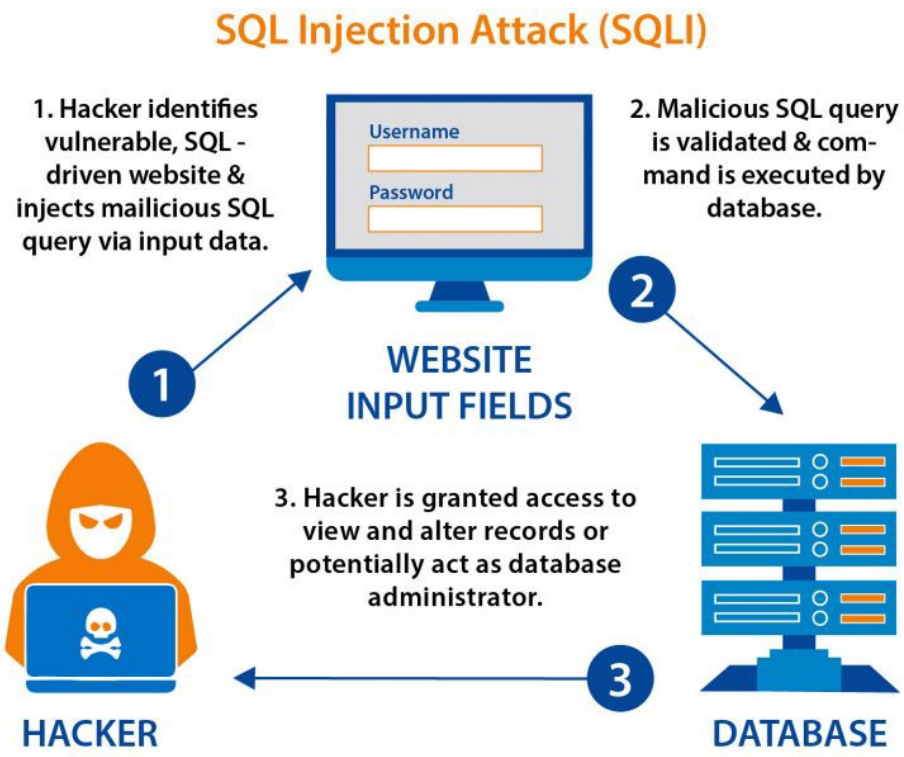


Figure 29

3.6.9 COMPONENT DIAGRAM

Component diagrams provide a simplified, high-order view of a large system. Classifying groups of classes into components supports the interchangeability and reuse of code. This diagram documents how these components are composed and how they interact in a system. The main purpose of a component diagram is to show the structural relationships between the components of a system. In UML, Components are made up of software objects that have been classified to serve a similar purpose. Components are considered autonomous, encapsulated units within a system or subsystem that provide one or more interfaces. By classifying a group of classes as a component, the entire system becomes more modular as components may be interchanged and reused. Component diagrams document the encapsulation of the component and the means by which the component interacts via interfaces.

Figure 30



CHAPTER 4

SYSTEM DESIGN

4.1.0 BASIC MODULES

This system comprises 2 major modules and their sub modules:

1. Admin Login: Admin needs to login by providing the login credentials to access the below given admin modules

- Product Entry:- Admin can enter details about new products.
- View Order:- Admin can view details about the order placed by the user.
- View Users Details:- Admin can view all the registered user's details.

2. User Login/Registration: User can register on the system and get his online account on site.

- View Products:- The products are arranged and can be viewed in categories.
- Add to Cart: - Users can add multiple products to cart.
- Pay using Card: - After the total bill is calculated the user can pay via credit card online.

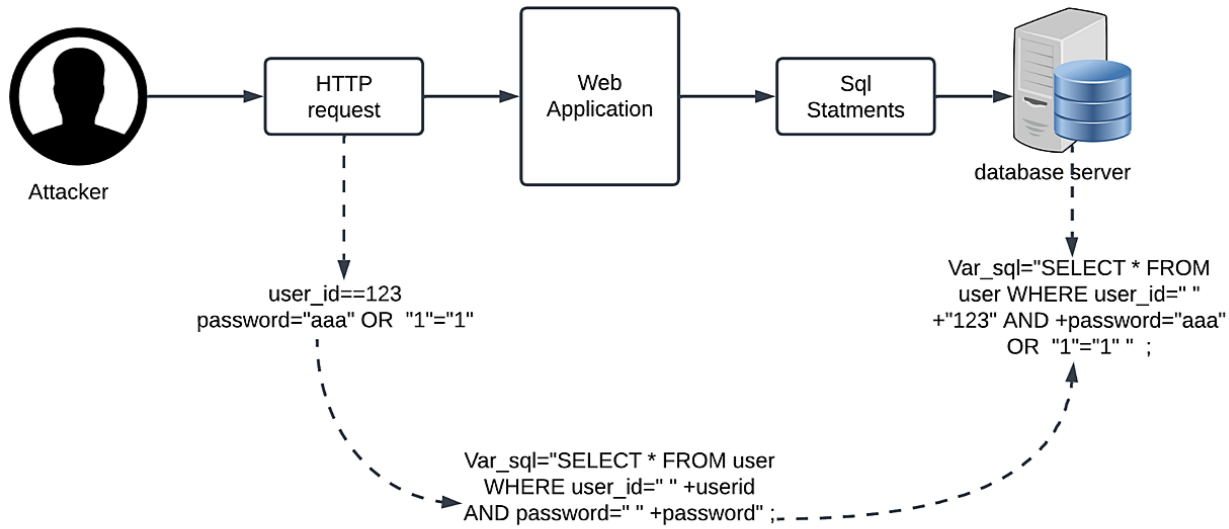
3. View Order: - User can view details about the order placed

4.2.0 DATA DESIGN

SQL Injection is one of the main issues in database security. It affects the database without the knowledge of the database administrator. It may delete the full database or records or tables without the knowledge of the respective user or administrator. It is a technique used to exploit the database system through vulnerable web applications. These attacks not only make the attacker breach the security and steal the entire content of the database but also, to make arbitrary changes to both the database schema and the contents. SQL injection attack could not be realized about compromised information until long after the attack has passed. In many scenarios, the victims are unaware that their confidential data has been stolen or compromised. With the help of a simple web browser, SQL Injection attacks can be performed by attackers.

4.2.0.1 DATA DESIGN DIAGRAM

Figure 31



4.2.1 SCHEMA DESIGN

When constructing the backend of an application, you need to take into account how the frontend will talk to the backend. More important, however, is the construction and design of your database. The relationships your data forms will lead to the construction of your database schema. **Database Schema** refers to a structure that represents relationships among data and defines how information is stored in a database. Without a proper schema, it is easy to drift away from the objective considering the scale of big data projects. As schema also represents the relationship among tables, different databases have different schema designs to support varying business requirements. A Schema organizes data into Tables with appropriate Attributes, shows the interrelationships between **Tables** and **Columns**, and imposes constraints such as Data types. A well-designed Schema in a Data Warehouse makes life easier for Analysts by:

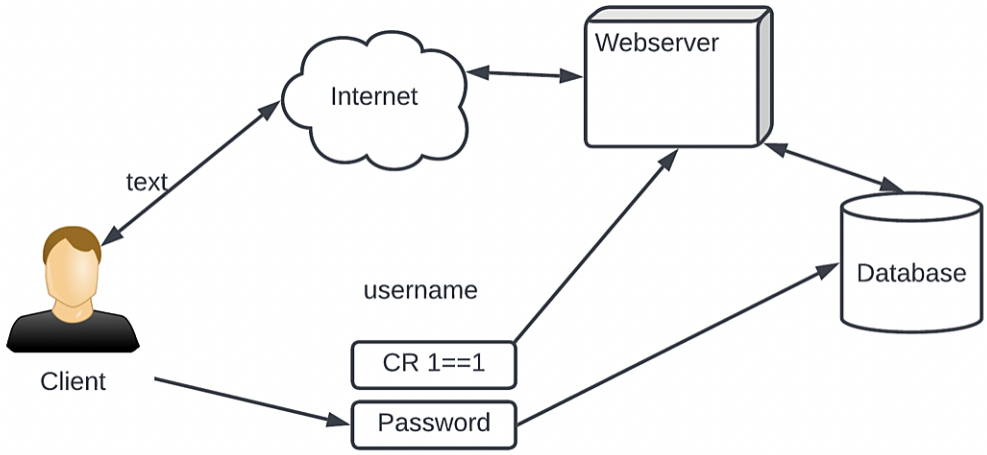
- removing cleaning and other preprocessing from the analyst's workflow
- absolving analysts from having to reverse-engineer the underlying Data Model
- providing analysts with a clear, easily understood starting point for analytics

In other words, a well-designed Schema clears the way to faster and easier creation of Reports and Dashboards. By contrast, a flawed Schema requires Data Analysts to do extra modeling and forces every Analytics query to take more time and system resources, increasing an organization's costs and irritating everyone who wants their analytics right away.

It processes the user SQL queries which are used to retrieve information from a database. A vast majority of the web applications are susceptible to vulnerabilities in validation of user inputs and hence are defenseless against SQLinjection. SQLIA is moderately simple to perform and hard to prevent.The user inputs are utilized for making SQL queries which are exchanged with the database. If the entered values are found as expected the user's access is allowed otherwise access will be denied. The attacks of SQL injection attacks are in many forms, which includes Error-based Injection, Tautology based Injection, Union-based Injection and Blind SQL injection. In a tautology-based attack user gets unauthorized access to the database by including a tautology that always evaluates to true in the query.

4.2.1.1 SCHEMA DESIGN DIAGRAM

Figure 32



4.2.2 DATA INTEGRITY AND CONSTRAINTS

We have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission.SQL injection attack is a web security attack by using SQL statements exploiting the poorly designed input elements of a web form. This compromises the confidentiality and integrity of users' sensitive data.. The attacker would have to produce an attack that directly matches either an imprecision of the analysis or a specific pattern. Moreover, in both cases, the type of attacks that could be exploited would be limited by the constraints imposed by the rest of the model that was used to match the query. It is worth noting that, in our empirical evaluation, neither false positives nor false negatives were generated.

The integrity constraints are one of the protocols that should be followed by the table's data columns. Constrains are generally used to restrict the multiple types of information that can be entered into a table to ensure the integrity of the data. Achieving the complete configuration of the constraints ensures that the data in the database is accurate and reliable to be consumed in the application. We can apply the integrity constraints at the column or table level. The table-level Integrity constraints apply to the entire table, while the column level constraints are only applied to one column.

Constraints can be defined in two ways:

Column Level

The constraints can be specified immediately after the column definition with the CREATE TABLE statement. These are called column-level constraints.

Table Level

The constraints can be specified after all the columns are defined with the ALTER TABLE statement. This is called table-level constraints.

Types of sql constraints:

Primary Key Constraints

The primary key is a set of one or more fields/columns of a table that uniquely identify each record/row in the database table. It can not accept null, duplicate values.

The primary key is a field of a database table that is used to uniquely identifies every record or the table row. And at the same time, the primary key is also one type of integrity constraint ultimately. The primary keys must have distinct values which means one of the columns of the table should have a unique value from the other. By default with the primary key, null values are not allowed in a primary key column of the table. Any table may only have a single primary key, that can be made up of one or multiple fields. By doing all these, it creates a composite primary key when the multiple fields are used as the primary key

Unique Key Constraints

A unique key is a set of one or more fields/columns of a table that uniquely identify each record/row in a database table. It is like a Primary key but it can accept only one null value and it cannot have duplicate values.

A set of one or more table fields or the column that uniquely identifies a specific record on a table is known as a "unique key" constrain. A unique key is also a type of integrity constraint which is similar to the primary key but keep in mind that a unique key will only accept one null value and should not have any duplicate values across the table.

Foreign Key Constraints

Foreign Key is a field in a database table that is the Primary key in another table. It can accept multiple nulls and duplicate values.

Foreign keys are generally used to ensure the consistency of the database data while providing some feasibility. The foreign is also a type of integrity constraint as we have the primary key. The foreign key constraint identifies any column referencing the primary key in another different table. It defines the relationship or connection between the two columns of the same table or in between the different tables.

Not Null Constraints

This constraint ensures that all rows in the database table must contain a value for the column which is specified as not null means a null value is not allowed in that column.

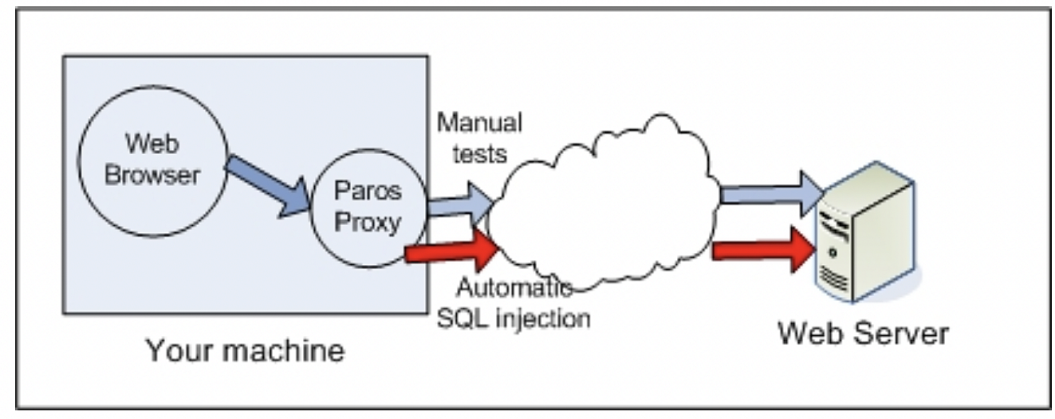
This constraint is used to ensure that all the rows of the table contain a definite value for the column which is specified as not null. IT means that the null value is not allowed.

4.3.0 PROCEDURAL DESIGN

Event-driven procedural diagram (EPC) defines the procedural organization of the company as well as the links between the objects of the data, function and organizational view. A procedural sequence of functions is illustrated using process chains. In the diagram, the start and end events of every function are specified. Events are triggers for functions or can be results of functions. The changed state of an information object can refer to the first occurrence of this information object. A process diagram consists of activities, events, and gateways, which a sequence flow puts in a flow sequence. Activities, events, and gateways are summarized under the term flow object. A procedure flowchart is a diagram that shows the sequential steps of a process and the decisions needed to make the process work. Within the chart/visual representation, every step is indicated by a shape. These shapes are connected by lines and arrows to show the movement and direction of the process. Procedure flowcharts are standardized such that anyone who has an understanding of flowcharts can look at one and know what is happening. They follow the logical flow of information so that business stakeholders have a guide as to how to fulfil processes properly.

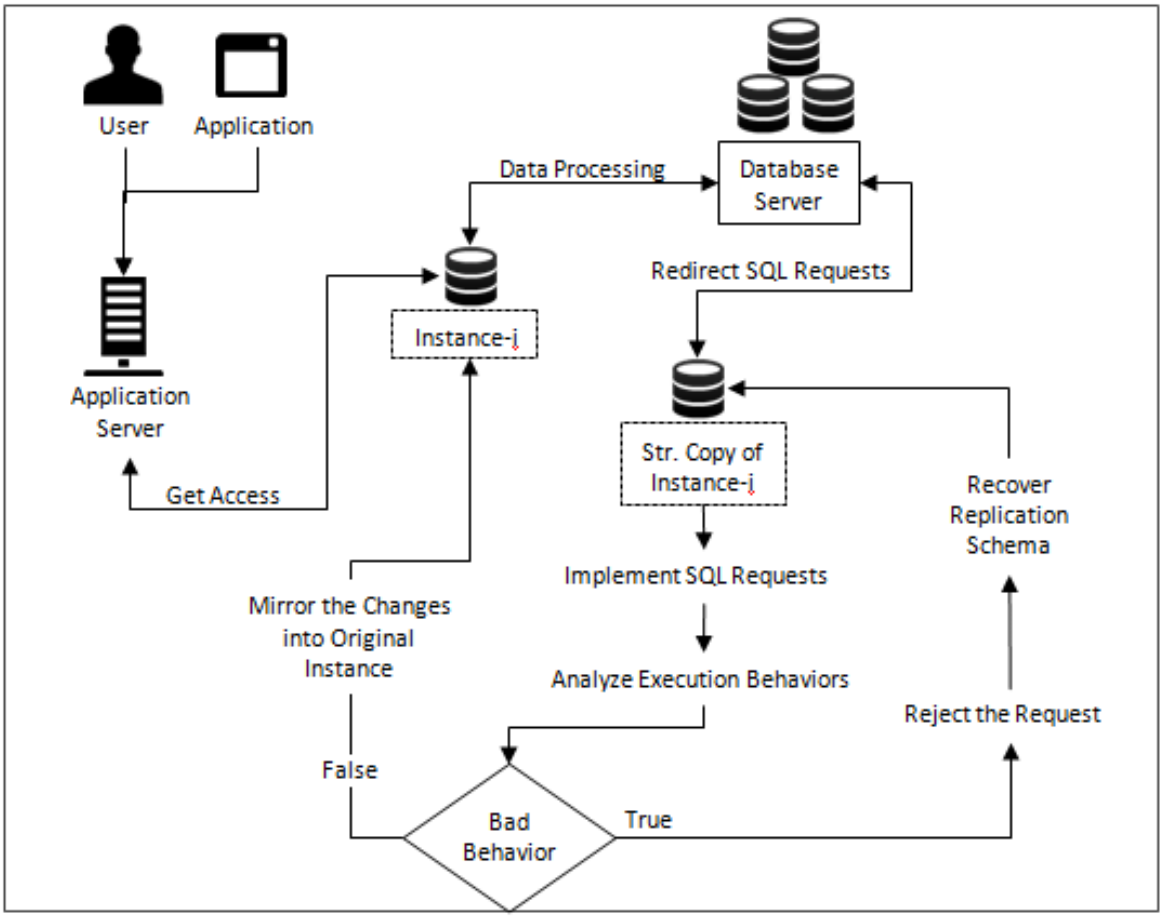
4.3.0.1 PROCEDURAL DESIGN DIAGRAM

Figure 33



PROCEDURAL DIAGRAM

Figure 34



4.3.1 LOGIC DIAGRAM

A project schedule network diagram is used for pictorial representation of logical relationships among the project activities. They are drawn to depict project dependencies or relationships between two activities and/or milestones. They are also called logical network diagrams. Logic tree diagrams are a useful problem solving tool. They enable you to break down problems – dis-aggregate them – into digestible pieces that can be solved. One key is to develop a clear problem statement. The diagram enables you to record what your initial statement, begin thinking it through, and even go back and revise your problem statement. Once you have done a rough cut logic tree diagram, you will have a visual representation of the whole problem. You will be able to visualize the problem and see relationships among the constituent parts.

Types of Logic Tree Diagrams

The following are some basic types of logic tree diagrams:

- Factor / Lever/ Component – These diagrams are helpful in beginning to understand a problem, when limited knowledge and information is available. The first level elements are factors, levels, or components of the problem. The second level elements are some further breakdown that helps dis-aggregate and understand the first level elements.
- Inductive Logic Tree – This type of tree is actually the reverse of all the others. Since inductive reasoning is deriving meaning from actual observations, the tree starts with sub-elements (observations) on the left, moving to more generality on the right, and finally the problem or thesis statement to the far right.
- Deductive Logic Tree – Deduction is the exact opposite of induction. With an obvious idea of the structure of the problem, the problem statement is on the left, the first level elements describe a component relationship to the problem, and the second level develops factors contributing to the relationships.

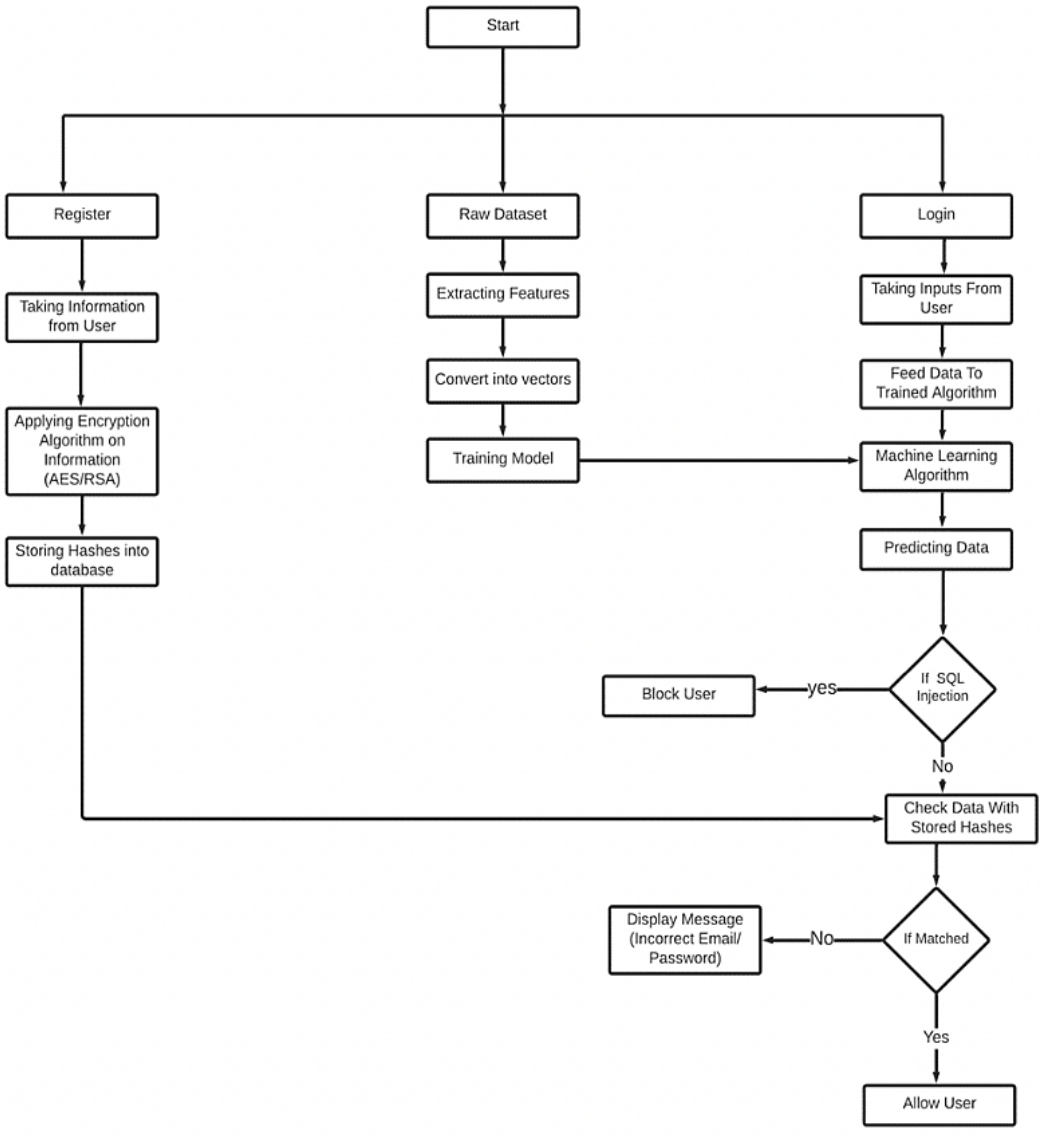
Hypothesis Tree – This is used when you know enough about your problem structure to begin to hypothesize about solutions. The first level elements are the hypotheses about possible solutions or conclusions about the problems. The second level elements are various facts and assertions to use in testing the hypotheses.

- Decision Tree – When a great deal is known, the Decision Tree can be used to clearly lay out the information logically. The tree would include an if-then structure where you could explore potential outcomes of each decision. It's not just one decision, but a series of decisions.

  The user request and its processing are done here. It involves the server side programming logic. Forms the intermediate layer between the user interface tier and the database tier.

4.3.1.1 LOGIC DIAGRAM

Figure 35



4.3.2 DATA STRUCTURES

SQL Injection is one of the main issues in database security. It affects the database without the knowledge of the database administrator. It may delete the full database or records or tables without the knowledge of the respective user or administrator. It is a technique used to exploit the database system through vulnerable web applications.

These attacks not only make the attacker breach the security and steal the entire content of the database but also, to make arbitrary changes to both the database schema and the contents. SQL injection attack could not be realized about information compromisation until long after the attack has passed. In many scenarios, the victims are unaware that their confidential data has been stolen or compromised.

Data structure diagram (DSD) is intended for description of conceptual models of data (concepts and connections between them) in the graphic format for more obviousness. Data structure diagram includes entities description, connections between them and obligatory conditions and requirements which connect them. Data structure diagram is a subspecies of the “entity-relationship” diagram type (ERD).

Entities in data structure diagrams are presented in the form of rectangles or rounded rectangles, and connections between them in the form of arrows. The description of an entity is usually placed inside the rectangle, which denotes the entity. The text description of the entity requirements connecting entities is placed near the line which denotes the connection between entities.

Figure 36

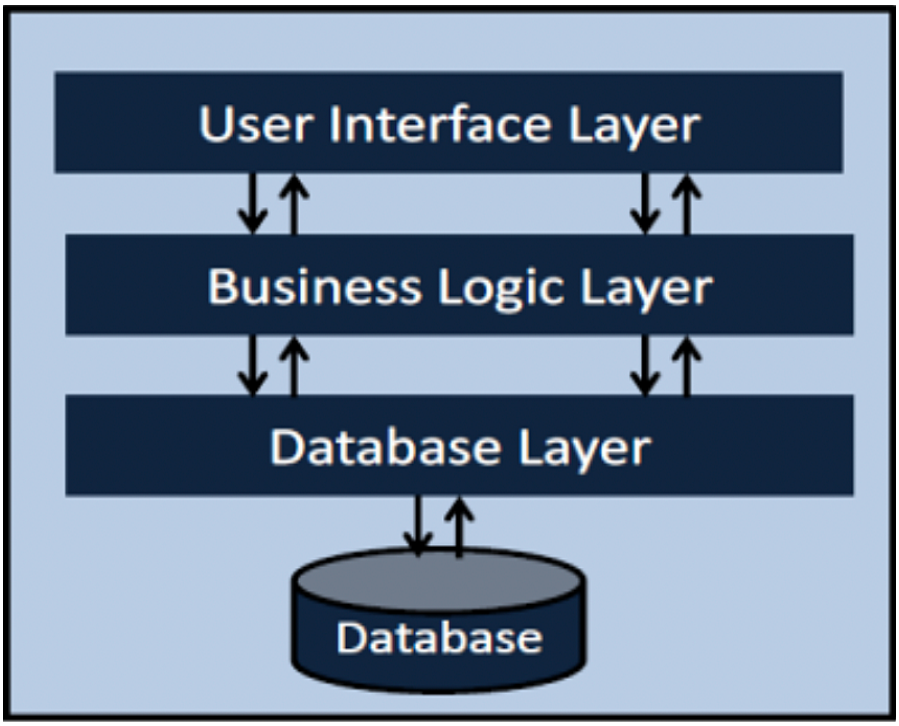
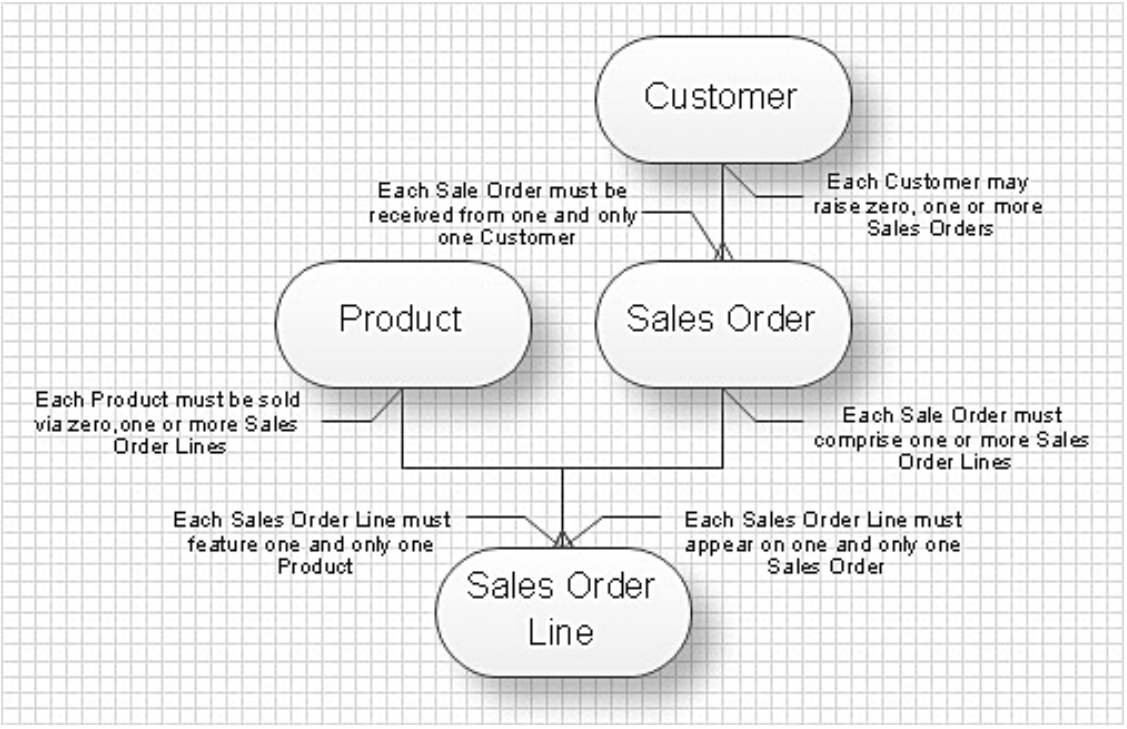


Figure 37



4.3.3 ALGORITHMS DESIGN

Details such as field and table names are required for a hacker to modify the database, so we try to propose a solution to the above problem by preventing it using a randomization-based encryption algorithm, and another solution is to use another algorithm, which is a divide-and-conquer approach to reduce time and space complexity. It has better performance and provides increased security compared to existing solutions. Also, cracking a database takes more time when techniques like dictionaries and brute force attacks are deployed. Our main goal is to provide increased security by developing a tool that prevents illegal access to the databases.

An Algorithm is a procedure to solve a particular problem in a finite number of steps for a finite-sized input.The algorithms can be classified in various ways. They are:

1. Implementation Method
2. Design Method
3. Design Approaches
4. Other Classifications

Classification by Implementation Method

5. **Recursion or Iteration:** A recursive algorithm is an algorithm which calls itself again and again until a base condition is achieved whereas iterative algorithms use loops and/or data structures like stacks, queues to solve any problem. Every recursive solution can be implemented as an iterative solution and vice versa. Example: The Tower of Hanoi is implemented in a recursive fashion while Stock Span problem is implemented iteratively.
6. **Exact or Approximate:** Algorithms that are capable of finding an optimal solution for any problem are known as the exact algorithm. For all those problems, where it is not possible to find the most optimized solution, an approximation algorithm is used. Approximate algorithms are the type of algorithms that find the result as an average outcome of sub outcomes to a problem.
7. **Serial or Parallel or Distributed Algorithms:** In serial algorithms, one instruction is executed at a time while parallel algorithms are those in which we divide the problem into sub problems and execute them on different processors. If parallel algorithms are distributed on different machines, then they are known as distributed algorithms.

The following is a list of several popular design approaches:

1. Divide and Conquer Approach: It is a top-down approach. The algorithms which follow the divide & conquer techniques involve three steps:

Divide the original problem into a set of sub problems.

- Solve every sub problem individually, recursively.
- Combine the solution of the sub problems (top level) into a solution of the whole original problem.

2. Greedy Technique: Greedy method is used to solve the optimization problem. An optimization problem is one in which we are given a set of input values, which are required either to be maximized or minimized (known as objective), i.e. some constraints or conditions.

- Greedy Algorithm always makes the choice (greedy criteria) looks best now, to optimize a given objective.
- The greedy algorithm doesn't always guarantee the optimal solution however it generally produces a solution that is very close in value to the optimal.

3. Dynamic Programming: Dynamic Programming is a bottom-up approach we solve all possible small problems and then combine them to obtain solutions for bigger problems.

4. Branch and Bound: In Branch & Bound algorithm a given sub problem, which cannot be bounded, has to be divided into at least two new restricted sub problems. Branch and Bound algorithm are methods for global optimization in non-convex problems. Branch and Bound algorithms can be slow, however in the worst case they require effort that grows exponentially with problem size, but in some cases we are lucky, and the method coverage with much less effort.

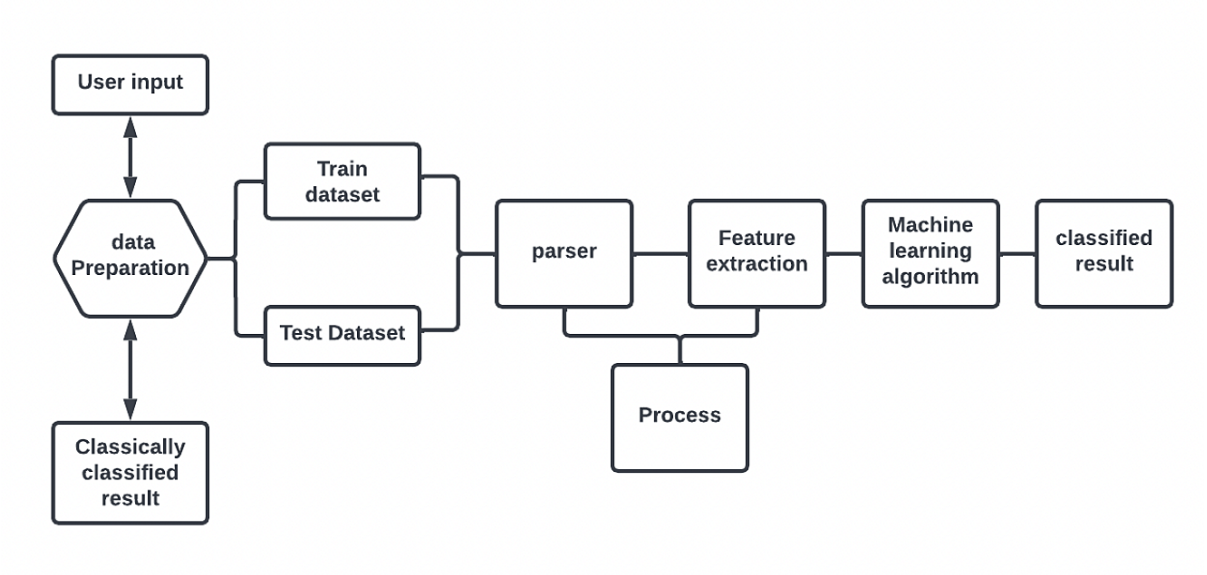
5. Randomized Algorithms: A randomized algorithm is defined as an algorithm that is allowed to access a source of independent, unbiased random bits, and it is then allowed to use these random bits to influence its computation.

6. Backtracking Algorithm: Backtracking Algorithm tries each possibility until they find the right one. It is a depth-first search of the set of possible solutions. During the search, if an alternative doesn't work, then backtrack to the choice point, the place which presented different alternatives, and try the next alternative.

7. Randomized Algorithm: A randomized algorithm uses a random number at least once during the computation make a decision.

4.3.3.1 ALGORITHMS DESIGN DIAGRAM

Figure 38



4.3.4 SEQUENCE DIAGRAM

A sequence diagram describes interactions among classes in terms of an “Exchange of messages over time”. Sequence diagrams are used to depict the time sequence of message exchanged between objects. Message can correspond to operation in class or an event trigger.

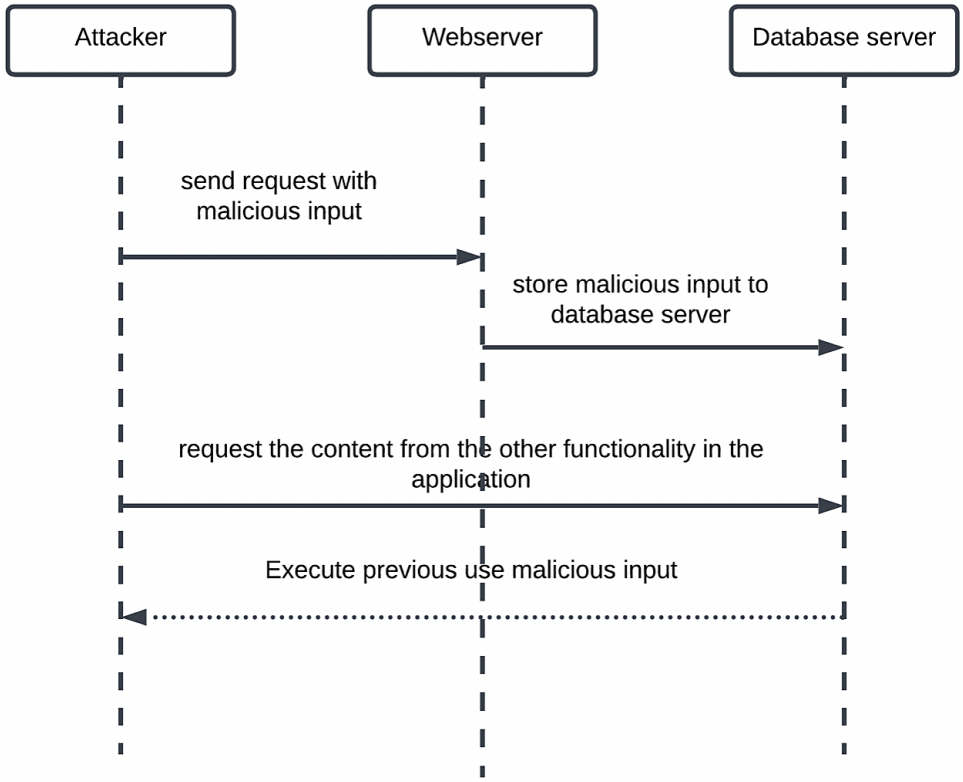
Notations of a Sequence Diagram include:

- Lifeline: It is a vertical dashed line that represents the “lifetime” of an object.
- Arrows: They indicate the flow of messages between objects.

Activation: It is a thin rectangle showing a period, during which an object is performing an action.An interaction diagram shows an interaction, consisting a set of objects and their relationship including the messages that may be dispatched among them. It is an introduction that empathizes the time ordering of messages. Graphically a sequence diagram is a table that shows objects arranged along the X-axis and messages ordered in increasing time along the Y-axis.

4.3.4.1 SEQUENCE DIAGRAM DESIGN

Figure 39



4.4 USER INTERFACE DESIGN

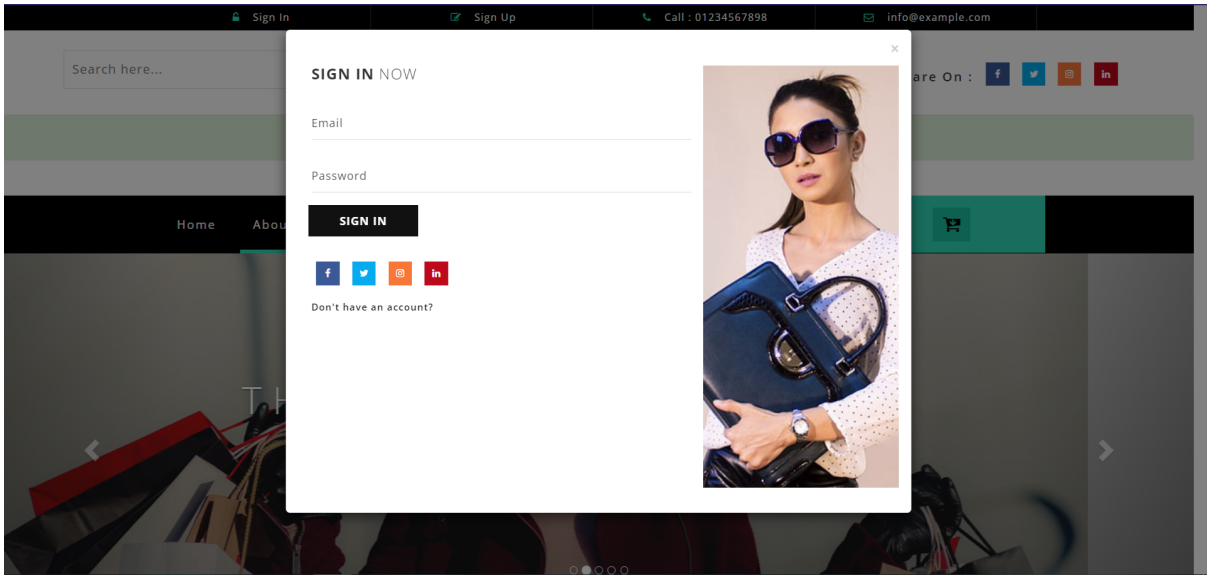
This layer forms the front end of the web application. It interacts with the other layers based on the inputs provided by the user.SQL injection attack is a web security attack by using SQL statements exploiting the poorly designed input elements of a web form. This compromises the confidentiality and integrity of users' sensitive data.

SQLIA takes place between the user interface layer and the business logic layer. To understand the essence of SQL injection let us see the following example. SELECT * from tablename WHERE user=‘ ’ and password= ‘ ’; A sample SQL statement containing two input parameters is considered .Instead of typing the actual username and password, if a hacker attempts illegally to access the database by inputting SQL statements, it is said to be a SQLi attempt. For example if the hacker inputs, ’ OR ‘1’=‘1’ - -, the statement becomes,

SELECT * from tablename WHERE user=‘’ OR ‘1’=‘1’- - and password= ‘ ’ ;

Here the user gets unauthorized access to the system because 1=1 is always true and - - indicates the statements following it are comments. Therefore, if the inputs by the user were not properly sanitized, it might lead to a critical web security attack. This describes the basic principle involved in SQL injection.

Figure 40



4.5 SECURITY ISSUES

The consequences of these attacks are merciless therefore providing an increased amount of security for users and their data becomes essential the most important vulnerability described in the open web application security projects top 10 web security issues is sql injection attack sqlia this article focuses on how you can take advantage of randomization to prevent sql injection attacks in web applications this thesis provides an overview of sql security issues the study discusses the history of sql injections as well as detection strategies it can compromise the confidentiality and security of websites that depend entirely on databases sql injection attacks sqlia are launched to gain access to databases containing sensitive data by penetrating websites with security gaps this is a critical attack that can bypass many security layers such as encryption and firewalls and is launched by exploiting input validation vulnerabilities these problems lead to too many security holes for web servers web applications and user privacy due to poor coding and validation of input fields these web applications are vulnerable to sql injection and other security issues instead of using the latest third-party frameworks web development language and versioning database server another factor that disrupts web server services can be programming sensor sockets at the production level.

1. **The physical environment:** One of the most-often overlooked steps in increasing database security is locking down the physical environment. While most security threats are, in fact, at the network level, the physical environment presents opportunities for bad actors to compromise physical devices. Unhappy employees can abscond with company records, health information or credit data. To protect the physical environment, start by implementing and maintaining strict security measures that are detailed and updated on a regular basis. Severely limit access to physical devices to only a short list of employees who must have access as part of their job. Strive to educate employees and systems technicians about maintaining good security habits while operating company laptops, hard drives, and desktop computers. Lackadaisical security habits by employees can make them an easy target.
2. **Network security:** Database administrators should assess any weak points in its network and how company databases connect. An updated antivirus software that runs on the network is a fundamental essential item. Also, ensure that secure firewalls are implemented on every server. Consider changing TCP/IP ports from the defaults, as the standard ports are known access points for hackers and Trojan horses.

3. **Server environment:** Information in a database can appear in other areas, such as log files, depending on the nature of the operating system and database application. Because the data can appear in different areas in the server environment, you should check that every folder and file on the system is protected. Limit access as much is possible, only allowing the people who absolutely need permission to get that information. This applies to the physical machine as well. Do not provide users with elevated access when they only need lower-level permission

In summary, basic database security is not especially difficult but requires constant vigilance and consistent effort. Here is a snapshot review:

4. Secure the physical environment.
5. Strengthen network security.
6. Limit access to the server.
7. Cut back or eliminate unneeded features.
8. Apply patches and updates immediately.
9. Encrypt sensitive data such as credit cards, bank statements, and passwords.
10. Document baseline configurations, and ensure all database administrators follow the policies.
11. Encrypt all communications between the database and applications, especially Web-based programs.
12. Match internal patch cycles to vendor release patterns.
13. Make consistent backups of critical data, and protect the backup files with database encryption.

4.6 TEST CASES DESIGN

4.6.1 SQL INJECTION TEST CASE METHODS :

SQL injection testing requires understanding an application's interaction with a database server to access data. Applications often communicate with a database to authenticate web forms (checking credentials against the database) and perform searches (user-submitted input can extract data from the database via SQL queries). Testers must list the input fields with values that could end up in an SQL query.

Here are some of the testing methods to help identify SQL vulnerabilities:

1. Stacked Query Testing

In the stacked query method, testers complete an SQL statement and write a new one. Testers and developers should ensure that their applications do not support stacked queries (where possible). For example, developers should avoid using a multi-query statement that enables stacked queries.

2. Based Injection Testing

Error-based injection exploits the SQL error messages displayed to users. Users attempt something that likely causes an error and retrieve data from the error message produced. Users with access to information like the names of tables can more easily compromise the underlying database.To prevent error-based injection attacks, teams must ensure the application never displays internal SQL errors to the user. The application should handle errors internally.

3. Boolean-Based Injection Testing

The Boolean method involves appending conditions to conditional statements (true in some cases, false in others). Attackers can perform several conditional queries to learn about the database. Testers can use this attack method to identify Boolean-based injection vulnerabilities.To prevent boolean-based injection attacks, teams must ensure the application never runs user input as SQL code. One way to achieve this is with prepared statements that ensure SQL does not interpret user input as code.

4. Out-of-Band (Blind) Exploit Testing

Out-of-band exploit tests are useful for assessing blind SQL injection vulnerabilities, where the attacker doesn't know anything about the operation's outcome. This method uses Database Management System (DBMS) functions to perform out-of-band connections and deliver query results to the attacker's server.

5. Time Delay Exploit Testing

Time delay exploits are useful for blind SQL injection situations. This method involves sending injected queries and monitoring the server's response time (if the conditional is true).

4.6.2 MANUAL VS AUTOMATED TESTING

Before an organization can secure its applications or websites, it is essential to know about any SQL injection vulnerabilities they contain. SQL injection is a popular attack technique that often impacts businesses severely. Testing teams should test application code for SQL injection vulnerabilities on a regular basis.Organizations should ideally test their code upon each update. Frequent testing allows security and development teams to identify and address issues introduced in code changes. Testers can look for SQL injection vulnerabilities using manual or automated methods, leveraging scanning tools to accelerate the process. Manual SQL injection testing involves manually applying user-supplied inputs to various fields to assess the application or website's input validation. It is often a time-consuming method, especially when testing many fields. Manual techniques may be inadequate to test everything thoroughly. Given the sheer scale of the task, testers can easily overlook some vulnerabilities. Organizations often use automated scanning tools to identify SQL injection vulnerabilities, allowing developers to address coding issues. Web security scanning tools offer a fast, comprehensive testing technique, returning detailed results about any vulnerabilities they discover. Testers can more easily identify affected parameters and URLs, saving time and enabling frequent software updates.

4.6.3 Standard SQL Injection Testing

Classic SQL Injection

Consider the following SQL query:

SELECT * FROM Users WHERE Username='$username' AND Password='$password'

A similar query is generally used from the web application in order to authenticate a user. If the query returns a value it means that inside the database a user with that set of credentials exists, then the user is allowed to login to the system, otherwise access is denied. The values of the input fields are generally obtained from the user through a web form. Suppose we insert the following Username and Password values:

$username = 1' or '1' = '1

$password = 1' or '1' = '1

The query will be:

SELECT * FROM Users WHERE Username='1' OR '1' = '1' AND Password='1' OR '1' = '1'

4.6.4 Few more testing methods :

1. Random SQL

Start with a simple test. Insert an SQL statement in any field and check the response.

<login>

<username><User>SELECT * from userstable</username>

<password>*</password>

</login>

Although this might seem simple, consider the following message:

Microsoft OLE DB Provider for ODBC Drivers error '80040e07' [Microsoft] [ODBC SQL Server Driver][SQL Server]Syntax error Invalid string or buffer length.

This message reveals information about the database you have. Also, it allows to identify the platform that has been used to create the web service. Keep in mind that malicious users use this information in further attacks.

2. Wildcards

In this step, enter an SQL Wildcard.

<login>

<username>*</username>

<password>*</password>

</login>

Two previous steps are similar and unlikely to result in errors. However, you should eliminate the possibility of using SQL wildcards by potential attackers.

3. Classic SQL

This test is the most common SQL injection test:

```
<login>
   <username> ' or 1=1--</username>
   <password>' or 1=1--</password>
</login>
```

The following SQL statement allows checking the login:

SELECT * FROM users WHERE username = '[username]' AND password ='[password]';

If the content of the element is not checked, the statement should look as follows:

SELECT * FROM users WHERE username = or 1=1 - -' AND password ='[password]';

This statement causes the SQL server to exclude everything after the - sign and return the first user in the database. With this statement, a potential attacker may be able to log in. Since the first user is often an administrator, the necessity of protection should be obvious.

4. Empty strings

```
<login>
   <username> ' or ''='</username>
   <password>' or ''='</password>
</login>
```

This request results in the following SQL and returns all records in the database and provides the possibility of gaining access to the service:

SELECT * FROM users WHERE username ='' or ''='' and Password = '' or ''=''

5. Type conversions

Try to expose the database by sending type conversions which will fail in the database.

```
<login>
   <username>CAST('smartbear' AS SIGNED INTEGER)</username>
   <password>pass12bwQ19!</password>
</login>
```

This SQL statement is aimed at exposing the database by sending an error message. As mentioned above, any information about the database or the application platform gives additional information on specific vulnerabilities for that environment.

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 IMPLEMENTATION APPROACHES

First, whenever anyone wants to access the site, the user has to login to see the details of the items. So, the user has to give the respective login credentials like the username and password, and the password is converted into its hashed format using the MD5 algorithm, and it checks with the hashed password stored in that username record in the database. if both the hashes match, the user logins to the system. If the hashes do not match or if the username is not present, the user cannot login to the system. If an attacker wants to access the system, he/she will not be having username and password so they will write some malicious code to get into the database. So if any hacker inserts SQL injection queries in the input field like password, they can access the system and read all data from the database.

Here, the MD5 algorithm is used so that if any malicious injection code is written the hacker cannot access the system. Here the hacker tries multiple SQL injection queries, but he cannot login to the system. Steps are as follows :

- Create a web application
- Conduct SQLIA before implementing a cryptography algorithm.
- Implementation of cryptographic algorithm on the web application
- Conduct SQLIA after implementing Observe the changes

5.2 CODING DETAILS AND CODE EFFICIENCY

5.2.1 CODE EFFICIENCY

Using the concept of Command Injection Attack. It allows programs to protect themselves from SQL injection attacks. Static analysis, dynamic analysis, and intelligent code re-engineering are key components of the security testing process for protecting existing properties.A new method of detecting SQL injection sites. Runtime testing for potentially exploitable vulnerabilities is part of their strategy, as is later application code review to ensure protection. By altering the original SQL statement, the vulnerability attack established its own targets.

Using this method, new vulnerabilities were discovered. The most important finding was that SQL injections may be detected and prevented. They show that by fostering intelligent applications, an intelligent system can help to prevent cyber-attacks SVMs were used to investigate both original and suspect queries. For classification, a dataset of various sizes is used. Precision, detection time, training time, TPR, TNR, FPR, FNR, and graphical details on our system's performance are all displayed here.

Our method has the highest level of output accuracy, at 96.5 percent.After implementation of the AES encryption, the data is encrypted and the data of the customers is safe. Databases are susceptible to unauthorized access which can lead to tampering of data, integrity issues, modification and deletion of data etc. These types of attacks can be prevented by using AES encryption, as is shown in this project. The data stored in the database is encrypted using a secret key.

5.3.0 TESTING APPROACH

5.3.0.1 BLACK BOX TESTING

Black Box Testing is a testing the software without any knowledge of the inner workings, structure or language of the module being tested. This type of testing is based entirely on software requirements and specification. In black box testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program. Black box tests, like most other kinds of tests, must be written from a definitive source document, such as specification or requirements document. It gives abstraction from code and focuses on testing effort on the software system behavior. It is a testing which the software under test is treated as a black box you cannot “see” into it. The tests provide inputs and respond to output without considering how the software works. The main focus of black box testing is on the validation of your functional requirement. And it also facilitates testing communication amongst modules.

Black Box Testing :

- Conducted without any prior intel of the target system.
- Only tests the exposed environment.
- Not at all in-depth.
- Consists of guesswork, and endless hit & miss sessions.
- Automation is heavily used.
- ETAs are unpredictable. Can be very fast or take months on end.
- Cheaper to carry out.

Types of black box testing

Black box testing can be applied to three main types of tests: functional, non-functional, and regression testing.

- **Functional Testing**

Black box testing can test specific functions or features of the software under test. For example, checking that it is possible to log in using correct user credentials, and not possible to log in using wrong credentials. Functional testing can focus on the most critical aspects of the software (smoke testing/sanity testing), on integration between key components (integration testing), or on the system as a whole (system testing).

- **Non-Functional Testing**

Black box testing can check additional aspects of the software, beyond features and functionality. A non-functional test does not check "if" the software can perform a specific action but "how" it performs that action. Black box tests can uncover if software is:

1. Usable and easy to understand for its users
2. Performant under expected or peak loads
3. Compatible with relevant devices, screen sizes, browsers or operating systems
4. Exposed to security vulnerabilities or common security threats

- **Regression Testing**

Black box testing can be used to check if a new version of the software exhibits a regression, or degradation in capabilities, from one version to the next. Regression testing can be applied to functional aspects of the software (for example, a specific feature no longer works as expected in the new version), or non-functional aspects (for example, an operation that performed well is prolonged in the new version).

5.3.0.2 GREY BOX TESTING

While white box testing assumes the tester has complete knowledge, and black box testing relies on the user's perspective with no code insight, grey box testing is a compromise. It tests applications and environments with partial knowledge of internal workings. Grey box testing is commonly used for penetration testing, end-to-end system testing, and integration testing. You can perform grey box testing using Interactive Security Testing (IAST) tools. IAST tools combine DAST and Static Application Security Testing (SAST), which is used in white box testing to evaluate static code. IAST tools enable you to combine the work of testers and developers and increase test coverage efficiently. For example, you can perform more directed tests which focus on areas or user paths that are likely to contain flaws. You can implement gray box testing as a form of penetration testing that is unbiased and non-obtrusive. In these tests, the tester typically knows what the internal components of an application are but not how those components interact. This ensures that testing reflects the experiences of potential attackers and users. Gray box testing is most effective for evaluating web applications, integration testing, distributed environments, business domain testing, and performing security assessments.

The Gray Box Testing Process

In gray box testing, the tester is not required to design test cases. Instead, test cases are created based on algorithms that evaluate internal states, program behavior, and application architecture knowledge. The tester then carries out and interprets the results of these tests. When performing gray box testing, you take the following steps:

- Identify and select inputs from white and black box testing methods.
- Identify probable outputs from these inputs.
- Identify key paths for the testing phase.
- Identify sub-functions for deep-level testing.
- Identify inputs for sub-functions.
- Identify probable outputs from sub-functions.
- Execute sub-function test cases.
- Assess and verify outcomes.

Gray Box Testing Techniques

Gray box testing techniques are designed to enable you to perform penetration testing on your applications. These techniques enable you to test for insider threats, such as employees attempting to manipulate applications, and external users, such as attackers attempting to exploit vulnerabilities.

- **Matrix Testing**

Matrix testing is a technique that examines all variables in an application. In this technique, technical and business risks are defined by the developers and a list of all application variables is provided. Each variable is then assessed according to the risks it presents. You can use this technique to identify unused or un-optimized variables.

- **Regression Testing**

Regression testing is a technique that enables you to verify whether application changes or bug fixes have caused errors to appear in existing components. You can use it to ensure that modifications to your application are only improving the product, not relocating faults. When performing regression testing, you need to recreate your tests since inputs, outputs, and dependencies may have changed.

- **Pattern Testing**

Pattern testing is a technique that evaluates past defects to identify patterns that lead to defects. Ideally, these evaluations can highlight which details contributed to defects, how the defects were found, and how effective fixes were. You can then apply this information to identifying and preventing similar defects in new versions of an application or new applications with similar structures.

5.3.0.3 WHITE BOX TESTING

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It has a purpose. It is used to test areas that cannot be reached from a black box level. White box testing is testing of a software solution’s internal structure, design and coding. In this type of testing the code is visible to the tester. It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security. White box testing is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing. It is usually performed by developers.

It is one of two parts of the Box Testing approach to software testing. Its counterpart, Black box testing, involves testing from an external or end-user type perspective. On the other hand, White box testing is based on the inner workings of an application and revolves around internal testing. The term "White Box" was used because of the see-through box concept. The clear box or White Box name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings.

What is white box Security testing?

In order to understand the white box security we need to understand security testing and white box testing. We have already mentioned about security testing. The next section helps to understand White box testing. White box testing: A white box testing is an approach to validate and verify the design of the system, the underlying code logic involved, the data flow, the control flow, coding practices and error and exception handling capabilities of a system. So, white box security testing (WBST) can be defined as an approach to verify the code implementation as per design, to validate the implemented security functionality and to uncover the vulnerabilities that can be exploited.

Test Strategy for WBST (White Box Security Testing)

As exhaustive testing is not possible and also is not cost effective. Risks identified in the Risk Analysis document are ranked and prioritized and effort is made to focus on high risk areas.Test strategy should be designed in such a way that optimum coverage is achieved within stimulated time and budget. Test strategy includes identifying testing scope, testing techniques, coverage metrics, test staff skill requirement and test environment. Test strategy should focus on achieving great level of test effectiveness and efficiency.

5.3.1 UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases. Unit testing is commonly automated but may still be performed manually. Software Engineering does not favor one over the other but automation is preferred. A manual approach to unit testing may employ a step-by-step instructional document.

Under the automated approach-

- A developer writes a section of code in the application just to test the function. They would later comment out and finally remove the test code when the application is deployed.
- A developer could also isolate the function to test it more rigorously. This is a more thorough unit testing practice that involves copy and paste of code to its own testing environment than its natural environment. Isolating the code helps in revealing unnecessary dependencies between the code being tested and other units or data spaces in the product. These dependencies can then be eliminated.
- A coder generally uses a UnitTest Framework to develop automated test cases. Using an automation framework, the developer codes criteria into the test to verify the correctness of the code. During execution of the test cases, the framework logs failing test cases. Many frameworks will also automatically flag and report, in summary, these failed test cases. Depending on the severity of a failure, the framework may halt subsequent testing.

Test objectives :

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested :

- Verify that the entries are of the correct format
- No duplicate entries should be allowed

Unit Testing Best Practices

- Unit Test cases should be independent. In case of any enhancements or changes in requirements, unit test cases should not be affected.
- Test only one code at a time.
- Follow clear and consistent naming conventions for your unit tests
- In case of a change in code in any module, ensure there is a corresponding unit Test Case for the module, and the module passes the tests before changing the implementation.
- Bugs identified during unit testing must be fixed before proceeding to the next phase in SDLC
- Adopt a “test as your code” approach. The more code you write without testing, the more paths you have to check for errors.

5.3.2 INTEGRATED TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or one step up software applications at the company level interact without error. Integration testing is known as the second level of the software testing process, following unit testing. Integration testing involves checking individual components or units of a software project to expose defects and problems to verify that they work together as designed. As a rule, the usual software project consists of numerous software modules, many of them built by different programmers. Integration testing shows the team how well these disparate elements work together.

Advantages of Integration Testing

- Integration testing ensures that every integrated module functions correctly
- Integration testing uncovers interface errors
- Testers can initiate integration testing once a module is completed and doesn't require waiting for another module to be done and ready for testing
- Testers can detect bugs, defects, and security issues
- Integration testing provides testers with a comprehensive analysis of the whole system, dramatically reducing the likelihood of severe connectivity issues

5.3.2.1 TYPES OF INTEGRATION TESTING

- **Big Bang Method**

This method involves integrating all the modules and components and testing them at once as a unit. This method is also known as non-incremental integration testing.

- **Bottom-Up Method**

This method requires testing the lower-level modules first, which are then used to facilitate the higher module testing. The process continues until every top-level module is tested. Once all the lower-level modules are successfully tested and integrated, the next level of modules is formed.

- **Hybrid Testing Method**

This method is also called "sandwich testing." It involves simultaneously testing top-level modules with lower-level modules and integrating lower-level modules with top-level modules, and testing them as a system. So, this process is, in essence, a fusion of the bottom-up and top-down testing types.

- **Incremental Approach**

This approach integrates two or more logically related modules, then tests them. After this, other related modules are gradually introduced and integrated until all the logically related modules are successfully tested. The tester can use either the top-down or bottom-up methods.

- **Stubs and Drivers**

These elements are dummy programs used in integration testing to facilitate software testing activity, acting as substitutes for any missing models in the testing process. These programs don’t implement the missing software module’s entire programming logic, but they do simulate the everyday data communication with the calling module.

- **Top-Down Approach**

Unlike the bottom-up method, the top-down approach tests the higher-level modules first, working the way down to the lower-level modules.

5.3.3 BETA TESTING

Beta testing is an opportunity for real users to use a product in a production environment to uncover any bugs or issues before a general release. Beta testing is the final round of testing before releasing a product to a wide audience. The objective is to uncover as many bugs or usability issues as possible in this controlled setting. Beta testers are “real” users and conduct their testing in a production environment running on the same hardware, networks, etc., as the final release. This also means it’s the first chance for full security and reliability testing because those tests can’t be conducted in a lab or stage environment.Beta tests can either be open or closed. In an open test, anyone can use the product and is usually presented with some messaging that the product is in beta and given a method for submitting feedback. In closed beta, the testing is limited to a specific set of testers, which may be composed of current customers, early adopters, and/or paid beta testers. Sometimes they are conducted by diverting a certain percentage of users to the beta site instead of the current release.Testing can either last for a set period or run until new issues stop being reported and all-important ones have been addressed.Beta testing is usually black-box testing, meaning test participants don’t know anything about the backend and don’t have access to source code. Since beta testing happens most of the time at the end user’s side, it cannot be a controlled activity. Beta testing can provide precious insights – genuine scenarios of interactions with a product. A Beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing helps in minimization of product failure risks and it provides increased quality of the product through customer validation. It is the last test before shipping a product to the customers. One of the major advantages of beta testing is direct feedback from customers.

Characteristics of Beta Testing:

1. Beta Testing is performed by clients or users who are not employees of the company.
2. Reliability, security, and robustness are checked during beta testing.
3. Beta Testing commonly uses black-box testing.
4. Beta testing is carried out in the user’s location.
5. Beta testing doesn’t require a lab or testing environment

Types of Beta Testing: There are different types of beta testing:

1. Traditional Beta testing: Product is distributed to the target market and related data is gathered in all aspects. This data can be used for Product improvement.
2. Public Beta Testing: Product is released publicly to the world through online channels and data can be collected from anyone. Based on feedback, product improvements can be done. For example, Microsoft conducted the largest of all Beta Tests for its operating system Windows 8 before officially releasing it.
3. Technical Beta Testing: Product is released to a group of employees of an organization and collects feedback/data from the employees of the organization.
4. Focused Beta Testing: Software product is released to the market for collecting feedback on specific features of the program. For example, important functionality of the software.
5. Post-release Beta Testing: Software product is released to the market and data is collected to make improvements for the future release of the product.

Advantages of Beta Testing:

- It reduces product failure risk via customer validation.
- Beta Testing allows a company to test post-launch infrastructure.
- It helps in improving product quality via customer feedback.
- Cost-effective compared to similar data gathering methods.
- It creates goodwill with customers and increases customer satisfaction.

5.4 MODIFICATIONS AND IMPROVEMENTS

As Modifications work, we want to evaluate methods using different web-based application scripts with public domain to achieve great accuracy in SQL injection prevention approaches. Integrate SQLiX with nikto HTTP scanner, HTTP scanning proxies, and with metasploit will help to detect other web vulnerabilities. Also add Modifications to dump vulnerable databases and database schema.

In this example, the variables username and password are retrieved from the HTTP POST that was submitted by the user. The strings are taken as-is and inserted, via string interpolation, into the query string. Since no validation is done on the input, the user can enter characters that will modify the structure of the query. For example, if the user enters ‘ or 1=1;

for the username, and nothing for the password, the variable sql will now equal:

$sql = “select USER_ID from USERS where USERNAME=’’ or 1=1; #’ and PASSWORD=’$password’;”

In the MySQL database engine, the “#” sign is a comment, so everything that comes after it is ignored in the query. There are no users with a blank username, but the condition “1=1” is always true, so the query will always succeed, returning all user IDs in the database. The subsequent code will likely only check that at least one record was returned, and it will likely grab just the first ID, which in most cases, will be that of the administrative user.

Doing SQL injection manually requires a fair bit of knowledge of how SQL works. On top of that, there are many different SQL engines, each with slight variations in syntax, such as PostgreSQL, MySQL, Microsoft SQL Server, Oracle, IBM DB2, and others. SQL Injection “cheat sheets” can help pen testers figure out the required syntax for testing a web application, but SQL Injection is still a very time-consuming attack to carry out.

5.5 TEST CASES

The first step in this test is to understand when the application interacts with a DB Server in order to access some data. Typical examples of cases when an application needs to talk to a DB include:

- Authentication forms: when authentication is performed using a web form, chances are that the user credentials are checked against a database that contains all usernames and passwords (or, better, password hashes).
- Search engines: the string submitted by the user could be used in a SQL query that extracts all relevant records from a database.
- E-Commerce sites: the products and their characteristics (price, description, availability, etc) are very likely to be stored in a database.

5.5.1 TEST CASES

Method	Strength	Weakness	Type of attack prevented
RC4 and blowfish encryption [25]	Proposed mechanism has faster performance, requires no modification, and has less execution overhead	Unable to protect against second order SQL injection	Bypass authentication, unauthorized knowledge of database, and injected additional query
AES and MD5 algorithms [26]	Combination of AES and MD5 algorithms had better performance in terms of security in preventing SQL injection attack	Independent implementation of these algorithms is inefficient in improving web security	Bypass authentication, unauthorized knowledge of database, injected additional query, and second order SQL injection
Password encryption [27]	Mitigate a huge domain of SQL injection attack	Does not focus on advanced SQL injection attack methods such as hybrid and second-order attack for future studies	Unauthorized access at database storage level
AES-128 and token-base64 [28]	Improved the web service load time performance with a 31.26% increase in response time with 95% of the file size web services becomes greater	Does not improved the average grade in web services performance	Tautologies, Union, illegal/logical incorrect queries, piggyback queries, inference, and stored procedures

5.6 CODING SEGMENT

1. Function for sql injection code:

```
# mymodel = load_model('models/my_model_additional_data.h5')
# myvectorizer = pickle.load(open("models/vectorizer_additional_data", 'rb'))


def clean_data(input_val):
    input_val=input_val.replace('\n', '')
    input_val=input_val.replace('%20', ' ')
    input_val=input_val.replace('=', ' = ')
    input_val=input_val.replace('((', ' (( ')
    input_val=input_val.replace('))', ' )) ')
    input_val=input_val.replace('(', ' ( ')
    input_val=input_val.replace(')', ' ) ')
    input_val=input_val.replace('1 ', 'numeric')
    input_val=input_val.replace(' 1', 'numeric')
    input_val=input_val.replace("'1 ", "'numeric ")
    input_val=input_val.replace(" 1'", " numeric'")
    input_val=input_val.replace('1,', 'numeric,')
    input_val=input_val.replace(" 2 ", " numeric ")
    input_val=input_val.replace(' 3 ', ' numeric ')
    input_val=input_val.replace(' 3--', ' numeric--')
    input_val=input_val.replace(" 4 ", ' numeric ')
    input_val=input_val.replace(" 5 ", ' numeric ')
    input_val=input_val.replace(' 6 ', ' numeric ')
    input_val=input_val.replace(" 7 ", ' numeric ')
    input_val=input_val.replace(" 8 ", ' numeric ')
    input_val=input_val.replace('1234', ' numeric ')
    input_val=input_val.replace("22", ' numeric ')
    input_val=input_val.replace(" 8 ", ' numeric ')
    input_val=input_val.replace(" 200 ", ' numeric ')
    input_val=input_val.replace("23 ", ' numeric ')
    input_val=input_val.replace("'1', "'numeric')
```

```
    input_val=input_val.replace('1'', '''numeric')
    input_val=input_val.replace(''7659'', 'numeric')
    input_val=input_val.replace(" 37 ", ' numeric ')
    input_val=input_val.replace(" 45 ", ' numeric ')
    return input_val
```

2. Code for registering on website :

```
@app.route("/register", methods = ['GET', 'POST'])
def register():
    if request.method == 'POST':
        #Parse form data
        # print("hii register")
        email = request.form['email']
        password = request.form['password']
        username = request.form['username']
        add = request.form['add']
        postc = request.form['postc']
        mob = request.form['mob']
        productId = 0
        print(password)
        print(email)
        print(username)
        print(add)
        print(postc)
        print(mob)
        print(productId)
        allData = [email,password,username,add,postc,mob]
        DetectionOp = [predict_sqli_attack(i) for i in allData]
        print("DetectionOp")
        print(DetectionOp)
        email_part = DetectionOp[0][0]
        pass_part = DetectionOp[1][0]
```

```
        username_part = DetectionOp[2][0]
        add_part = DetectionOp[3][0]
        postc_part = DetectionOp[4][0]
        mob_part = DetectionOp[5][0]

        try:
            if "SQL Injection" == email_part or "SQL Injection" == pass_part or "SQL Injection" == username_part or "SQL Injection" == add_part or "SQL Injection" == postc_part or "SQL Injection" == mob_part:
                msg = "SQL Injection detected"
                print("we are in injection mode")
                return render_template('login.html',FinalMsg=msg)
            else:
                con = dbConnection()
                cursor = con.cursor()
                sql1 = "INSERT INTO users (username, email, pass) VALUES (%s, %s, %s)"
                val1 = (username, email, password)
                cursor.execute(sql1, val1)
                print("query 1 submitted")
                sql2 = "INSERT INTO address (username, email, address, postcode, mobile) VALUES (%s, %s, %s, %s, %s)"
                val2 = (username, email, add, str(postc), str(mob))
                cursor.execute(sql2, val2)
                print("query 2 submitted")
                # sql3 = "INSERT INTO kart1 (username, productId) VALUES (%s, %s)"
                # val3 = (username, int(productId))
                # cursor.execute(sql3, val3)
                # print("query 3 submitted")
                con.commit()
                FinalMsg = "Congrats! Your account registerd successfully!"
        except:
            con.rollback()
            msg = "Database Error occured"
```

```
            print(msg)
            return render_template("login.html", error=msg)
        finally:
            dbClose()
        return render_template("login.html",FinalMsg=FinalMsg)
    return render_template("login.html")
```

3.Code for Login

```
@app.route("/", methods = ['POST', 'GET'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        allData = [email,password]

        DetectionOp = [predict_sqli_attack(i) for i in allData]
        print("DetectionOp")
        print(DetectionOp)
        email_part = DetectionOp[0][0]
        pass_part = DetectionOp[1][0]
        if "SQL Injection" == email_part or "SQL Injection" == pass_part:
            msg = "SQL Injection detected"
            print("we are in injection mode")
            return render_template('login.html',FinalMsg=msg)
```

```
        else:
            con = dbConnection()
            cursor = con.cursor()
            result_count = cursor.execute('SELECT * FROM users WHERE email = %s AND
pass = %s', (email, password))
            result = cursor.fetchone()
            print("result")
            print(result)
            if result_count>0:
                print("len of result")
                session['uname'] = result[1]
                session['userid'] = result[0]
                return redirect(url_for('root'))
            else:
                return render_template('login.html')
    return render_template('login.html')
```

4.Code for connecting database

```
def dbConnection():
    try:
        connection = pymysql.connect(host="localhost", user="root", password="root",
database="sqlinjection",charset='utf8')
        return connection
    except:
        print("Something went wrong in database Connection")
def dbClose():
    try:
        dbConnection().close()
    except:
        print("Something went wrong in Close DB Connection")
con=dbConnection()
cursor=con.cursor()
```

CHAPTER 6

RESULTS AND DISCUSSION

6.1 TEST REPORTS

The main goal is to prevent an attacker from successfully logging in. successful injection of a SQL query by an attacker. So here we used a cryptographic algorithm to prevent this. details of users with passwords that are stored in the database. Here, the password is stored in a hashed format in the database. Here, when an attacker tries multiple SQL injection queries but cannot log into the system, The results emphasize the importance of cryptographic algorithms in system security, without which it is impossible to prevent attackers from attacking. All the above test cases passed successfully. No defects were found.

To detect SQLIAs, SQLUnitGen instruments ap- application byte code with the AMNESIA runtime monitor. When the SQL query constructed from a test case and the SQL query model built by AMNESIA statically do not match, the AMNESIA runtime monitor raises a SQLIA Exception.

To help programmers to easily identify vulnerable locations in the program, SQLUnitGen generates a textual test result summary and a graphical test result summary. A textual test result summary shows the test cases that succeeded or failed for each method. Test success in a test case means that SQLIA was not detected from the test case. Test failure in a test case means that SQLIA was detected from the test case.

A graphical test result summary shows a colored call graph indicating the flow of input between methods, and the demonstrated vulnerability of each method. On the call graph, methods are represented as ovals. A green oval indicates that all the test cases for the method succeeded. A red oval indicates that all the test cases for the method failed. A yellow oval indicates that some of the test cases for the method succeeded and that some of them failed. A black oval indicates there were no test cases for the method.

The numbers after a method name indicates the number of successful test cases and failed test cases. For example, all the test cases for Login succeeded, which means the method is not vulnerable to the test input. All the test cases for isRegistered failed, which means the method was vulnerable to the test input. Partial success can happen, as in getUserInfo, when input filtering is performed only for some of the arguments, or input filter does not check some of vulnerable characters.

Even though isRegistered failed in all the test cases, isRegistered can be considered not to be vulnerable because the caller method Login is the only path to isRegistered, and Login filtered all the vulner- able input successfully. If isRegistered can be called with user input without passing through Login, a programmer should consider putting input filters between isRegistered and the hotspot.

A textual test result summary also provides the query used for each test case. Therefore, programmers can easily identify why the test case failed and which user input is vulnerable using the textual test result summary and attack test cases with concrete attack input values. Based on the information, programmers can add input filters in a proper location in a program or use safer APIs, or combine the two approaches.

6.2 USER DOCUMENTATION

User input might take a circuitous path from the user interface through one or more methods that may or may not be input filter methods, and ultimately to the SQL command to be executed. Our approach traces the flow of the input values that are used for a SQL query by using the AMNESIA SQL query model and string argument instrumentation. Based on the input flow analysis, we generate test attack input for the method arguments used to construct a SQL query.

There are three major challenging enhancements that I have completed successfully in this project.

i) enhanced the crawler to handle HTTP post methods and fill forms automatically.

ii) Created a graphical user interface (GUI) for SQLiX.

iii) Added a module to detect cross-site scripting (XSS) attacks.

Our initial set of nine attack patterns included in SQLUnitGen v0.5 were collected from published black lists [2, 17, 19, 21]. These initial nine are only a small set of commonly-referenced attacks. To generate attack inputs, we used the following attack patterns for character type variables that require a single quotation mark:

AP1: 1’ OR ‘1’=’1

AP2: 1’OR‘1’=’1

AP3: 1'; exec master..xp_cmdshell

'dir';--

AP4: 1'; drop table sqliatest; create

table sqliatest (name varchar(10))-- *AP5*: 1'; delete from sqliatest;--

AP2 is similar to AP1, but AP2 does not have a space between 1’ and OR. This attack pattern was added to test if the program tries to detect SQL injection attack based on a wrong regular expression, for example, “’ OR”. In some cases, the application cannot just block a single quotation mark, because a single quotation mark in a name (e.g. O’Reilly) is legitimate data. Even though the regular expression “’ OR” can detect AP1, this regular expression will fail to detect AP2. In MySQL and PostgreSQL, ‘-- ' makes remaining content in a SQL query a comment so that the remaining content is ignored.

We used the following attack patterns for integer type variables:

AP6: 1 OR 1=1--

AP7: 1;exec master..xp_cmdshell 'dir'; -- *AP8*: 1; drop table sqliatest; create table sqliatest (name varchar(10)) -- *AP9*: 1; delete from sqliatest;--

User input from web interface is transmitted as string data types. Even though the input is for an integer type column in a database table, the string type user input can be used to construct a valid SQL query as long as the user input consists of numerical digits. Therefore, AP6-9 can be used as attack input value for an integer type column if an application does not convert the user input the integer type.

AP1, AP2, AP6 are intended to detect attacks based on tautology as explained in Section 2.1. Some DBMSs allow multiple SQL statements in one query execution by default or by option. AP3, AP4, AP5, AP7, AP8, and AP9 are intended to detect attacks involving multiple SQL statements. AP4, AP5, AP8, and AP9 show that attackers can arbitrarily manipulate data in a database. With AP3, attackers can execute system commands using built-in stored procedure. Microsoft SQL Server is known to be vulnerable to AP3 [2]. For repeatable test, AP8 creates table sqliatest after the table is dropped.

User input APIs

Our approach only can test user input passed as method arguments. If a user input API and a hotspot that uses the input reside in the same method, test case cannot be generated properly. One way to solve this problem is to give warning to programmers when the user input is used at a hotspot without being passed as a method argument. Such user input can be identified when an augmented SQL query model has no tagged information for a variable in a SQLquery.

Underlying techniques

Our approach is limited by the ability of query-model building of AMNESIA and its underlying string analyzer, JSA. SQLUnitGen inherits the possibility of false negatives and false positives of AMNESIA due to the over- approximation of underlying string analysis . In addition, the current implementation of JSA does not support the analysis of the string in a class field as it does for local string variables for its own purpose. Therefore, when a method argument is a class and a member field of the class is used to construct a SQL query, our approach does not trace user input precisely. SQLUnitGen is also limited by the scalability of AMNESIA and JSA due to the possible large number of states and transitions in the generated automata .

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

We have proposed a solution to prevent SQL injection on e-commerce sites. Using the Message Digest 5 (MD-5) method, we converted the plaintext password to cipher text. The results demonstrate the importance of cryptographic methods in system security. As a result, data security is critical to the smooth operation of the website. because there are many attackers trying to gain access to systems and steal data. SQL injection is one of the most important web security threats that must be addressed to increase the security of users and their data. This article deals with an application-specific randomized encryption algorithm for its detection and prevention.

Furthermore, its effectiveness was compared with other existing techniques, and its effectiveness was quantified. That's why we looked at this web security weakness and analyzed its attack types. The SQLA security threat is really high, and it is very necessary to protect user data in web applications because it is very confidential and sensitive. SQL is one of the best web security scanners for finding SQL injection vulnerabilities from a webpage, although it is not enough to list other possible types of web attacks. It is very efficient in terms of speed, injecting a large number of injections, and injecting a custom function. You can build a side-by-side comparison of the above test cases.

I enhanced this tool by adding a graphical user interface, HTTP POST form autofill, and XSS attacks. The GUI helps novice users tryout all combinations of attacks without memorizing all the options, and the HTTP Post method increases the number of injections injected into the database. We presented an automatic test case generation technique to identify SQL injection vulnerability. We used a combined approach of static analysis, runtime detection and automatic testing.

Static analysis using AMNESIA and string argument instrumentation is used to trace the user input to a vulnerable location (hotspot) in an application. In our evaluation, the prototype tool SQLUnitGen v0.5 had no false positives and small number of false negatives.

Many of the SQLIA vulnerabili- ties that can be identified with SQLUnitGen can be addressed via the use of PreparedStatements. Since many legacy applications were not written using PreparedStatements, SQLUnit- Gen can be particularly helpful in finding and addressing SQLIA vulnerabilities in legacy code as well as new code written that does not take advantage of newer, safer techniques.

7.1.1 SIGNIFICANCE OF THE SYSTEM

The results demonstrate the significance of cryptographic methods in ensuring the security of the system. As a result, data security is critical for the website's smooth operation. because there are numerous attackers attempting to get access to systems and steal data.

1.Safe Authentication

2. Reduce the time and space complexity.

3.More secure online transactions are possible.

The major issue with web application security is SQL injection, which can give attackers unrestricted access to the databases that underlie web applications :

1. highly automated approach for protecting Web applications from SQLAs.
2. provides DBMS auditing methods to find out the transactions.
3. does not require customized and complex runtime environments.
4. requires no modification of the runtime system.

7.2 LIMITATIONS OF THE SYSTEM

For the technique to be practical, the monitoring overhead must not affect the usability of the web application. We analyze the costs of running AMNESIA monitoring in terms of space and time. The space requirement of monitoring is dominated by the size of the generated SQL query models. In the worst case, the size of the query models is quadratic in the size of the application. This case corresponds to the unlikely situation of a programme that branches and modifies the query string for each programme statement. In typical programs, the generated automata have a linear programme size. In fact, it has been our experience that most slots are actually quite small for the size of the corresponding application. The time complexity of the approach depends on the cost of running the query tokens against the models. Since we are checking a set of tokens against NDFA, the complexity of the comparison is worst-case exponential in the number of tokens in the query (in the worst case, all states are visited for each token). However, in practice, SQL query models typically reduce to trees, and the cost of matching is nearly linear with query size. Our experience shows that the cost of the running access phase is negligible. In terms of imitations, our technique can generate false positives and false negatives.

Although the string analysis we use is conservative, false positives may be generated in situations where the string analysis is not accurate enough. For example, if the parser cannot determine that a hard-coded string in the application is a keyword, it may assume that it is an input-related value and incorrectly represent it as a /3 in the SQL query model. At runtime, the original keyword would not match a placeholder for a variable, and AMNESIA would mark the corresponding query as SQL. False negatives can occur when a constructed SQL query model contains dummy queries and an attacker is able to generate an injection attack that matches one of the dummy queries.

In this work, an attempt was made to improve the effectiveness of the techniques above by employing a combination approach to protecting web applications from SQL Injection attacks.

7.3 FUTURE SCOPE OF THE PROJECT

This project has a comprehensive scope for the future. The project can be implemented and updated on the software in the future when the need arises because it is flexible in terms of expansion. If there are any changes that need to be made or improvements to the software to make it more appealing to the audience, we will make them. We've tried to ensure security from the start by using fencing to prevent attackers from accessing the database from the same location. system for the next 48 hours. Even if an attacker gets past the initial phase, we have transformed the way of querying into a semantic query process using an encrypted query parser where SQL injection can be prevented at the edge level. In the future, our team will focus on enhancing our schema and queries using fuzzy sets. This project has an enormous scope for the future.

The project can be implemented and updated on the software in the future when the need arises because it is supple in terms of expansion. If any changes need to be made or the software improved to make it more appealing to the audience, they will be made.

In this future improvement of the project, it is not overshadowed that big data significantly affects IT companies, and thanks to the development of new technologies, we are the only ones who deal with them managerial. Big data is completely changing the way organizations, governments, and academic institutions operate as they use different tools to manage big data. Hadoop and NoSQL databases will be in high demand in the future.

As part of future work, we want to evaluate methods using different web application scripts in the public domain to achieve high accuracy in SQL injection prevention approaches. SQL integration with Nikto HTTP Scanner, HTTP Scanning Proxy, and Metasploit will help uncover additional web vulnerabilities. also add functionality to dump vulnerable databases and database schema. The proposed work can be expanded to detect a greater number of attacks and improve performance. The system can be modified to detect and prevent XML and web service attacks can be extended to detect and prevent more attacks.

REFERENCES

1. Sonakshi, Rakesh Kumar, and Girdhar Gopal Preventing SQL Injection Attacks Using RC4 and Blowfish Encryption Techniques Preventing SQL Injection Attacks Using RC4 and Blowfish Encryption Techniques (ijert.org)

2. Srinivas Aired, Varalakshmi Perumal, Narayan Gowran, Ram Srivatsa Kannan pdf (Srinivas Reddy. github.io) An Application-Specific Randomized Encryption Algorithm to Prevent Accidental SQL4 Injection

3. W.H. Next, Mohan Rondônia, and Viraj Samaranayake. SQL Injection Attacks Identification and Mitigation Tool (SQLA) | IEEE Conference Publication | IEEE Xplore

4. Karan Ray, Nitish Pol, and Suraj Singh Data leak detection through SQL injection prevention in e-commerce Data Leak Detection through SQL Injection Prevention in E-Commerce (ijser.org)

5. Ramarao, Ingrain Lasundra SQL injection attacks on web applications should be avoided.Emerging (ijres.org)

6. chin ping, wang jinshan, yang lauan, pan lin.SQL Injection Training Using SQL Labs https://ieeexplore.ieee.org/document/9236904

7. Yash Swarup, Anuj Kumar, Abhishek Tyagi, and Vimal Kumar Using query hashing to prevent SQL injection attackshttps://ieeexplore.ieee.org/document/9581804

8. Anshuman Jana, Priya Bordello, depends on friendship. AN ANALYSIS APPROACH BASED ON INPUT THAT PREVENTS SQL INJECTION ATTACKS https://ieeexplore.ieee.org/document/9236904

9. Vinod Kannan. Sal injection revolution and prevention of database attacks, published by academia.edu. il_work_card=view-paper