**Data Breach Avoidance System**

**Abstract:**

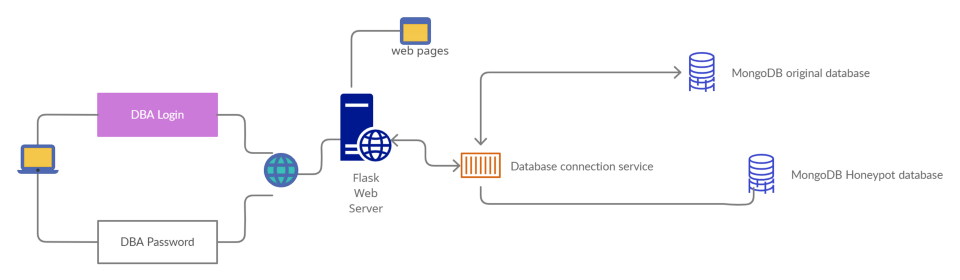
A data breach is an event where information is stolen or attained from a system without the knowledge or authorization of the system’s owner. This can be a serious concern for organizations that involve losing sensitive, proprietary, or confidential information such as credit card numbers, customer data, or trade secrets. With the right and smart application of Computer science aptitude, Data breaches cannot only be avoided but also helps understand attackers’ moves. One such strategy is Honeypot. This strategy involves maintaining a sacrificial database that’s intended to attract cyberattacks. It mimics a target for hackers and uses their intrusion attempts to gain information about cyber criminals and the way they are operating or to distract them from other targets. This system is also powered by detecting change using SHA256 hashing on piece-wise data.

**CHAPTER 1**

**INTRODUCTION**

**1.1 Introduction**

In Organizations, when network security is at stake, the next big thing the organization or cloud service provider aims at is Information Security. The expression “data security” signifies shielding data and data frameworks from unapproved access, use, divulgence, interruption, alteration, or annihilation to give honesty, classification, and availability. Here, honesty implies preparing for ill-advised data change or destruction and incorporates guaranteeing data non-disavowal and realness. Secrecy implies protecting approved limitations on divulgence and access, counting implies guarding restrictive and individual protection data. Also, accessibility implies guaranteeing convenient and dependable admittance to and utilization of data. Distributed computing gives registering and capacity assets on request without the requirement for the inward foundation which guarantees cost-saving advantages. As the innovation course of action turns out to be more famous, extra distributed computing safety efforts are important to guarantee the proceeded with insurance of the privacy, accessibility, and trustworthiness of big business information. The actual limits of information and moving that information between confided-in accomplices safely and dependably is changed by distributed computing. To guarantee the most recent security abilities are being utilized appropriately, this capacity of distributed computing will require encryption and trust models to be continually assessed. This capacity might be upgraded by involving the right specialist organization in the cloud. To guarantee data security information stockpiling and protection security should be considered. Cloud-based capacity is by and large broadly utilized as a feasible answer for the issue of information stockpiling in settings where monetary and viable contemplations disallow the utilization of privately based equipment and programming assets. Client reservations and legitimate imperatives anyway have led to inquiries regarding the evidence of the trustworthiness of the put-away information, particularly on account of the public cloud foundation. Another issue has thus emerged, that of examining put-away records to acquire Proof of Retrievability. Secure distributed storage frameworks are restricted by the overheads they expect to give the necessary security levels. Joined utilization of cloud and neighborhood computational assets is important to empower positive client encounters. With expanding neighborhood handling limits, the main pertinence is experienced in the Big Data Processing worldview. The volumes of information that should be handled are overpowering so much that approaches which utilize limitless measures of force, for handling and capacity are not attainable. This paper centers around a new investigation of hash work prerequisites for enormous information applications and a related key-based hash work plan procedure that makes the continuous assortment, rundown, examination, and decision-making in light of streaming information. A record-examining strategy is recommended that involves fundamental big data mass processing operations to foster an effective and reliable verification of recoverability calculation.

System Design

**1.2 Motivation**

The main motivation of the project is to address data breach problems and come up with a strategy to avoid in case of there is a breach in web security.

**1.3 Objectives**

The main objective is to implement a strategy, called the Honeypot strategy, which involves maintaining a sacrificial database that’s intended to attract cyberattacks. It mimics a target for hackers and uses their intrusion attempts to gain information about cyber criminals and the way they are operating or to distract them from other targets.

**1.4 Proposed Solution**

In a Cloud environment, when web security is breached or fails, the next big thing that organizations aim at is, securing information stored in the servers and avoiding data breaches. In the proposed system, we aim at deploying a Honeypot server that triggers when intrusion is detected by the system. Once triggered, a smart session analysis is done on the attacker's move, to know even more about the attacker.

**1.5 Contribution of the Project**

>Smart session analysis on Attacker’s moves.

>Main data is not exposed to intruders.

>Alerts cloud providers about incidents to make the system even more resistant.

**CHAPTER 2**

**Software Requirement Specification**

**2.1 Introduction**

**Software Requirement Specification (SRS) Format** as the name suggests, is a complete specification and description of requirements of the software that needs to be fulfilled for the successful development of a software system. These requirements can be functional as well as non-functional depending upon the type of requirement.

**2.2 Literature Survey**

**Honeypot-based intrusion detection system: A performance analysis, Published in 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)**

Attacks on the internet keep on increasing and it causes harm to our security system. In order to minimize this threat, it is necessary to have a security system that has the ability to detect zero-day attacks and block them. “Honeypot is the proactive defense technology, in which resources are placed in a network with the aim to observe and capture new attacks”. This paper proposed a honeypot-based model for intrusion detection systems (IDS) to obtain the best user data about the attacker. The ability and the limitations of Honeypots were tested and aspects of it that need to be improved were identified. In the future, they aim to use this trend for early prevention so that pre-emptive action is taken before any unexpected harm to our security system.

**A Study on Advancement in Honeypot based Network Security Model, Published in 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)**

Throughout the years, honeypots have been very useful in tracking down attackers and preventing different types of cyber-attacks on a very large scale. More companies should rely on and adopt this system because of intrusion detection features and low false-positive rates. But with time, the attackers tend to start discovering loopholes in the system. Hence it is very important to be up to date with the technology when it comes to protecting a computing device from emerging cyber-attacks. Timely advancements in the security model provided by the honeypots help in more efficient use of the resource and also lead to better innovations in that field. This paper reviewed different methods of honeypot network and also gives an insight into the problems that those techniques can face along with their solution. Further, it also gives detail about the most preferred solution among all of the listed techniques in the paper.

**Intrusion Detection and Prevention using Honeypot Network for Cloud Security, Published in 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)**

With the rapid increase in the number of users, there is a rise in issues related to hardware failure, web hosting, space, and memory allocation of data, which directly or indirectly lead to the loss of data. With the objective of providing services that are reliable, fast, and low in cost, we turn to cloud-computing practices. With tremendous development in this technology, there is an ever-increasing chance of its security being compromised by malicious users. A way to divert malicious traffic away from systems is by using Honeypot. It is a colossal strategy that has shown signs of improvement in the security of systems. Keeping in mind the various legal issues one may face while deploying Honeypot on third-party cloud vendor servers, the concept of Honeypot is implemented in a file-sharing application that is deployed on a cloud server. This paper discusses the detection attacks in a cloud-based environment as well as the use of Honeypot for its security, thereby proposing a new technique to do the same.

**2.3 Functional requirements**

The main functional requirements of the system are:

>Session Analysis

A session is a sequence of actions performed by an intruder (in the network) in the application during a limited period of time. Analyzing the session is one of the major requirements for this application. A smart session analysis helps understand the attacker’s moves.

>Information Security of Bank Cards

Not exposing main and important data to intruders is one of the high-priority functional requirements as data like user bank cards are sensitive.

>Promt Notifications and Alerts

The System should be able to Alert cloud providers/ system developers about incidents to make the system even more resistant.

>Storage of users’ bank cards

The application should be able to store user bank cards that include all the related information of those cards such as name, card number, and expiry date.

>Administrative functions

The system should provide Administrative functions to administrators like DBA (Database Administrator) to maintain and look after important aspects of the application.

>Business and Government Rules

The system is to be developed based on business rules and government rules, for instance, a bank card application should not store a CVV value without user permission.

>Addition, modification, and deletion of existing bank Cards

The system should provide an interface to manage existing added cards in the application.

>Authentication

The users need to be authenticated based on 1-factor (or 2-factor) authentication.

>Authorization levels

The system should provide various authorization levels or different kinds of users to have increased security and reliability. For instance, an end-user is not authorized to access the internal database.

>Audit Tracking

A system audit is a periodic examination an organization does to assess and document the performance of its systems. The goal of such an audit is to see how well a system is doing in meeting a set of established controls and best practices.

**2.4 Interface requirements**

The following are the interface requirements for the My Bank Cards Manager application:

* Attractive
* Simple to use
* Responsive in a short time
* Clear to understand
* Consistent on all interface screens

Maintain consistency across a family of applications: The development of some set of applications should follow and implement the same design, and rules so that consistency is maintained among applications.

* Flexible interaction
* Interruptable and undoable

**2.5 Performance Requirements**

>Less memory consumption to load and run the application.

This can be achieved by optimal performance design and implementation.

>Load screens faster

**2.6 Nonfunctional atributes**

>Better individual control of data

>Increased reliability

>High guarantee the authenticity

>Powerful Auditing

>Extended Access Control

**CHAPTER 3**

**PROJECT DESIGN**

**3.1 Introduction**

Today most organizations tend to focus much more on Web security than Information security, but considering attackers' capabilities, it is essential to have a secure strategy to safeguard Information. The existing systems include using robust access rights manager, systems powered by cryptography, etc.

Proposed System:

In a Cloud environment, when web security is breached or fails, the next big thing that organizations aim at is, securing Information stored in the servers and avoiding data breaches. In the proposed system, we aim at deploying a Honeypot server that triggers when the system detects intrusion. Once started, a smart session analysis is done on the Attacker's move to know even more about the Attacker.

Advantages of the Proposed System:

>Smart session analysis on Attacker's moves.

>Main data is not exposed to intruders.

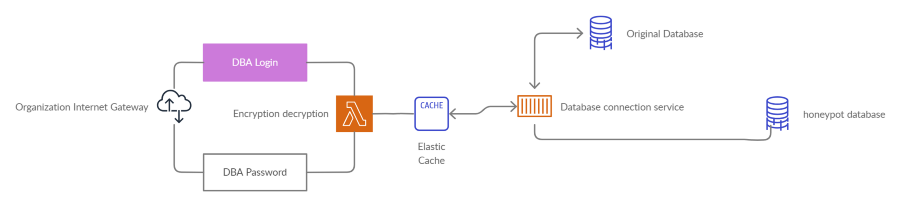
>Alerts cloud providers about incidents to make the system even more resistant.

Research Challenges or Future Improvements:

To optimize for cost, it is not recommended to keep these decoy systems scaled up all the time; they should spawn only when an attacker crosses a certain level of control measures.

**3.2 System Design and Architecture**

Architecture

Architecture

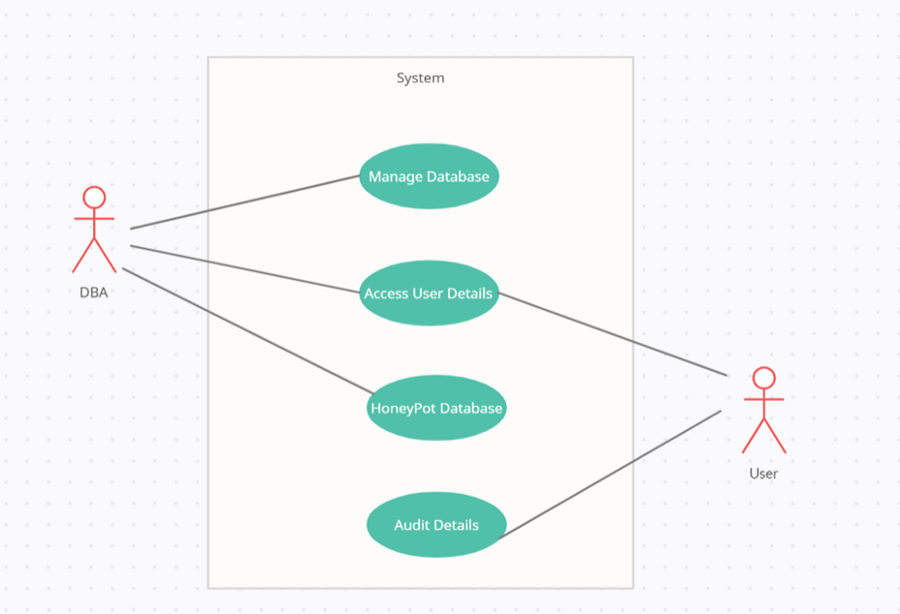
System Design

Diagram

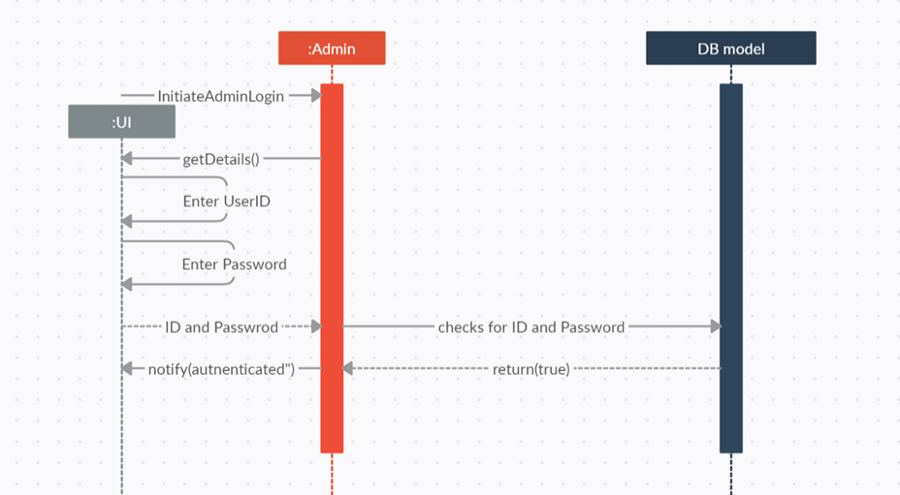
Description automatically generatedSystem Design

### UML Diagrams:

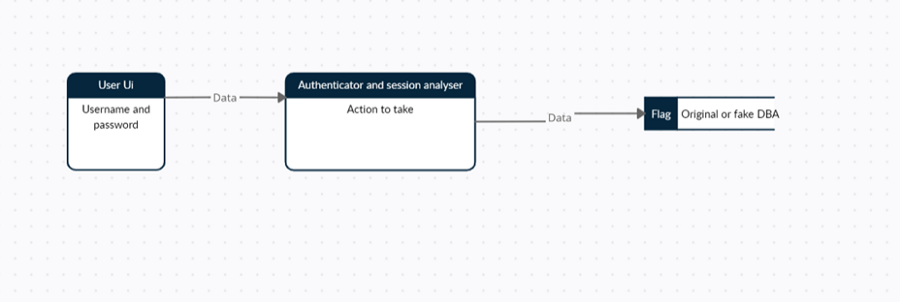
>Use Case Diagram



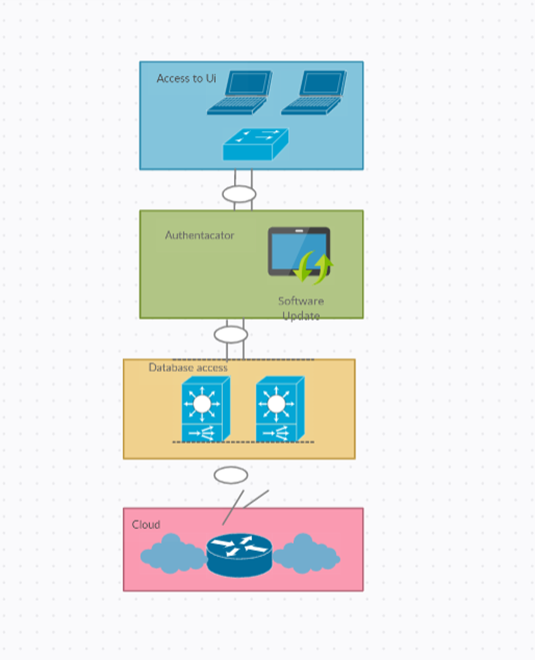
>Sequence Diagram



>Dataflow Diagram

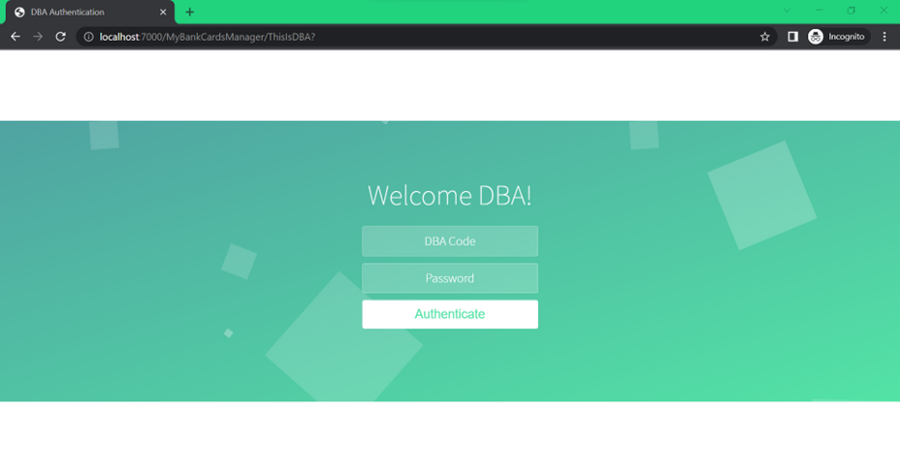


>Deployment Diagram

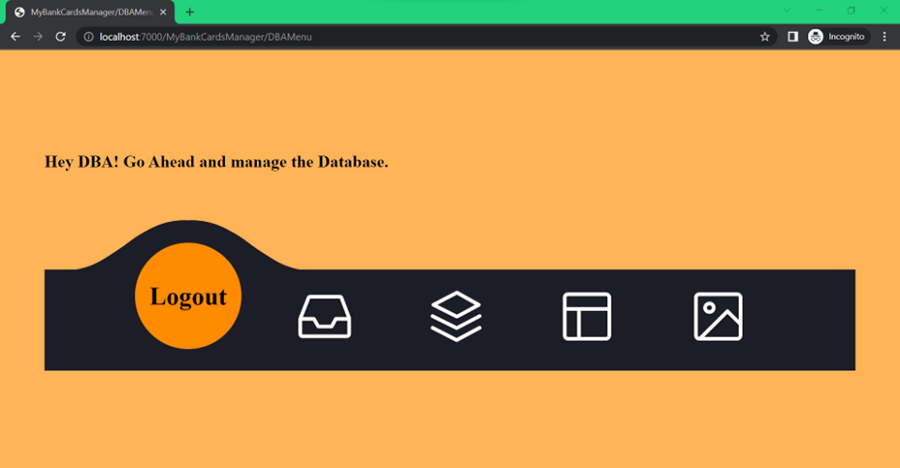


**3.3 GUI Design**

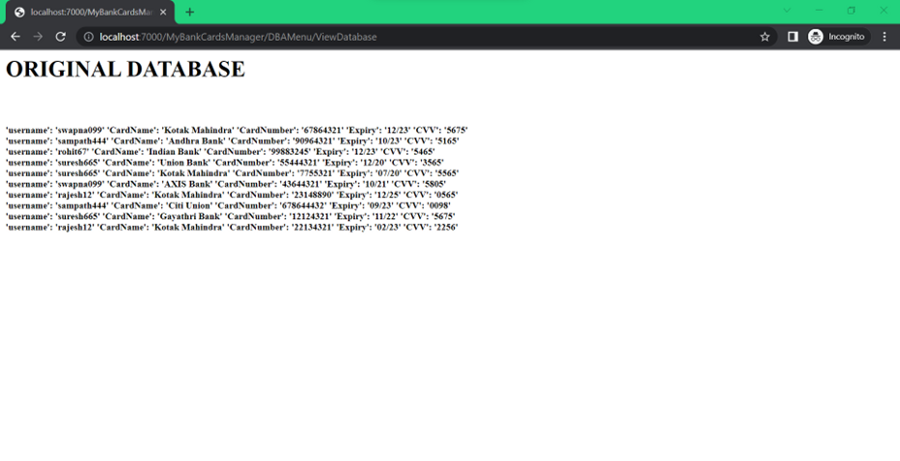
>DBA Authentication Page



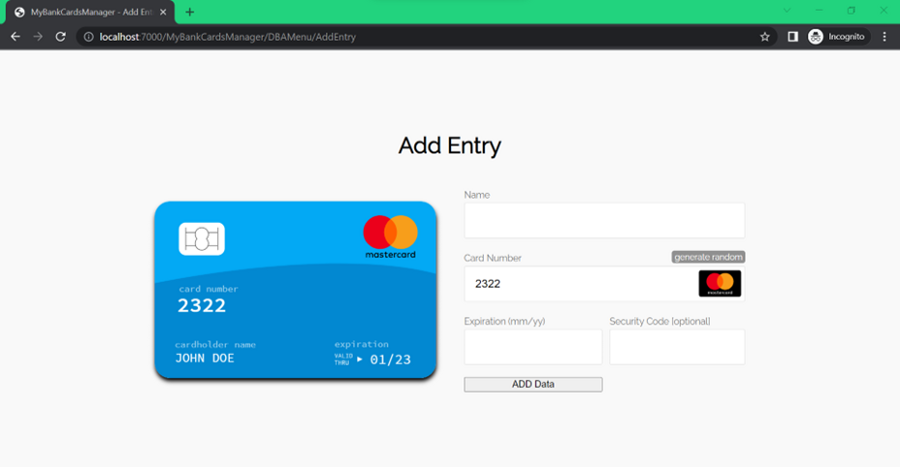
>When DBA Login is Successful



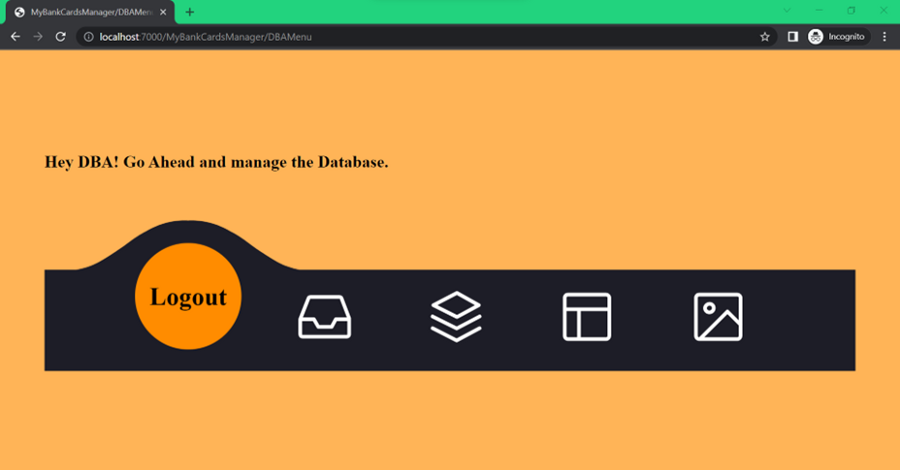
>Viewing Database



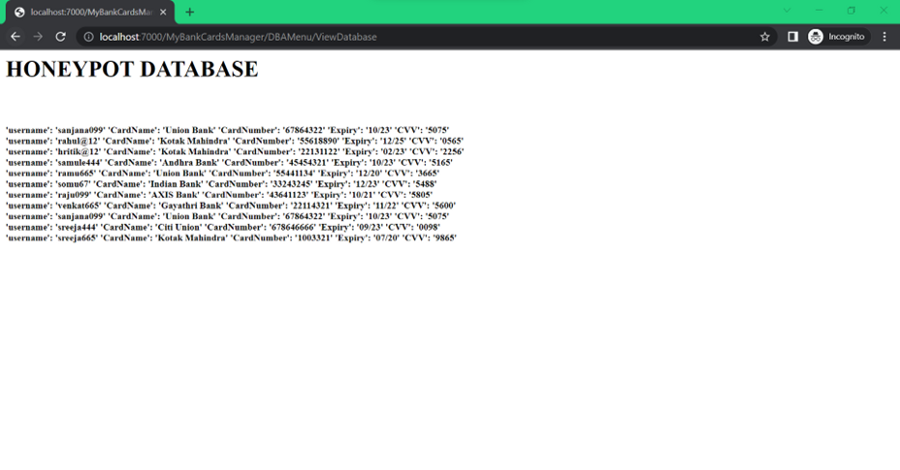
>Adding Entry into Database



>When Login fails 3 times (Honeypot screen)



>Viewing database in Honeypot Session



**Technology Stack**

The Data breach avoidance System consists of the following modules and parts:

* MongoDB Application with MongoDB Server running at the desired Port Number.
* Python Driver Code
* Web pages along with Script Sheets (.js) and Style Sheets (.css).
* Database (JSON Data of Users)

**Chapter 4**

**IMPLEMENTATION**

**My Bank Cards Manager**

My Bank Cards Manager is a simple web application that stores and manages user bank cards. It supports a few functionalities such as adding a new card, View Stored Cards, and Drop Cards. The user data is securely stored in a database.

**System Design and Technologies Used:**

Front End: **Mark-Up Sheet + Script Sheet + Style Sheet**

· Mark-Up Sheet: **HTML**

· Script Sheet: **JS**

· Style Sheet: **CSS**

Back End:

· Running HTTP Server, and for Server-side Scripting: **Python Flask**

· Database: **MongoDB** (Driver: PyMongo)

### Modules:

The Data breach avoidance System consists of the following modules and parts:

* MongoDB Application with MongoDB Server running at the desired Port Number.
* Python Driver Code
* Web pages along with Script Sheets (.js) and Style Sheets (.css).
* Database (JSON Data of Users)

The first Module, MongoDB Software, will be downloaded from the official MongoDB site and installed at the desired location. Next, a Mongo DB server will be running in the background at port number 27017. This server will be receiving the Queries from Module 2, which is Python driver Code. Once the query is received, the server will execute it and sends the result back to that driver code.

In Second Module, The Python program will be using two essential libraries, Flask and PyMongo. Python program will use Flask for all web application creation and maintenance tasks.

Flask Server:

Without a web server, In a traditional static web application, we write the “on click” function for the button in the web page to redirect any other desired web page (rendering a web page without doing any computation).

But with a web server and an API like python flask, we want a specific python function to be executed whenever we click on an action-listener button. So in the “on click” field of any button, we now mention the function to be executed instead of mentioning the name of the web page to be rendered.

Now In the python function, we do perform some computations like taking input from text fields, analyzing some logic/flags, and then, with some intelligence, we render an appropriate web page.

The Flask server will be running at the desired port number (7000 in our application).

The Third Module consists of all the desired web pages along with Style Sheets i.e., CSS files, and Script Sheets i.e., JavaScript Files needed for the application.

The Fourth module is the database collection of User data, and Also HoneyPot Data. These files are encrypted and securely stored.

**System testing** is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design but also the behavior and even the developer's expectations. It is also intended to test up to and beyond the bounds defined in the software requirements specification(s).

The following examples are different types of testing that are done during System testing:

**>**GUI test has been done on the system to ensure that the all-desired animations and input buttons, text fields, etc. are working as per requirement.

>System testing has been done on the data breach avoidance system to ensure that the entire integrated software system meets our requirements, and to ensure known and predictable results. An example of system testing here is, testing if the process flow is working properly like having visibility of the database only after the login page appears, etc.

>White Box Testing has been performed with the knowledge of internal workings, structures, and purposes to test a few areas that cannot be reached from a black-box level.

>Performed Black Box Testing in which the software under test is treated, as a black box. One cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

>Unit Testing has been done on all the individual units and modules that are part of the system, for example, testing of data loader, data migrator, driver application, etc.

>Finally, the functional testing is performed to ensure that all the Functional requirements such as being able to implement a Smart session analysis on Attacker’s moves, Not to expose main and important data to intruders and Alerts cloud providers about incidents to make the system even resistant.

**Verification and Validation**

Verification is the process of checking that software achieves its purpose with no bugs. It is the procedure to ensure whether or not the product this is developed is right or not. It verifies whether or not the developed product fulfills the necessities that we have. Verification is static testing. The system has been verified at major stages to eliminate logical errors and bugs.

Validation is the process of checking whether the software product is up to the mark or in other words product has high-level requirements. It is the process of checking the validity of the product i.e., it checks what we are developing is the right product. it is a validation of actual and expected products. Validation is dynamic testing.

The system has been validated for the functional requirements. Fig 4 and Fig 7 in the 4.2 section ensure that the main purpose of the system is achieved.

**Chapter 5**

### ****Deployment of the Application:****

The following are the Steps the deploy the application:

**Step 1**: Download and install Python (v3+) and MongoDB.

>Note the MongoDB installation path, which would be C:\Program Files\MongoDB

**Step 2**: Run mongod.exe from MongoDB Folder

>Path is C:\Program Files\MongoDB\Server\4.4\bin\mongod.exe

>Note: If this is crashing, probably due to exit code 100.

To resolve this, create a folder path: C:\data\db

**Step 3:** Run MyBankCardsManager\Load MongoDB Data\LoadData.py script

**Step 4:** Run MyBankCardsManager\drivercode.py

**Step 5:** Check accessibility by reaching<http://localhost:7000/MyBankCardsManager>

### ****Running the Application:****

Following are the Steps to Run the Application once the Deployment is done.

**Step 1:**

Run the MongoDB server first (File name: mongod.exe)

path: C:\Program Files\MongoDB\Server\4.4\bin\mongod.exe

**Step 2:**

Now run the python driver code

(Present in MyBankCardsManager/drivercode.py)

**Step 3:**

Enter the following URL in Chrome:

<http://localhost:7000/MyBankCardsManager>

### ****Results and Outputs:****

>Viewing Database

Text

Description automatically generated

>Viewing database in Honeypot Session

Text

Description automatically generated

**Chapter 6**

**Conclusion and Future Scope**

**Advancements that can be made to the project:**

>Use an SSH connection to access the data instead of MongoDB. (looks more close to reality)

>To optimize for cost, it is not recommended to keep these decoy systems scaled up all the time, they should spawn only when an attacker crosses a certain level of control measures.

**REFERENCES**

[1] Honeypot-based intrusion detection system: A performance analysis, Published in 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)

[2] A Study on Advancement in Honeypot-based Network Security Model, Published in 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)

[3] Intrusion Detection and Prevention using Honeypot Network for Cloud Security, Published in 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)

[4] [online] <https://www.developer-tech.com/news/2020/sep/02/why-you-need-honeypots-in-the-cloud-a-guide/>