

# Faculty of Information Technology

## IOT SECURITY DASHBOARD

By:

No.	Student Name	Student ID
1	Andrew Ashraf Agyaby	20-00380
2	Ali Mohamed Abohesiba	20-01148
3	Ahmed Mohamed Fathy	20-01225
4	Micheal Maximums	20-00040
5	Yass Sameh Hana	20-00076
6	Micheal Zaher Ramses	20-01278
7	Beemen Nader Gad	20-00363

Under Supervision of:

**Dr. Basem Mohamed**

-lecturer in Computer and  
Information Technology  
Egyptian E-Learning University.

▪ **Eng. Dina Mohamed**

- Teaching assistant in  
Computer and  
Information Faculty at  
EELU

This thesis is submitted as a partial fulfillment of the requirements for the  
degree of Bachelor of Science in Computer & Information Technology.

**EELU – Asyut 2024**

## ACKNOWLEDGEMENTA

First and for most, praises and thanks to the God, the Almighty, for his blessing throughout our years in the college and our graduation project to complete this stage of our life successfully. We have made efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them. Thanks to Egyptian E-Learning University especially Assiut center for helping us to reach this level of awareness. We would like to express our deep and sincere gratitude to our project supervisor Dr. Basem Mohamed for giving us the opportunity to lead us and providing invaluable guidance throughout this project. It was a great privilege and honor to work and study under his guidance. We are extremely grateful for what he has offered us. We are especially indebted to say many thanks to Eng. Dina Mohamed for guidance and constant supervision as well as for his patience, friendship, empathy, and great sense of humor. His dynamism, vision, sincerity and motivation have deeply inspired us. Finally, we would like to express our gratitude to everyone who helped us during the graduation project.

## Abstract:

This study aims to design and develop a security dashboard for the Internet of Things (IoT) to provide a centralized interface for managing and monitoring device and data security in an IoT environment. IoT security is crucial due to the advancement of technology and the proliferation of connected devices.

The dashboard provides a comprehensive display of relevant security information, such as current attacks, device security status, and early warnings of potential threats. Data is collected from multiple sources in the IoT environment, including sensors, connected devices, and event logs, and visually presented on the dashboard.

Smart analysis techniques and machine learning are employed using the collected data to provide accurate security assessments and recommendations for improvement. Information is represented visually and in an easily readable manner through charts, graphs, and other visual formats on the dashboard.

The dashboard enables quick interaction with the displayed data, allowing users to take immediate actions to address security threats and enhance the security of the IoT environment. An intuitive and user-friendly interface is provided for interacting with the dashboard and executing necessary actions.

The use of this IoT security dashboard enhances security awareness and contributes to overall security in the IoT environment. It serves as a powerful tool for security administrators and operators in the IoT field to address the growing security challenges and maintain the integrity of devices and data associated with the IoT infrastructure.

## Table of Contents

<b>ACKNOWLEDGEMENTA.....</b>	<b>2</b>
<b>Abstract: .....</b>	<b>3</b>
Chapter 1 .....	8
Introduction.....	8
Introduction: .....	9
The Role of an IoT Security Dashboard .....	10
Problem statement: .....	12
Problem solution: .....	13
A Deep Dive into the SDLC .....	15
Chapter 2.....	21
Dashboard .....	21
What is the meaning of dashboard? .....	22
What are the features of Dashboard? .....	22
Why we use Dashboard? .....	23
Dashboards: A Comprehensive Overview .....	25
What are Dashboards Made of? .....	25
What are the Uses of Dashboards?.....	26
What are the Benefits of Using Dashboards? .....	26

How to Create a Dashboard .....	27
Chapter 3 .....	29
Literature review .....	29
Chapter 4 .....	33
Background .....	33
Python overview: .....	34
Python Syntax compared to other programming languages .....	36
What is the Frontend? .....	36
Firebase overview: .....	38
Main responsibilities: .....	39
Client Software (Front-End) .....	39
What is JavaScript? .....	40
What is Bootstrap? .....	41
What is Responsive Web Design? .....	41
What is the jQuery? .....	42
What is Firebase? .....	43
Firebase Features: .....	47
Authentication: .....	47
Backend overview: .....	48

How to become a back-end developer?.....	48
Back-end developer technical skills.....	48
Chapter 5.....	52
Chapter 6.....	60
Software Requirements.....	60
ERD .....	61
USE CASE .....	63
SEQUENCE DIAGRAM.....	65
Chapter 7.....	67
Implementation .....	67
and coding.....	67
Analyzing the system: .....	68
Entity Relationship Diagram (ERD): .....	68
Here are some key components of an ERD: .....	69
Dataset cleaning .....	71
Name of attacks .....	72
Name of devices in dataset.....	73
Register in website .....	74
login in website .....	75

Firebase.....	76
Chapter 7 .....	77
Conclusion .....	77
summary: .....	78
Conclusions: .....	79
Future work: .....	80
References .....	81
References .....	82



# Chapter 1

## Introduction



## Introduction:

The Internet of Things (IoT) is a term that refers to the internet-connected network composed of everyday devices and objects such as home appliances, sensors, cars, wearable devices, buildings, and more, which can communicate and exchange data.

The idea behind IoT is to connect physical objects to the internet and enable them to communicate and exchange data, allowing for smarter interactions and connectivity between devices and their surrounding environment. This is achieved using sensors and sensing devices that gather environmental data and transmit it over the network for processing and analysis.

IoT applications are diverse and include areas such as home automation, smart agriculture, smart industry, smart healthcare, smart transportation, smart cities, and many others. The goal of IoT is to improve efficiency, simplify processes, and enhance quality of life by providing accurate information and effective analytics based on the collected data from internet-connected things.

The term "dashboard" refers to a user interface that displays a set of information and data in a visual and organized manner. The dashboard aims to provide a quick and centralized overview of important data, performance, and other relevant information in a visual and simplified format.

The dashboard consists of various elements such as charts, graphs, tables, numbers, indicators, and other visual indicators that present information in an easily readable and understandable way. Dashboards can be customized for various purposes in fields such as business, analytics, project

management, marketing, healthcare, education, and more.

The dashboard is a powerful tool for data management and strategic decision-making, as it can aggregate different data sources and present quantitative data in a meaningful and analyzable form. Additionally, dashboards may include additional features such as interactivity, data filtering, alerts, customization of display, and reporting capabilities.

## The Role of an IoT Security Dashboard

This is where an IoT security dashboard comes in. It's a user interface specifically designed to display information and data related to the security posture of your IoT devices.

### What Does the Dashboard Display?

- **Device Status and Health:** Monitor device connectivity status and firmware updates to ensure they are operating correctly and securely.
- **Security Threats and Vulnerabilities:** Track potential security threats such as unauthorized access attempts and malware, allowing you to take preventive measures.
- **Suspicious Activity Alerts:** Receive immediate notifications if any suspicious activity is detected on your IoT network, enabling you to respond promptly.

- **Security Posture Trends:** Observe overall security health trends of your IoT devices over time, helping you identify areas that need improvement.

### **Benefits of an IoT Security Dashboard**

- **Comprehensive Visibility:** The dashboard provides a centralized view of all your IoT devices, making it easier to monitor the security of the entire network.
- **Faster Threat Response:** Real-time alerts allow you to detect and address threats quickly, minimizing the impact of potential attacks.
- **Informed Decision-Making:** The data and trends displayed on the dashboard help you identify weaknesses in your security system and focus your efforts on improving them.
- **Enhanced Compliance:** The dashboard can assist you in meeting regulatory requirements related to data security and privacy.

## Problem statement:

The increasing adoption of the Internet of Things (IoT) technology has led to a significant rise in the number of connected devices and the generation of vast amounts of data. However, along with these advancements comes the pressing issue of ensuring the security of IoT ecosystems. One critical aspect of IoT security is the need for an effective security dashboard that can provide real-time monitoring, analysis, and management of security threats and vulnerabilities.

Currently, there is a lack of comprehensive and centralized security dashboards specifically designed for IoT environments. Existing security solutions often focus on individual devices or specific aspects of security, making it challenging to gain a holistic view of the overall security posture of an IoT ecosystem. This fragmented approach hampers the ability of security administrators and operators to detect and respond to emerging threats promptly.

Moreover, the distributed and heterogeneous nature of IoT networks poses unique challenges to the design and implementation of an effective security dashboard. IoT systems encompass a wide range of devices with varying capabilities, communication protocols, and data formats, making it difficult to integrate and analyze security-related information from different sources. Additionally, the dynamic nature of IoT environments demands a dashboard that can adapt and scale to accommodate the evolving landscape of devices and threats.

Therefore, there is a critical need for an advanced IoT security dashboard that can address these challenges and provide a unified view of the security status of an IoT ecosystem. Such a dashboard should be capable of collecting, aggregating, and analyzing security-related data from diverse sources, enabling proactive threat detection, timely incident

response, and effective security management. By addressing these issues, the development of an IoT security dashboard will contribute to enhancing the overall security and trustworthiness of IoT deploy

## **Problem solution:**

To address the challenges associated with IoT security dashboards, the proposed solution is to design and develop an advanced IoT security dashboard that provides a centralized interface for managing and monitoring device and data security in IoT environments. The key features and components of the solution include:

**Comprehensive Security Monitoring:** The IoT security dashboard will collect data from various sources within the IoT ecosystem, such as sensors, connected devices, and event logs. It will provide real-time monitoring of security-related information, including current attacks, device security status, and potential threats.

**Smart Analysis and Machine Learning:** The collected data will be analyzed using smart analysis techniques and machine learning algorithms. This analysis will enable accurate security assessments, anomaly detection, and identification of potential vulnerabilities. The dashboard will provide recommendations for improving security based on the analysis results.

**Visual Representation:** The security information will be visually presented on the dashboard using charts, graphs, and other visual formats. This visual representation will make it easy for security administrators and operators to understand and interpret the security status of the IoT ecosystem briefly.

**Interactive Interface:** The dashboard will offer a user-friendly and intuitive interface that allows users to interact with the displayed data. Users will be able to take immediate actions to address security threats, such as initiating security protocols, updating device configurations, or implementing security patches.

**Scalability and Adaptability:** The IoT security dashboard will be designed to accommodate the dynamic nature of IoT environments. It will be scalable to handle large number of connected devices and capable of adapting to evolving device types, communication protocols, and security standards.

By implementing this solution, the IoT security dashboard will provide a holistic.

## A Deep Dive into the SDLC

The Internet of Things (IoT) revolution is transforming our world, but with increased connectivity comes heightened security risks. To effectively manage and secure your growing network of IoT devices, a robust IoT dashboard security system is essential. But how do you build one? Here's a comprehensive look at the software development life cycle (SDLC) for creating a secure IoT dashboard:

### 1. Requirements Gathering: Defining the Security Fortress

This is the foundation upon which your entire system is built. Here, you meticulously gather and document the specific security needs of your IoT environment. This includes:

- **Understanding Threats:** Identify potential security vulnerabilities in your IoT ecosystem. What are the most likely attack vectors?
- **Desired Features:** Outline the functionalities you want in your dashboard. Do you need real-time threat detection? Anomaly analysis? Device risk assessments?
- **Integration Requirements:** Ensure seamless integration of the dashboard with your existing IoT devices and network infrastructure.

### 2. Design: Blueprinting Security and User Experience

Now you translate the gathered requirements into a blueprint. This phase involves:

- **Architectural Design:** Define the system's technical backbone, considering scalability, performance, and security considerations.
- **User Interface (UI) Design:** Craft an intuitive and user-friendly interface that allows for easy access to security data and insights.
- **Machine Learning (ML) Integration:** Design how ML algorithms will be incorporated to enhance security. This could include threat detection, anomaly identification, and device behavior analysis.

### 3. Development: Bringing the Blueprint to Life

This is where the coding magic happens. Here, developers translate the design into a functional system:

- **Backend Infrastructure Development:** Build the core functionalities of the dashboard, including data storage, processing capabilities, and communication protocols.
- **UI Component Development:** Create a user interface that users will interact with to access security information and manage their IoT environment.
- **Machine Learning Integration:** Integrate ML libraries or frameworks to enable advanced security analytics and threat detection capabilities.



## 4. Testing: Ensuring Impregnability

Before unleashing your security champion onto the world, rigorous testing is crucial:

- **Unit Testing:** Test individual software components to ensure they function correctly.
- **Integration Testing:** Verify that all components work seamlessly together within the system.
- **Security Testing:** Conduct penetration testing and vulnerability assessments to identify and address any security weaknesses.
- **Model Testing:** Validate your ML models using appropriate datasets to ensure they can accurately detect and respond to security threats.

## 5. Deployment: Securing Your IoT Landscape

Once the system passes rigorous testing, it's time to deploy it in the real world:

- **Production Environment Setup:** Configure the necessary infrastructure, network connections, and ensure seamless integration with your IoT devices and networks.

- **User Training:** Provide user training on effectively utilizing the dashboard's functionalities and interpreting security insights.

## 6. Monitoring and Maintenance: Eternal Vigilance

Security is an ongoing battle. Here's how to ensure your dashboard remains a guardian against threats:

- **Continuous Monitoring:** Monitor the system for security threats, suspicious activity, and potential vulnerabilities.
- **Machine Learning Model Updates:** Regularly update your ML models with new data to improve their accuracy and effectiveness in detecting evolving threats.
- **Security Patch Management:** Promptly apply security patches and updates to address any identified vulnerabilities in the software or underlying infrastructure.
- **Incident Response:** Have a well-defined incident response plan in place to address security breaches and minimize potential damage.

## Security Best Practices: Building an Unbreachable Wall

Throughout the SDLC, prioritize security best practices. This includes:

- **Secure Coding Practices:** Employ secure coding techniques to mitigate vulnerabilities introduced during development.
- **Data Encryption:** Encrypt sensitive data at rest and in transit to safeguard against unauthorized access.

**Access Control Mechanisms:** Implement robust access controls to limit access to sensitive data and functionalities based on user privileges.

By following these steps and prioritizing security best practices, you can build an IoT dashboard security system that empowers you to manage and secure your connected environment effectively. Remember, in the ever-evolving world of IoT security, vigilance and continuous improvement are key to staying ahead of potential threats.

## Time Plan



# Chapter 2

## Dashboard

## **What is the meaning of dashboard?**

A collection of visual information in a centralized and organized way. The task of a dashboard is to provide a comprehensive and quick overview of a particular status or performance of a set of key data or indicators.

Dashboards are used in various fields and industries, such as business management, analytics, business management, operations partners, etc. Dashboards are designed partly visually for users, and rely on the use of scores, scores and other visual indicators to illustrate trading data.

It can include metrics dashboards and key performance indicators (KPIs) to see statistics, timelines, strategic analyzes and future predictions. The use of active billboards provides comprehensive visibility, quick analysis and important policy trends, helping companies, managers and teams make the most comprehensible decisions, based on important goals.

## **What are the features of Dashboard?**

**Tailored to needs:** The dashboard can be customized to meet the needs of the individual user or organization. You can choose which indicators and metrics you want to see, analyze, and organize them the way you prefer.

**Comprehensive analysis:** Dashboards display data and information from multiple sources in one place. Different data sources can include internal databases and external sources such as social analytics, web data, financial data, and more. This data can be compiled and analyzed holistically to provide detailed insight.

**Quick extraction of information:** The dashboard simplifies complex data

into easy-to-understand, quick-to-digest information. With visual design and proper organization, users can quickly and efficiently extract patterns, trends, and potential problems.

**Real-time analysis:** Dashboards enable real-time data display, allowing users to track events and changes instantly. Current performance can be monitored and compared with historical performance to identify significant trends and changes.

**Interactive Analysis:** Users can interact with the dashboard and analyze data in different ways. You can filter data, change the visual display, perform detailed analyses, and explore the data more deeply. This interaction allows new insights to be achieved and invisible patterns to be discovered in the data.

## **Why we use Dashboard?**

**Centralized data view:** Dashboards bring together data from various sources into a single, unified view. This eliminates the need to access multiple systems or databases to gather information, saving time and effort.

**Real-time data updates:** Dashboards can be configured to display real-time data, providing up-to-the-minute information. This is particularly valuable in dynamic environments where timely decision-making is crucial.

**Customization and personalization:** Dashboards can be customized to suit individual preferences and requirements. Users can choose the specific metrics, visualizations, and layout that best align with their needs, allowing for a personalized and tailored experience.

**Mobile accessibility:** Many modern dashboards are designed to be mobile-

responsive or have dedicated mobile applications. This enables users to access and monitor data on the go, facilitating remote work and decision-making.

**Alerts and notifications:** Dashboards can be equipped with alerting mechanisms that notify users of predefined thresholds or significant changes in data. This ensures that important events or anomalies are promptly brought to attention.

**Data drill-down and filtering:** Dashboards often provide interactive features that allow users to drill down into specific data points or apply filters to focus on

subsets of data. This flexibility supports detailed analysis and exploration.

**Data storytelling:** Dashboards can be used to tell a compelling data-driven story. By arranging visualizations and data elements in a logical sequence, dashboards can guide users through a narrative, highlighting key insights and supporting decision-making.

**Collaboration and sharing:** Dashboards can be shared with team members or stakeholders, promoting collaboration and transparency. Users can provide access to relevant parties, enabling them to view and interact with the dashboard, facilitating discussions and alignment.

**Historical data analysis:** Dashboards often include historical data, allowing users to compare and analyze trends over time. This historical perspective enables the identification of long-term patterns and the evaluation of performance against past benchmarks.



Scalability and integration: Dashboards can handle large volumes of data and can integrate with various data sources, including databases, APIs, and external systems. This scalability and integration capability make dashboards adaptable to different organizational needs and data environments.

## Dashboards: A Comprehensive Overview

In an increasingly complex world, where data flows from countless sources, the need for effective tools to understand and analyze this data has become more critical than ever. This is where **dashboards** play a crucial role.

### What are Dashboards Made of?

Dashboards typically consist of a combination of visual components, such as:

- **Charts:** Used to present quantitative data in an easily understandable format.
- **Tables:** Used to display organized data in a structured manner.
- **Gauges:** Used to quickly indicate key performance indicators (KPIs).
- **Maps:** Used to visualize geospatial data.
- **Colors and Images:** Used to make information more engaging and visually appealing.

## What are the Uses of Dashboards?

Dashboards are employed in a wide range of fields, including:

- **Business:** Dashboards are used to monitor financial performance, track KPIs, analyze customer data, and much more.
- **Analytics:** Dashboards are used to display the results of statistical analyses, uncover patterns in data, and identify emerging trends.
- **Project Management:** Dashboards are used to monitor project progress, identify risks, track tasks, and allocate resources.
- **Marketing:** Dashboards are used to track marketing campaigns, analyze customer behavior, and measure ROI.
- **Healthcare:** Dashboards are used to monitor patient health, track treatment outcomes, and improve quality of care.
- **Education:** Dashboards are used to track student performance, analyze assessment data, and identify areas for improvement.
- **Government:** Dashboards are used to monitor public services, analyze social and economic data, and make informed policy decisions.

## What are the Benefits of Using Dashboards?

Dashboards offer numerous benefits, including:

- **Improved Decision-Making:** Dashboards provide a comprehensive overview of data, enabling users to understand trends, patterns, and make informed decisions more quickly.

**Enhanced Productivity:** Dashboards help users focus on the m information, saving time and increasing productivity.

- **Promoted Communication:** Dashboards facilitate information sharing among stakeholders, fostering collaboration and teamwork.
- **Unveiling New Insights:** Dashboards can assist in uncovering new insights in data that may not be apparent through traditional methods.
- **Optimized Operations:** Dashboards can be used to identify areas for improvement in business processes, leading to increased efficiency and effectiveness.

## How to Create a Dashboard

Creating an effective dashboard involves the following steps:

**Define the Goal:** What is the purpose of the dashboard? What information do you want to display?

**Gather Data:** Collect data from relevant sources.

**Clean Data:** Clean the data and ensure its accuracy and consistency.

**Choose Visual Components:** Select appropriate visual components to represent the data.

**Design the Dashboard:** Design the dashboard to be easy to understand and visually appealing.

**Test the Dashboard:** Test the dashboard with users to gather feedback.

**Regularly Update the Dashboard:** Regularly update the dashboard with fresh data.

## Summery

Dashboards are powerful tools that can be used to understand data and make informed decisions. By designing dashboards effectively, organizations can reap numerous benefits, including improved decision-making, enhanced productivity, promoted communication, unveiled new insights, and optimized operations. Dashboards are essential tools for navigating the data-driven world of today.



# Chapter 3

## Literature review

Leonardo Baban *et.al* (2019) With the increasing popularity of IoT devices, several different companies are developing new platforms to manage and control the interaction between IoT and users. IoT apps are mainly used to capture and process the sensor data and to automate and execute tasks. In general, these IoT platforms provide the means for IoT devices to communicate with each other and to provide real-time analysis for users. Due to the wide variety of applications in IoT, these platforms vary in their target audience

Hamdan Hejazi *et.al* (2018) The idea of the Internet of Things is emerging rapidly on finding out their route to our modern life, aiming to enhance the finesses of life by linking many smart devices, technologies, and applications together. This paper has provided an overview of IoT architectures and their features, and the recent research addressing different aspects of the IoT. We must emphasize that in contrast to critical IoT applications we focused on mass IoT applications. Therefore, the IoT platforms we surveyed mostly reflected the massive IoT requirements.

B. Di Martino *et.al* (2020) Several Reference Architectures have been proposed to systematize Internet of Things environments. Some of these architectures have been standardized by international committees, while others have been developed by academic researchers and, despite their validity and general applicability, they still have not been standardized yet.

In this paper, such architectures have been analyzed and compared, trying to cover standard, commercial and academic proposals. According to the reference architectures, a

1st Yohanes Yohanan Fridolin Panhuman *et.al* (2021) The conducted Surveys and analyzes to compare several IoT platforms using such parameters or criteria such as Thing management, Connectivity, Data Storage, Data Abstraction, Interface, Analytical, Feedback &

Collaboration, Security, Scalability, Microservices, Plug and Play. The results of comparing the IoT platforms show that some of the IoT platforms have fulfilled these criteria like KAA platform.

Tibor Hinkler et.al (2022) The investigation covers many aspects such as device management, integration, security, protocols for data collection, types of analytics, support for visualizations. This, in turn, could expand the foundation of understanding the architecture and the role of different components and protocols that are framing the IoT. The description of our novel architecture represented in Section II might be useful.

Shaiful Ahdan et.al (2022) Based on the results of system testing at 90.3% usability, 85.8% functionality and 90% Reliability, the Efficiency aspect test obtained the highest level of CPU efficiency at the level of 28% and dropped stably at a percentage of 10%, for the highest memory usage of 119.2MB begins with a memory usage of 47.7MB out of a total allocation of 143.1MB. The conclusion of this study is that the IoT-based Result =  $\frac{\text{Actual Score}}{\text{Ideal Score}} \times 100\%$  smart energy Dashboard proposal can help in controlling electrical devices in real time.

J. Lorand El et.al (2016) In this paper, we have proposed a fast power and performance estimation approach for FPGA-based systems. Based on a user-defined scenario, an efficient comparison among several configurations of wireless communication systems can be realized. Note that the proposed methodology is not limited to wireless communication systems but can be applied to many applications in various domains. The methodology consists of three independent steps that are, first, an IP characterization phase and, second, the definition of a scenario and third, a system-level simulation. During the first step, hardware IP blocks or VHDL modules.

Megha Gupta et.al (2019) The ongoing research in the field of IoT

and its implementation in full or partial manner will improve the quality of life. Thus, the proposed project “IoT Based Advanced Vehicle System” would take the security level a step forward and try to cover many of the loopholes which are in existing technology. The verification shows that the IOT based advanced vehicle System is realistic and can control theft automatically. The response time delay is also less. This IOT based advance vehicle system enables user safety by seat belt compulsion, key less locking /unlocking system to operate the car. In addition to the above, it gives security from towing of car and theft through the car window. The system is ideal for cars, further it can be used for other vehicles too by using these components and modules used in this project. IOT based advance vehicle system offers utmost efficiency, convenience, safety & reliability. It is an ideal solution for car users.



# Chapter 4

## Background

## Python overview:

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It was created by Guido van Rossum and first released in 1991. Python has gained immense popularity among developers, data scientists, and researchers due to its ease of use and extensive libraries and frameworks.

Here is an overview of Python's key features and characteristics:

**Readability:** Python emphasizes code readability with its clean and straightforward syntax. It uses indentation and whitespace to define code blocks, making it easy to understand and maintain.

**Easy to Learn:** Python is considered one of the most beginner-friendly programming languages. Its simplicity and readability make it accessible to newcomers. Additionally, Python has a large community that provides ample learning resources and support.

**Versatility:** Python is a general-purpose programming language, which means it can be used for a wide range of applications. It supports various programming paradigms, including procedural, object-oriented, and functional programming styles.

**Vast Ecosystem:** Python boasts a rich ecosystem of libraries and frameworks that enable developers to accomplish diverse tasks efficiently. Some notable libraries include NumPy for numerical computing, Pandas for data manipulation, Matplotlib for data visualization, and TensorFlow and PyTorch for machine learning and deep learning.

**Cross-Platform Compatibility:** Python is available on major operating systems like Windows, macOS, and Linux, making it highly portable.

Python code written on one platform can typically run on other platforms without modifications.

**Interpretation:** Python is an interpreted language, meaning that code is executed line by line rather than being compiled into machine code before execution

This allows for faster development cycles and easier debugging

**Dynamically Typed:** Python is dynamically typed, which means variable types are inferred during runtime. Developers do not need to explicitly declare variable types, making the language flexible and allowing for rapid prototyping.

**Large Community and Support:** Python has a vast and active community of developers worldwide. This community contributes to the development of the language, creates libraries and frameworks, and provides support through online forums, tutorials, and documentation.

**Integration and Extensibility:** Python can be easily integrated with other languages and systems. It supports interfaces to languages like C/C++, allowing developers to leverage existing libraries and infrastructure. Python also has extensive support for web development, database connectivity, and network programming.

**Data Science and Machine Learning:** Python has become a popular choice for data analysis, scientific computing, and machine learning. Its libraries, such as NumPy, Pandas, and scikit-learn, provide powerful tools for data manipulation, analysis, and modeling.

The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.

In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, PyCharm, NetBeans or Eclipse which are particularly useful when managing larger collections of Python files.

### Python Syntax compared to other programming languages

Python was designed for readability and has some similarities to the English language with influence from mathematics.

Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

Python relies on indentation, using whitespace, to define scope, such as the scope of loops, functions and classes. Other programming languages often use curly brackets for this purpose.

## What is the Frontend?

The front-end of a web- or mobile application is the part the user interacts with directly. It is usually referred to as the application's "client side." The front end consists of everything that the user sees when interacting with the website or app, such as text colors and styles, photos, graphs and tables, buttons, colors, the navigation menu, and much more. Frontend developers provide the structure, appearance, behavior, and content of everything that appears on browser displays when websites.

The key focus points of front-end development are responsiveness and performance. A front-end developer must make sure that the site is responsive, meaning that it works properly on devices of all sizes. The application's performance should always be stable, no matter what device is used to access the application. The objective of designing a site is to ensure that when the users open the site, they see the information in a format that is easy to read and relevant. This is further complicated by the fact that users now use a large variety of devices with varying screen sizes and resolutions thus forcing the designer to take into consideration these aspects when designing the site.

They need to ensure that their site comes up correctly in different browsers (cross-browser), different operating systems (cross platform) and different devices (cross-device), which requires careful planning on the side of the developer. And when using Front End applications on smart phones, we must know the following Native mobile app developer: i.e. build an app in native languages like Kotlin and swift and these apps run on only one platform in Android or IOS. I'm getting more and more switching from java to Kotlin.

Also, Kotlin is somewhat like java in syntax. If you want to build apps for IOS, google swift language learning. Hybrid mobile apps developer: i.e. build hybrid apps and hybrid apps are the apps work on both android and iOS platforms. Do you want to build applications and hybrids using the web and other technologies? Each of these libraries/frameworks is excellent.

## Firestore overview:

The technique we used in Backend IS Firestore: -

Firestore is a Cloud-hosted, NoSQL database that uses a document-model. It can be horizontally scaled while letting you store and synchronize data in real-time among users. This is great for applications that are used across multiple devices such as mobile applications. Firestore is optimized for offline use with strong user-based security that allows for serverless based apps as well.

Firestore is built on the Google infrastructure and is built to scale automatically. In addition to standard NoSQL database functionality, Firestore includes analytics, authentication, performance monitoring, messaging, crash reporting and much more. Because it is a Google product, there is also integration into a lot of other products. This includes integration with Google Ads, Adom, Google Marketing Platform, the Play Store, Data Studio, Big Query, Slack, Jira, and more.

The Firestore APIs are packaged into a single SDK that can be expanded to multiple platforms and languages. This includes C++ and Unity, which are both popular for mobile development.

A Front-End Developer is someone who creates websites and web applications.

The difference between Front-End and Back-End is that Front-End refers to how a web page looks, while back-end refers to how it works.

You can think of Front-End as client-side and Back-End as server-side.

The basic languages for Front-End Development are HTML, CSS, and JavaScript.

Main responsibilities:

The main responsibility of the Front-End Developer is the User interface.

Simply put, create things that the user sees.

Tip: If you are curious about how to become a front-end developer, you can read our [How To Become a Front-End Developer Tutorial](#).

Client Software (Front-End)

The basic languages of Front-End Development are:

- HTML
- CSS
- JavaScript

Popular JavaScript and CSS frameworks and libraries:

- Bootstrap
- HTML DOM
- JSON
- jQuery
- Angular
- React

## What is JavaScript?

JavaScript, often abbreviated as JS, is a programming language that alongside HTML and CSS, is considered one of the core technologies of the web. It's what makes webpages interactive and dynamic.

Here are some key things to know about JavaScript:

**Makes webpages dynamic:** Unlike static HTML pages, JavaScript allows webpages to update and change their content, respond to user actions, and create animations and other interactive features.

**Runs on client-side:** JavaScript code is typically embedded within HTML pages and executed by the web browser's built-in JavaScript engine. This means the code runs on the user's computer rather than the web server.

**Versatile:** JavaScript can be used for a variety of tasks on a webpage including:

- Updating HTML content

- Changing HTML element attributes

- Manipulating data

- Validating user input

- Creating animations and other interactive features

**Popular:** Nearly all websites (around 99%) use JavaScript on the client-side for webpage behavior.

If you're interested in learning more about JavaScript, there are many resources available online, including tutorials and documentation.

**Beyond web pages:** While JavaScript is synonymous with web development, its applications have grown significantly. Today, JavaScript can be used for server-side scripting with frameworks like Node.js, enabling the creation of real-time applications and scalable web servers.



It's also used in mobile app development through frameworks like React Native, allowing you to build mobile apps with JavaScript code. This versatility makes JavaScript a powerful and in-demand skill for modern programmers.

---

## What is Bootstrap?

Bootstrap is a free front-end framework for faster and easier web development

Bootstrap includes HTML and CSS based design templates for typography, forms,

buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins

Bootstrap also gives you the ability to easily create responsive designs

Advantages of Bootstrap:

Easy to use: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap

Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops

Mobile-first approach: In Bootstrap 3, mobile-first styles are part of the core framework

Browser compatibility: Bootstrap is compatible with all modern browsers (Chrome, Firefox, Internet Explorer, Edge, Safari, and Opera)

---

## What is Responsive Web Design?

Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops

## What is the jQuery?

jQuery is a popular JavaScript library. In simpler terms, it's a collection of pre-written JavaScript code that simplifies common web development tasks.

Here's why jQuery is widely used:

**Makes JavaScript easier:** jQuery provides a simpler way to write JavaScript code. It offers functions (pre-written code blocks) that handle complex tasks with fewer lines of code compared to vanilla JavaScript (JavaScript without libraries).

**Cross-browser compatibility:** Different web browsers can interpret JavaScript slightly differently. jQuery helps ensure your code works consistently across different browsers.

**Common tasks simplified:** jQuery simplifies many common web development tasks such as:

**DOM manipulation:** This refers to changing the structure and content of your HTML page using JavaScript. jQuery makes it easier to target and modify HTML elements.

**Event handling:** This involves responding to user interactions like clicks, scrolls, or key presses. jQuery simplifies attaching event listeners to elements.

**Animations:** jQuery provides functions to create various animations and visual effects on your webpage.

**AJAX:** This stands for Asynchronous JavaScript and XML. It allows webpages to communicate with servers in the background without requiring a full page reload. jQuery simplifies making AJAX requests.

While jQuery is still widely used, it's important to note that other JavaScript frameworks have emerged in recent years. However, understanding jQuery can be a good foundation for learning other JavaScript libraries and frameworks.

## What is Firebase?

Firebase is a comprehensive app development platform developed by Google. It provides a suite of tools and services that help developers build, launch, and grow their mobile and web applications. Here are some key features and capabilities of Firebase

**Real-time Database:** Firebase offers a NoSQL cloud-hosted database that allows you to store and sync data in real-time. Data is synchronized across all connected clients in milliseconds, and it works offline as well

**Authentication:** Firebase Authentication provides several authentication methods, including email/password, phone, Google, Facebook, Twitter, and GitHub, making it easy to incorporate secure sign-in into your app

**Hosting:** Firebase Hosting is a fast and secure way to host your web app, static content, and even dynamic content served by a Node.js server

**Cloud Functions:** Firebase Cloud Functions allow you to run server-side code in response to events or HTTP requests without having to manage your own server infrastructure.

**Cloud Storage:** Firebase Cloud Storage provides a simple, powerful way

to store and serve user-generated content, such as photos or videos.

**Notifications:** Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that securely delivers messages at no cost.

**Crashlytics:** Firebase Crashlytics is a powerful, lightweight crash reporting solution that helps you track, prioritize, and fix stability issues

**Analytics:** Firebase Analytics is a comprehensive app measurement solution that provides deep insights into your app usage and user behavior

**ML Kit:** Firebase ML Kit provides a set of pre-trained machine learning models that you can use to add intelligent features to your app, such as text recognition, face detection, and more.

Firebase is a popular choice for developers building mobile and web applications, as it provides a comprehensive set of tools and services that simplify the development, deployment, and management of their applications.

## Real-time Database:

The Firebase Real-time Database is a NoSQL cloud-hosted database that allows your app to store and sync data in real-time.

Data is synchronized across all connected clients in milliseconds and remains available even when your app goes offline.

## Authentication:

Firebase Authentication makes it easy to integrate sign-in and authentication flows into your app.

It supports a wide range of authentication providers, including email/password, phone, Google, Facebook, Twitter, GitHub, and more. You can customize the authentication UI to match your app's branding and experience.

Firebase handles all the complex security and infrastructure required for authentication, leaving you to focus on the user experience.

## Hosting:

Firebase Hosting is a fast and secure way to host your web content, including single-page apps, static sites, and dynamic apps served by Cloud Functions.

It uses a global content delivery network (CDN) to ensure fast loading times for your users.

Hosting integrates seamlessly with other Firebase products, making it easy to deploy updates and manage your entire app infrastructure.

### Cloud Functions:

Firebase Cloud Functions allow you to run server-side code in response to events or HTTP requests, without having to manage your own server infrastructure.

You can use Cloud Functions to implement custom logic, trigger background tasks, and integrate with other cloud services.

Functions are written in Node.js and can access other Firebase services, such as the Real-time Database and Authentication.

### Storage:

Firebase Cloud Storage provides a simple, powerful way to store and serve user-generated content, such as photos and videos

It's built on Google Cloud Storage, providing a highly scalable and secure solution for your app's file storage needs.

Cloud Storage integrates seamlessly with other Firebase services, making it easy to build end-to-end solutions.

Firebase is a mobile and web application development platform that provides developers with a plethora of tools to build high-quality apps.

Google in 2014 and has since grown to become one of the most popular backend-as-a-service (BaaS) providers. Firebase's purpose is to make app development easier, faster, and more efficient. With Firebase, developers can focus on building great user experiences without having to worry about server infrastructure or complex backend logic.

## **Firebase Features:**

Firebase offers a wide range of features designed to help developers build better apps. Some of its key features include real-time database, authentication, cloud messaging, hosting, and analytics. For example, the real-time database allows developers to store and sync data in real-time across multiple clients and platforms. This makes it easy to build collaborative apps such as chat apps or collaborative editing tools.

**Real-time Database** The real-time database is one of Firebase's most powerful features. It allows developers to store and sync data in real-time across multiple clients and platforms. This means that any changes made to the data are immediately reflected on all connected devices. This is particularly useful for building collaborative apps such as chat apps or collaborative editing tools. It also eliminates the need for complex server-side code, making it easier and faster to build apps.

## **Authentication:**

Authentication is a critical part of any app that requires user data. Firebase makes it easy to implement authentication using a variety of methods such as email and password, Google sign-in, and Facebook login. By using Firebase's authentication feature, developers can ensure that only

authorized users have access to sensitive data or functionality within their app. This helps to improve the security of the app

## **Backend overview:**

What is the Backend?

Back-end Development refers to the server-side development. It focuses on databases, scripting, and website architecture. It contains behind-the-scenes activities that occur when performing any action on a website. It can be an account login or making a purchase from an online store. Code written by back-end developers helps browsers to communicate with database information.

## **How to become a back-end developer?**

There are many paths you can take to become a web developer. Whether you are a recent graduate or hoping to switch careers, it is important to assess what transferable skills you already have and consider building the new skills needed to pursue a back-end developer role.

## **Back-end developer technical skills**

As a back-end developer, there are certain technical skills you will need to learn to navigate developing the back end of the web or mobile application.

**Programming languages:** Any back-end developer needs to be well-versed in back-end programming languages such as Python, Java, and PHP. These make the website function when used alongside databases,



frameworks, and servers. Python is one of the most popular programming languages because it is compatible with artificial intelligence (AI) and machine learning and works well for writing clear and logical code. Basic knowledge of front-end languages HTML, CSS, and JavaScript is a bonus.

**Frameworks:** Frameworks are the libraries of back-end programming languages that help to build the server configuration. They tend to be linked with programming languages, so if you are familiar with Python, you'll also know Flask, Django, or another Python-based framework, and so on.

**Databases and servers:** You'll need to understand how to stack and recover data from databases, as back-end programming controls access to this information, including storage and recovery. MongoDB and MySQL are popular database programs. The database stores and organizes the easily arranged and recovered, just like you client's data so that it can be might use cloud storage for your photos. This database then runs on a server that provides data upon request.

**Application Program Interface (API):** An API is a series of definitions and rules for developing application software. In addition to internet browser Want a mobile app for iOS or Android. websites, companies often Knowledge of application-building languages like JavaScript will expand your job opportunities.

**Accessibility and security clearance:** You should develop knowledge of network protocols and web security. Knowing how to secure databases .and servers will be critical to your success as a back-end developer If you are interested in the cloud, consider enrolling in IBM's Full-Stack

Cloud Developer professional certificate for the full gamut of cloud-specific technologies. Guided by IBM experts, you will learn how to build cloud-based applications, understand front-end languages like HTML and ,CSS, back-end languages and frameworks like Express, Node.js, Python

and Django, and much more. These are foundational tools whether you decide to apply for cloud-related jobs.

Reasons for using the backend:

Let's take a closer look at the reasons you might need the backend for mobile apps. First and foremost, the reason for having a backend is the fact that you won't be able to communicate with the user without having it. It's an essential part if you must send something to the user and vice versa. If you need a professional mobile app that will serve you in the long run, you should completely forget about excluding a backend from your plans. Indeed, if you're looking for success, otherwise, you might leave things as they are. Backend Types and Their Use Cases: The implementation of the backend into your mobile app can be quickly done with multiple options. Typically, they can be classified by various aspects like technology, architecture, and the way they interrelate with the front-end applications. However, if you intend to create your own application, the efforts spent on this may be the basis for the classification as well. One of the most efficient.

And the easiest option is to create a custom backend from scratch. The following

Opportunity gives you incredible flexibility in terms of building your own project with an unlimited number of implemented features. Hiring a team of professionals can be a solution if you want professional assistance to get the best result possible. However, it's no wonder that the suggested solutions cost significant money and time resources. Fortunately, there are much cheaper options.

Like Back-end developer workplace skills Alongside technical skills, these workplace skills will enable you to work more efficiently, effectively, and seamlessly with team members.

Communication: A back-end web developer needs to thoroughly understand the engineer's vision to execute it. Strong written and conversation skills will help you communicate any ideas and troubleshoot with team members and stakeholders.

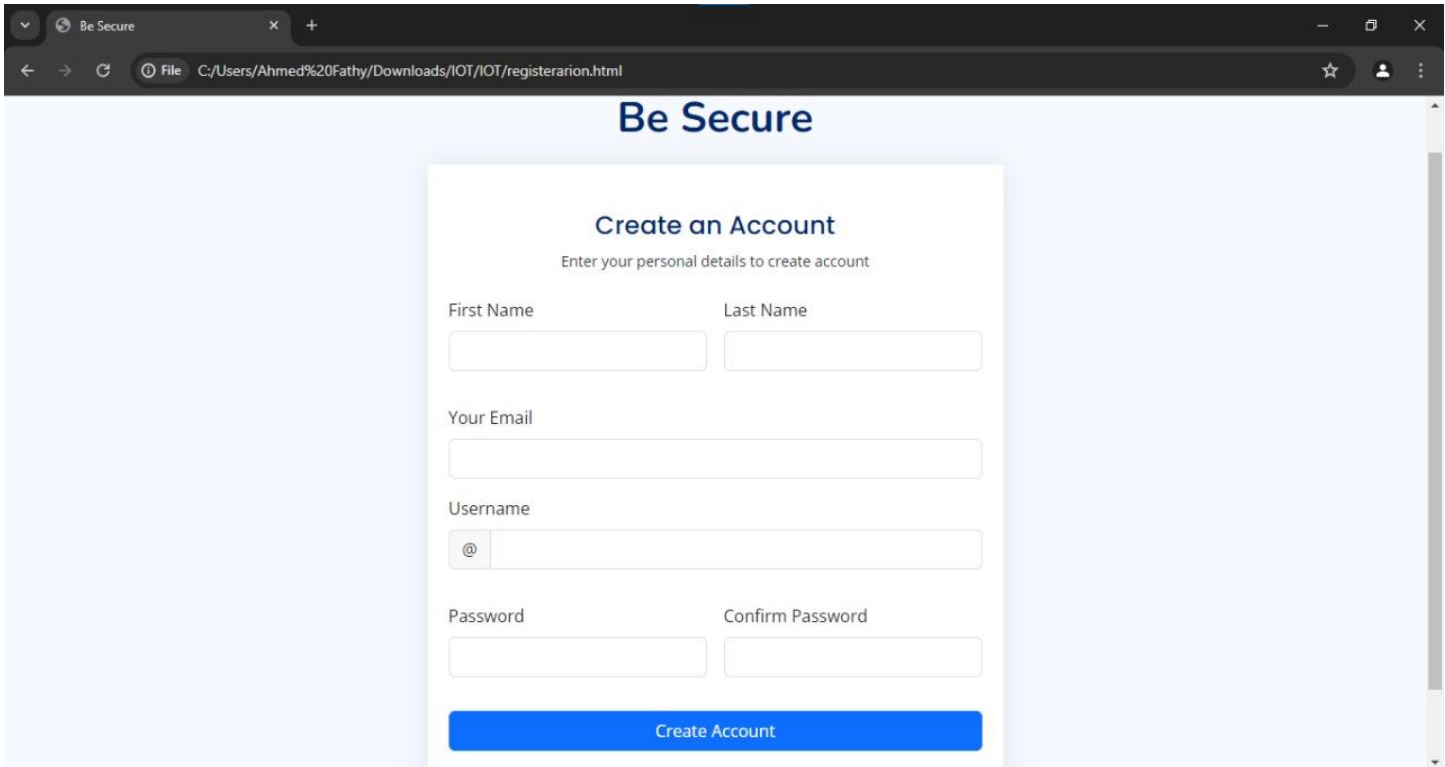
Problem-solving and analytical thinking: You will need to find creative solutions when developing a web or mobile app, such as debugging code and revising it without crashing the entire site. As a developer, you should be able to analyze why a portion of code does or does not work and anticipate and prevent errors.

Industry knowledge: Holistic understanding of the tech industry is always helpful to keep up with overall economic trends as well as updates to languages and platforms. To brush up, investigate blogs, forums, news, and books related to web and app development.



# Chapter 5

# Dashboard Design



**Be Secure**

### Create an Account

Enter your personal details to create account

First Name

Last Name

Your Email

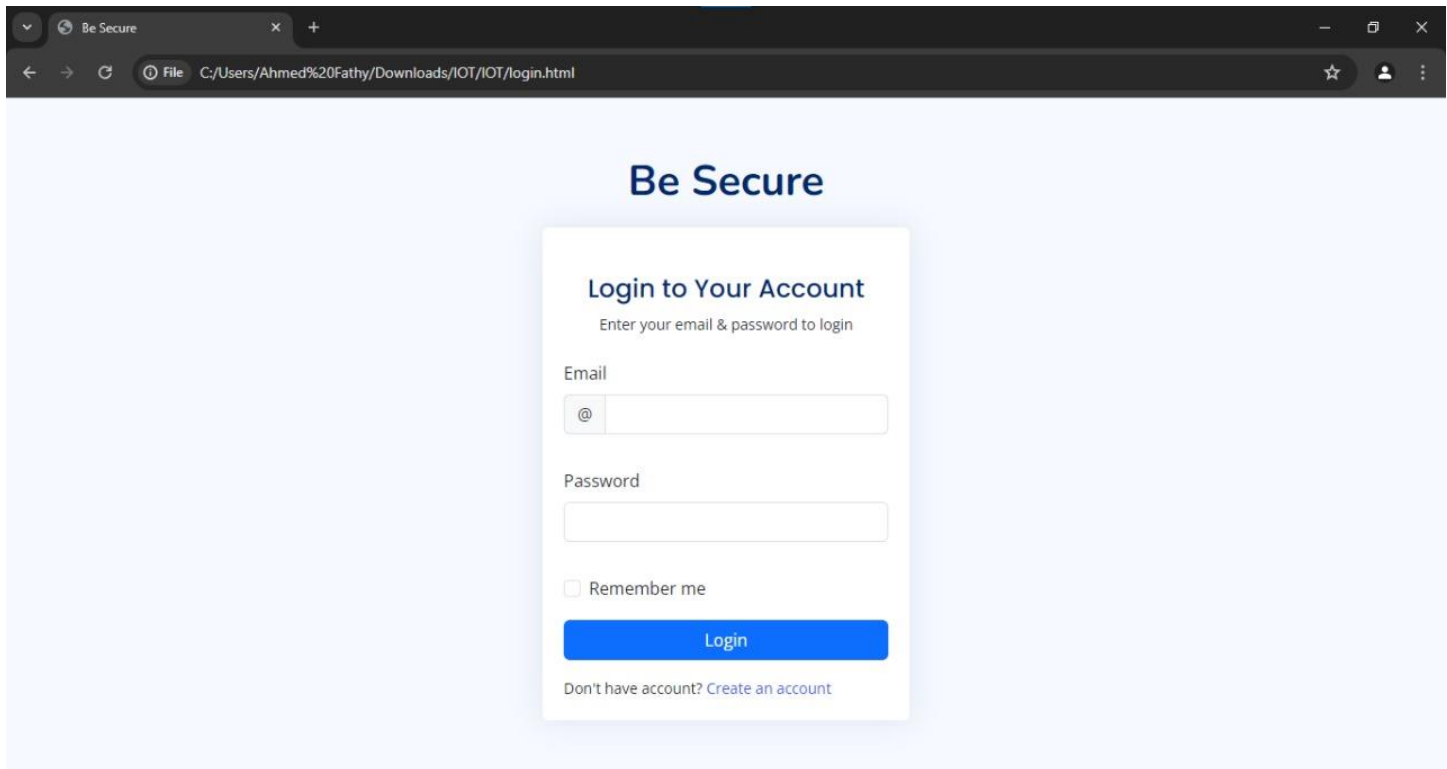
Username

Password

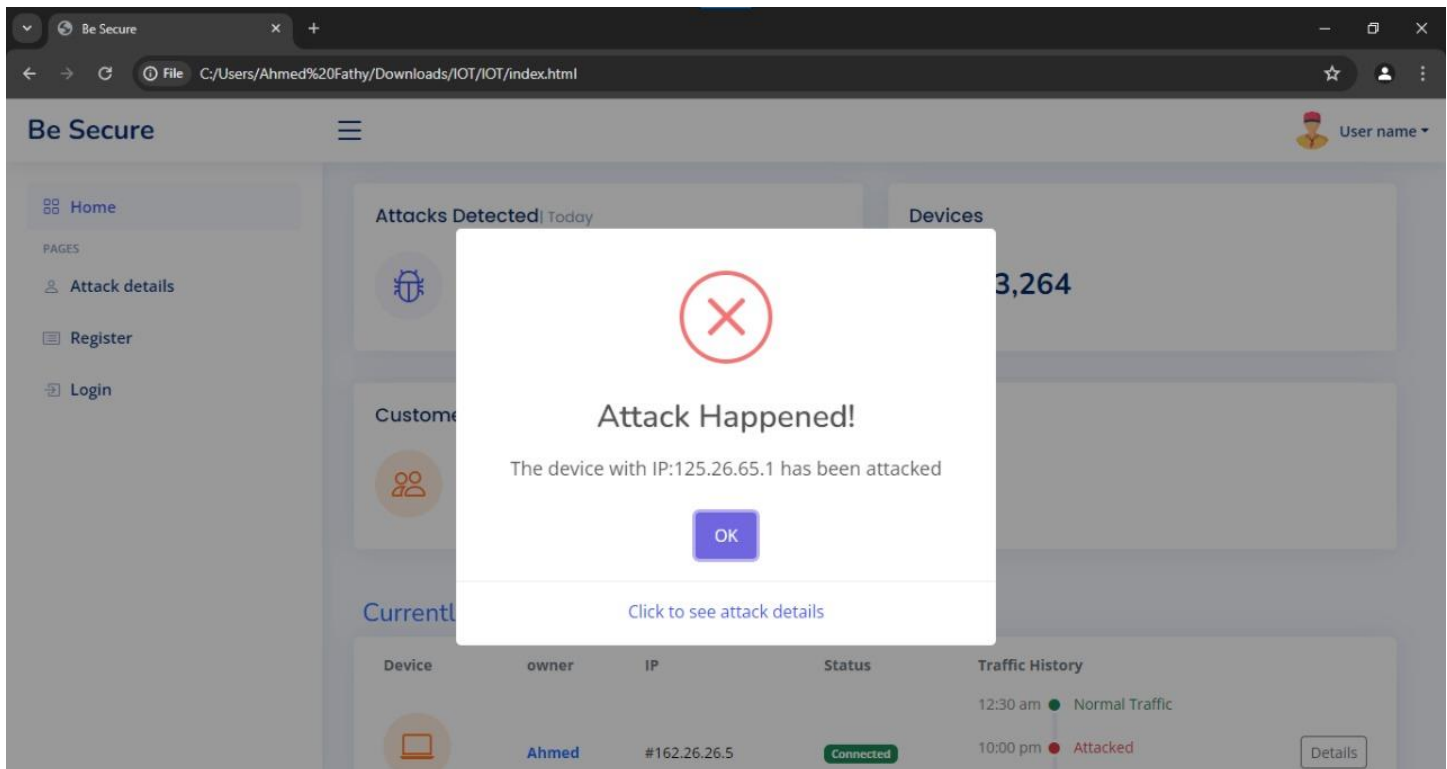
Confirm Password

**Create Account**

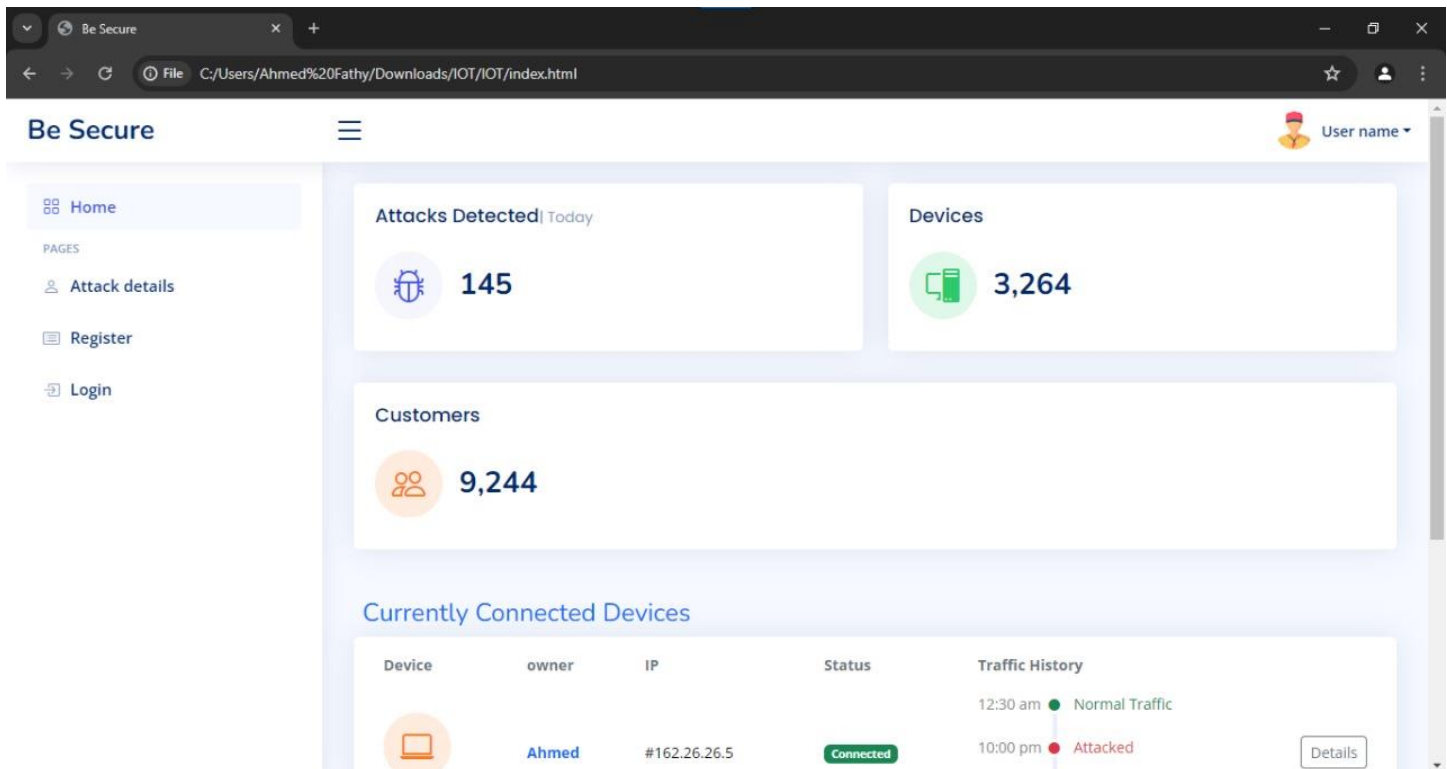
The "Be Secure" registration page allows users to create an account by entering their personal details such as first name, last name, email, username, and password. The form includes client-side validation to ensure correct input and uses Firebase for user authentication. Upon successful registration, users are redirected to the login page. The page is styled with Bootstrap and other CSS libraries, providing a clean and responsive user interface.



The "Be Secure" login page allows users to access their accounts by entering their email and password. It includes client-side validation to ensure proper input and uses Firebase for user authentication. The page provides a clean and responsive interface, styled with Bootstrap and other CSS libraries. If the login is successful, users are redirected to the main page. There are also options for users to remember their login and to create a new account if they don't have one.

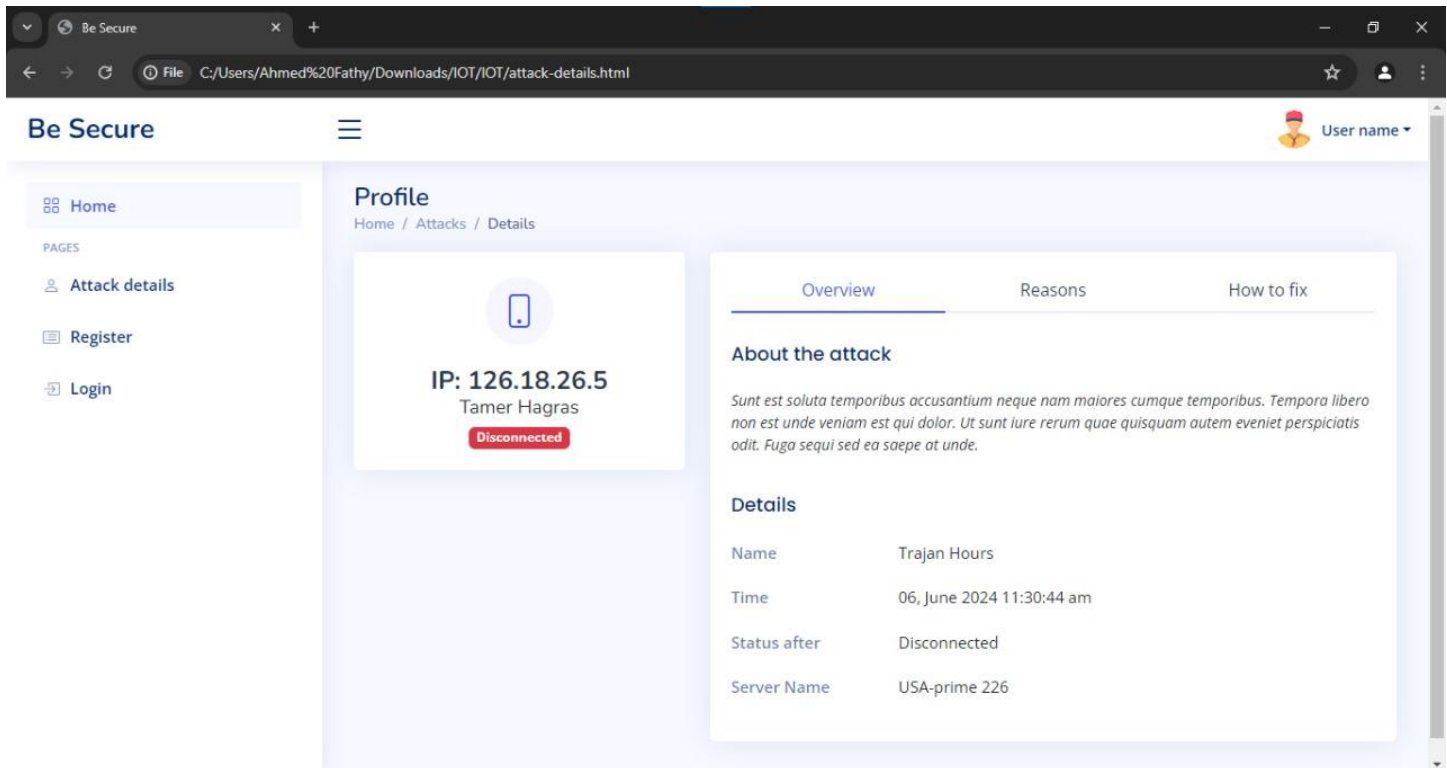


The "Be Secure" attack details page provides a detailed overview of a specific cyber-attack. It includes information about the attack, such as the attacker's IP address, name, and status (e.g., disconnected). The page is structured with a main section and a sidebar for navigation. Users can view various details and tabs, including an overview of the attack, reasons behind it, and suggestions on how to fix it. The page is styled using Bootstrap and other CSS libraries, ensuring a clean and professional appearance.

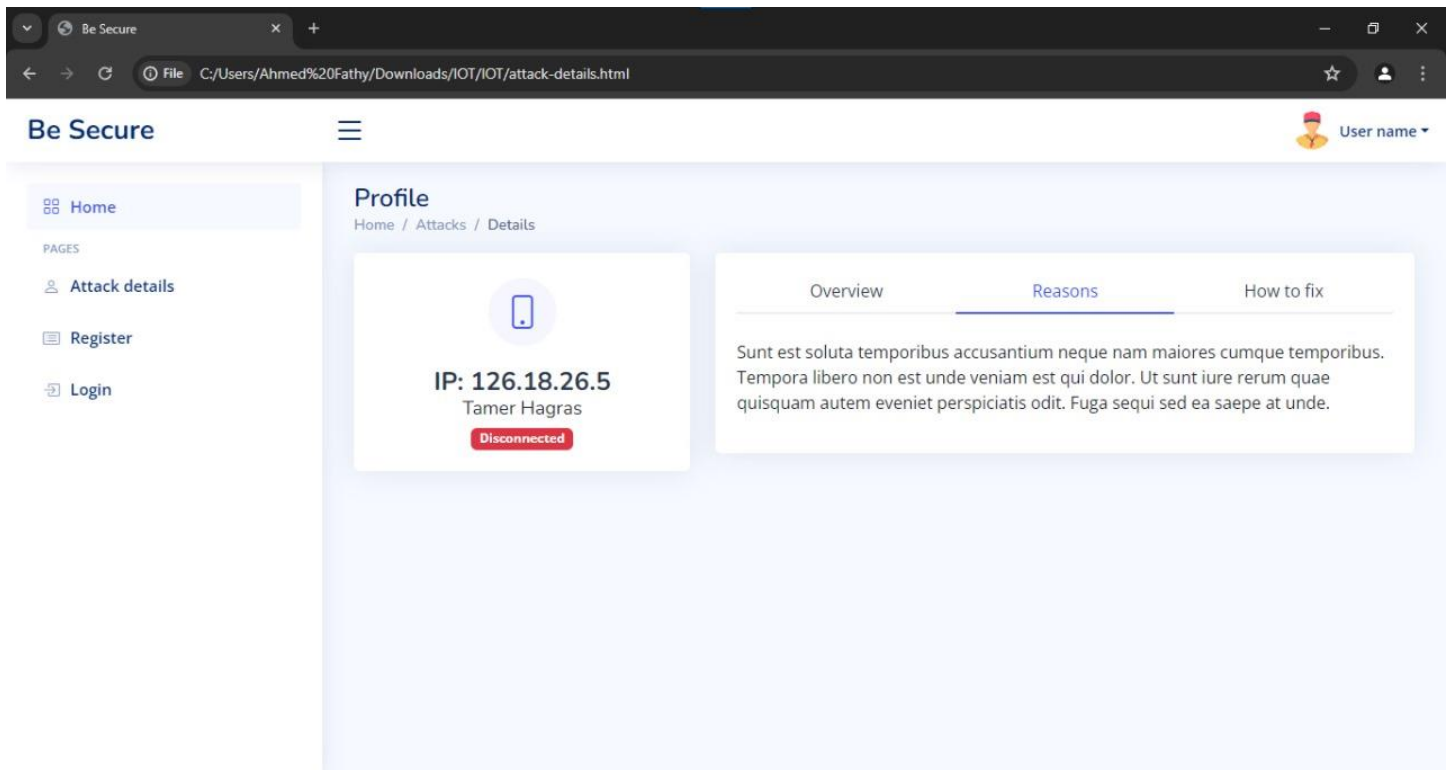


The “Be Secure” home page provides an overview of security-related information and features. The page displays the number of attacks detected on the current day, the total number of connected devices, and the number of registered customers. It also shows specific device information, including the device owner, IP address, status, and traffic history. Users can monitor the status of their devices, track traffic history, and identify any attacks or abnormal activities. The page aims to promote a secure environment by providing insights into security-related metrics and facilitating user management and device monitoring.

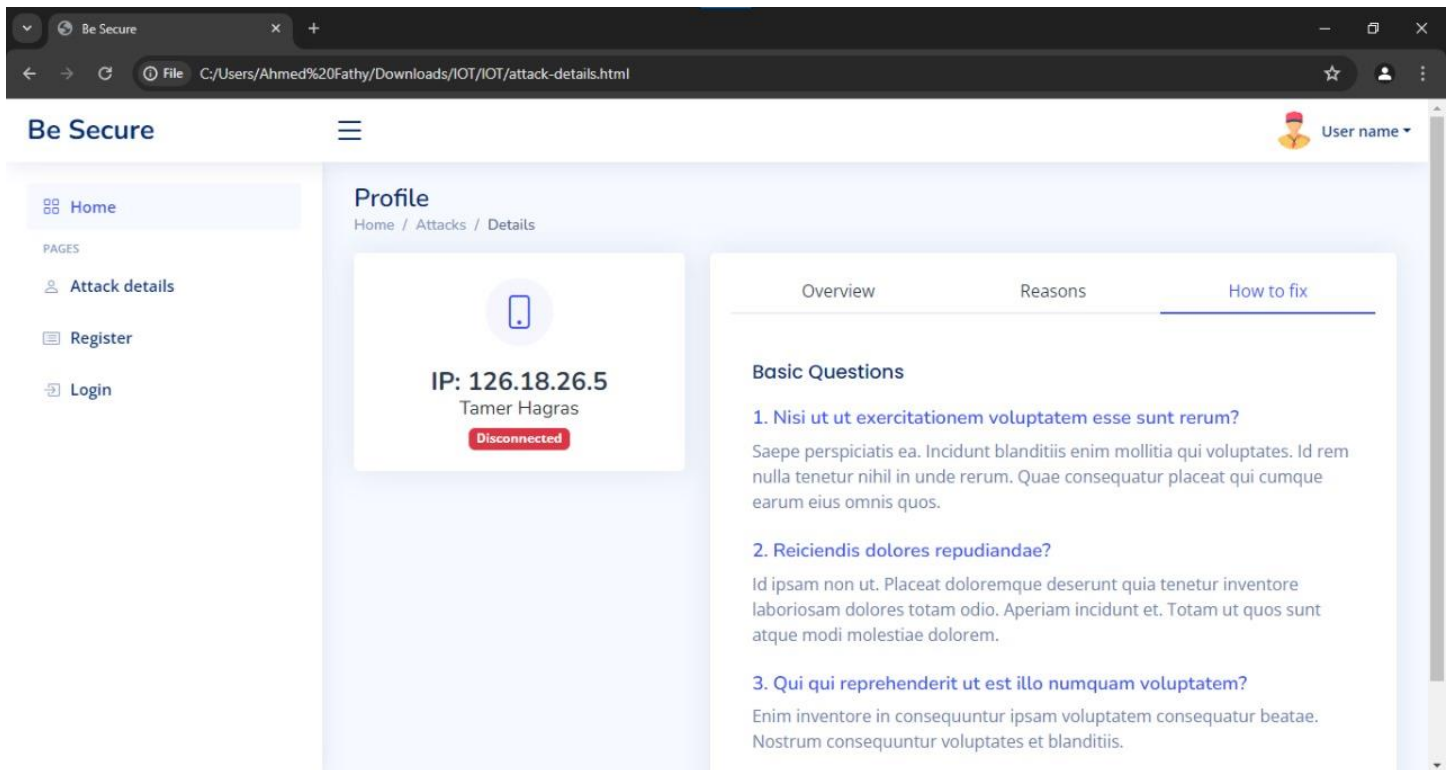




The overview page displays information about every device and some device details like: Attack name, the time of attacking, the status that the device has been after the attack and the server's name.



This page displays the reasons or the weakness existing in this device.



**Be Secure**

Home / Attacks / Details


**Profile**

IP: 126.18.26.5  
Tamer Hagra  
**Disconnected**

**Basic Questions**

- 1. Nisi ut ut exercitationem voluptatem esse sunt rerum?**  
Saepe perspicatis ea. Incidunt blanditiis enim mollitia qui voluptates. Id rem nulla tenetur nihil in unde rerum. Quae consequatur placeat qui cumque earum eius omnis quos.
- 2. Reiciendis dolores repudiandae?**  
Id ipsam non ut. Placeat doloremque deserunt quia tenetur inventore laboriosam dolores totam odio. Aperiam incidunt et. Totam ut quos sunt atque modi molestiae dolorem.
- 3. Qui qui reprehenderit ut est illo numquam voluptatem?**  
Enim inventore in consequuntur ipsam voluptatem consequatur beatae. Nostrum consequuntur voluptates et blanditiis.

This page displays how to fix the attack that happened and gives some advice to be secure.

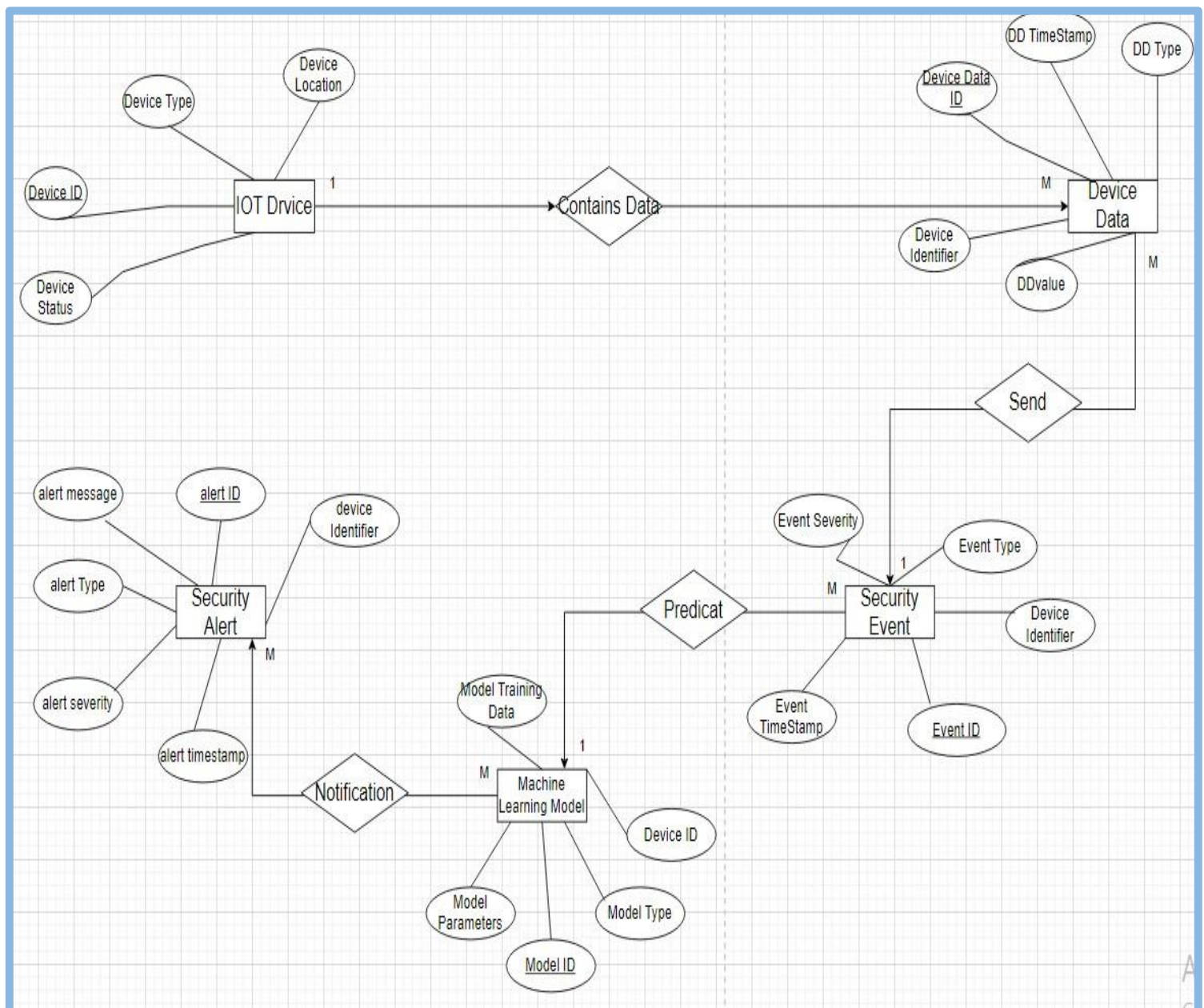


# Chapter 6

# Software

# Requirements

## ERD



### What is the ERD Diagram:

An Entity-Relationship Diagram (ERD) is a blueprint used to visualize a database's structure. It shows the different entities (like tables) that hold data, the attributes (columns) within those entities, and the relationships between them. Here's a breakdown of what an ERD typically includes:

**Entities:** Represented by rectangles, these are the core data categories in your system, often corresponding to tables in a relational database.

Examples include "Customer," "Order," or "Product."

**Attributes:** Shown as ovals within rectangles, these are the specific characteristics that define an entity. For instance, a "Customer" entity might have attributes like "customer ID," "name," and "address."

**Relationships:** Depicted as diamonds connecting entities, these represent the associations between different data categories. There are various types of relationships, such as "one-to-one," "one-to-many," or "many-to-many." Lines connect the diamonds to the relevant entities.

ERDs are valuable tools for database design because they:

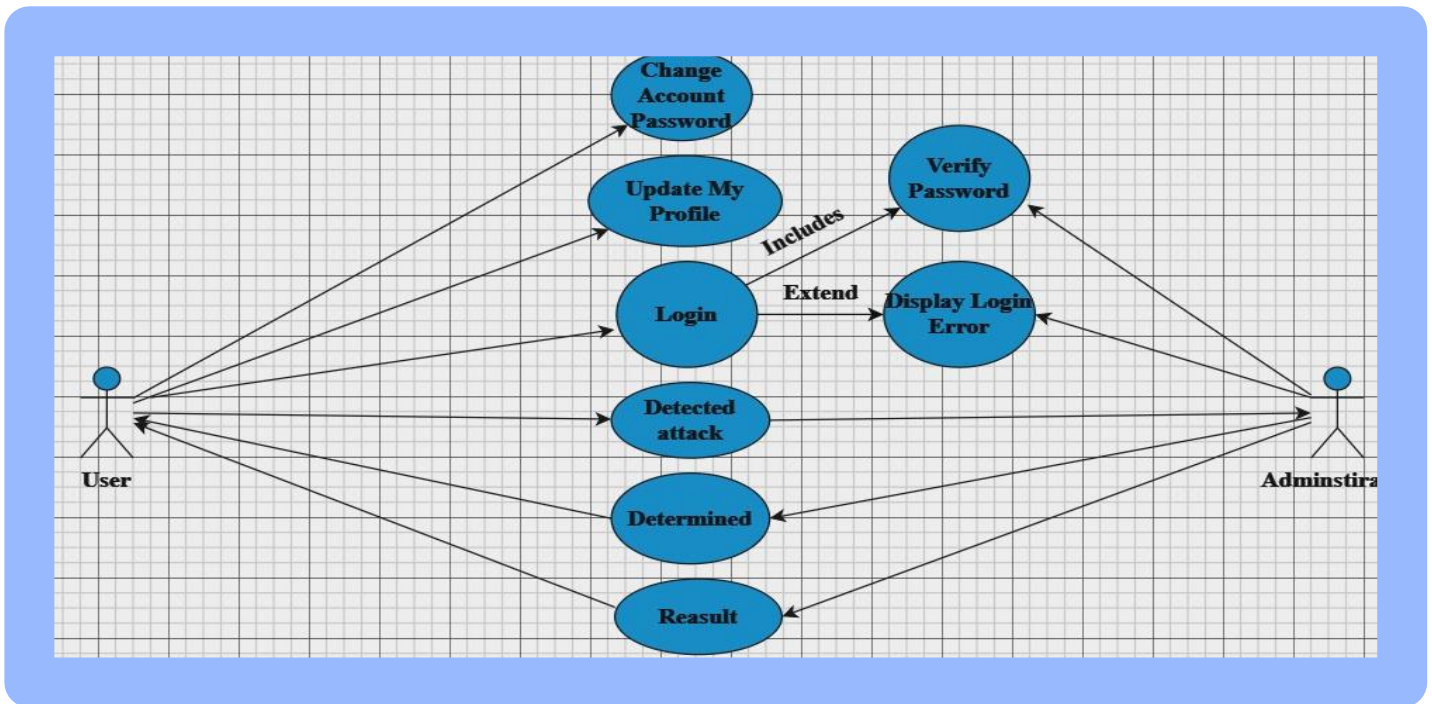
**Improve communication:** An ERD provides a clear visual representation of the database structure, making it easier for developers, designers, and other stakeholders to understand how the data is organized.

**Reduce errors:** By visualizing the relationships between entities, ERDs can help identify potential inconsistencies or flaws in the database design before it's implemented.

**Facilitate planning:** ERDs can be used to plan for future growth and modifications to the database structure.

If you'd like to learn more about ERDs, you can find resources online that include visual examples and tutorials on how to create them using diagramming tools

# USE CASE



**What is the Use Case:** A use case diagram is a visual representation of how users (or external systems) interact with a system to achieve specific goals. It's like a roadmap that shows the functionalities of a system from the user's perspective, focusing on "what" the system does rather than "how" it does it.

Here's a breakdown of the key elements in a use case diagram:

**Actors:** These represent the external entities that interact with the system. Actors can be human users like customers,



administrators, or even other software systems. They are depicted as stick figures or icons.

**Use Cases:** These are the functionalities or tasks that the system offers. They are shown as ellipses and represent a complete sequence of interactions between an actor and the system to achieve a specific goal. For example, "Place Order" or "Search Product" could be use cases.

**Relationships:** Arrows connect actors to use cases, indicating which actors participate in each use case. There are different types of relationships, such as includes (one use case incorporates another) and extends (a use case modifies another).

Use case diagrams are beneficial for several reasons:

**Improved Communication:** They provide a clear, shared understanding of the system's functionalities among developers, designers, and stakeholders.

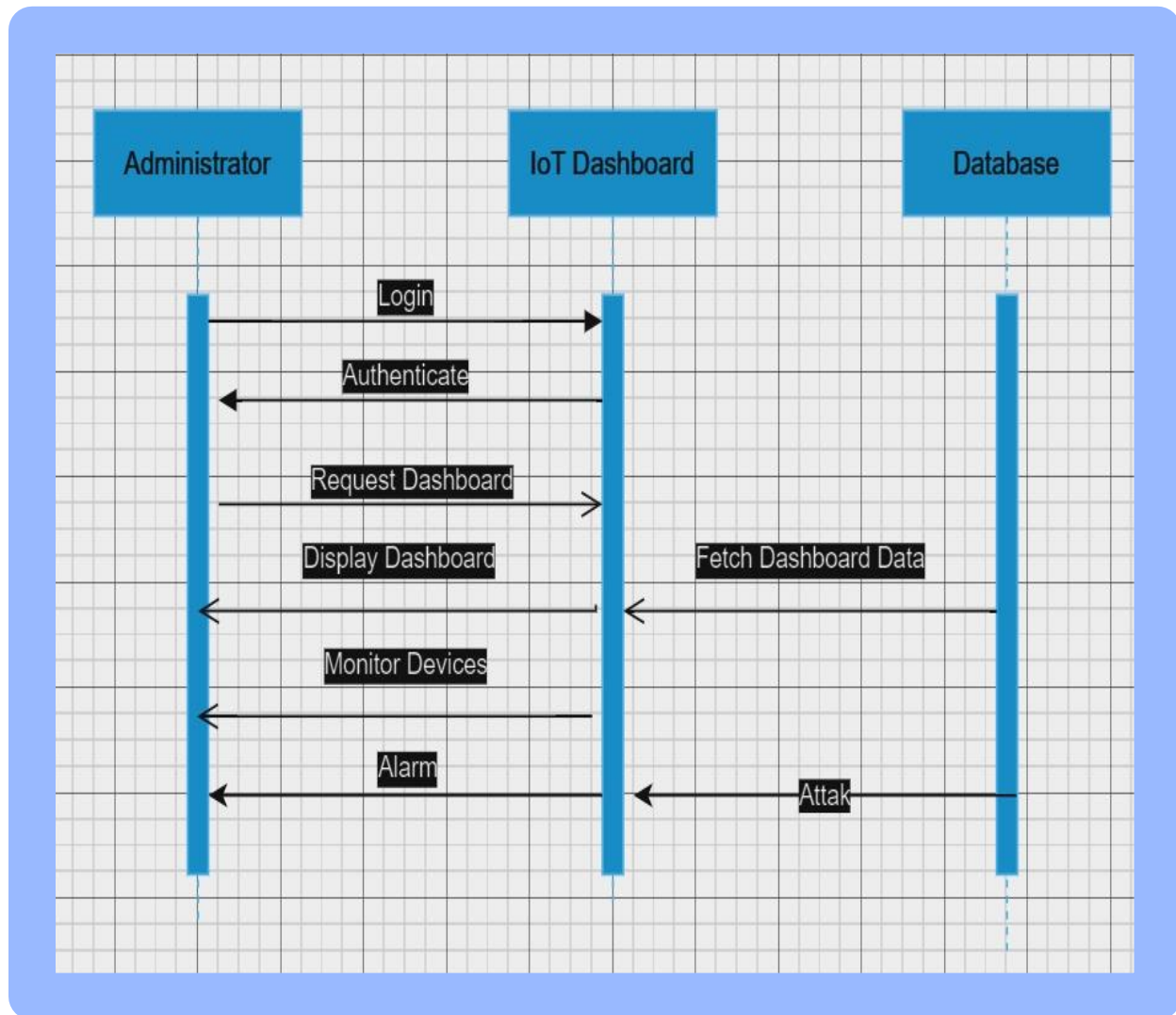
**Requirement Capture:** They help identify and document the system's functional requirements from a user-centric perspective.

**System Validation:** They serve as a basis for validating the system's architecture and ensuring it meets user needs.

**Test Case Generation:** Use cases can be used to derive test cases that ensure the system functions as intended.



# SEQUENCE DIAGRAM



## What is the sequence diagram:


A sequence diagram, like a use case diagram, focuses on interactions within a system, but it dives deeper to show how objects communicate with each other to achieve a specific functionality. It's essentially a step-by-step visual representation of the message exchange between objects involved in a particular scenario.

Here are the key elements of a sequence diagram:

**Lifelines:** Represented by vertical lines, these depict the objects participating in the interaction. The lifelines run along the top of the diagram, from beginning to end, signifying the objects' existence throughout the scenario.

**Messages:** Shown as arrows between lifelines, these represent the communication between objects. The arrows indicate the flow of messages, including requests, responses, and method calls.

**Time Sequence:** The order of messages is crucial in a sequence diagram. Messages are placed chronologically along the lifelines, showcasing the sequence of interactions.



# **Chapter 7**

## **Implementation and coding**

## Analyzing the system:

System analysis is the process of examining and studying all parts of the system, and how they perform their work. The concept of the system in this context includes individuals, machines, and elements that collectively make up the system and have an effective role in achieving the desired goal for specific job.

The importance of system analysis is to divide the complex system in its structure into its main components in a logical manner, considering the scope of the system, its objectives, and the organizational framework of the system, and this comes on the sidelines of the analyst's attempt to seek to develop and improve the system. Below we try to explain some steps of our system with (ERD Diagram - Network Diagram - Activity Diagram), because that will be able to help the readers to understand the new system more easily.

### Entity Relationship Diagram (ERD):

is a visual representation of the relationships between entities in a database. An entity is a person, place, thing, event, or concept about which data is Stored. The relationships between entities are represented by lines, connecting them, and the nature of the relationship is indicated by the type of line and any symbols used.

## Here are some key components of an ERD:

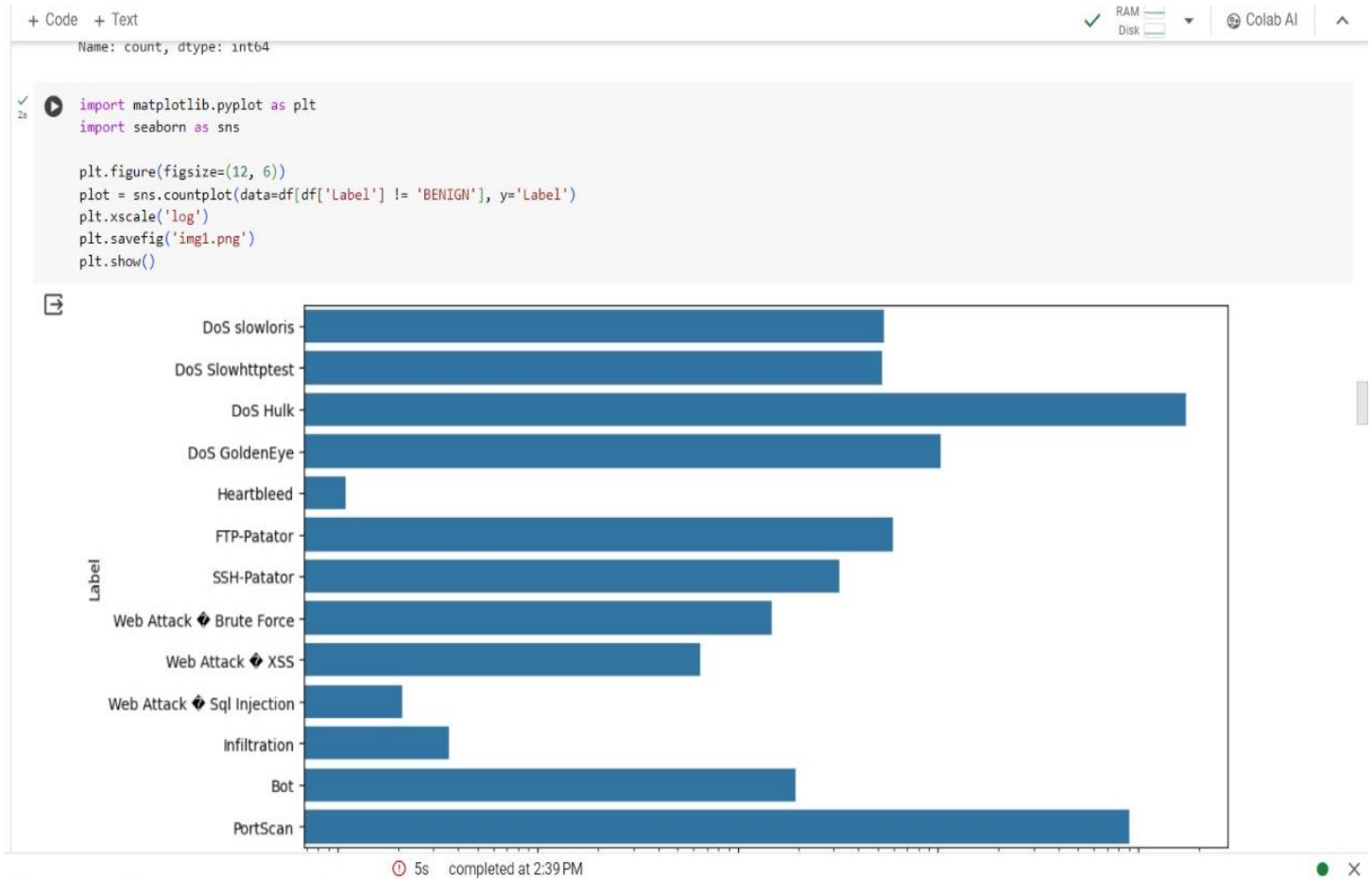
- 1. Entities:** These are the objects or concepts that are represented in the Database. Each entity is represented by a rectangle with its name inside.
- 2. Attributes:** These are the characteristics or properties of an entity. Each Attribute is represented by an oval connected to the corresponding entity.
- 3. Relationships:** These are the connections between entities, representing how they relate to each other. Relationships are represented by lines Connecting two entities, and can be one-to-one, one-to-many, or many-to Many.
- 4. Cardinality:** This refers to the number of instances of one entity that can be associated with another. Cardinality is indicated by symbols near the end of the relationship lines.
- 5. Primary key:** This is a unique identifier for each entity, used to Distinguish it from other entities in the database. The primary key is Usually Underlined in the ERD.

ERDs are commonly used in software development and database design to help visualize the structure of the database and how the various entities relate to each other. They can be used to identify potential issues with the design, such as redundant data or inconsistent relationships, and to help ensure that the database is organized in an efficient and logical manner.

## Dataset cleaning

```
def data_cleaning(df):  
    df.columns=df.columns.str.strip()  
    print("Dataset Shape: ",df.shape)  
  
    num=df._get_numeric_data()  
    num[num<0]=0  
  
    zero_variance_cols=[]  
    for col in df.columns:  
        if len(df[col].unique()) == 1:  
            zero_variance_cols.append(col)  
    df.drop(columns=zero_variance_cols,axis=1,inplace=True)  
    print("Zero Variance Columns: ",zero_variance_cols, " are dropped!!")  
    print("Shape after removing the zero variance columns: ",df.shape)  
  
    df.replace([np.inf,-np.inf],np.nan,inplace=True)  
    print(df.isna().any(axis=1).sum(), "rows dropped")  
    df.dropna(inplace=True)  
    print("Shape after Removing NaN: ",df.shape)  
  
    df.drop_duplicates(inplace=True)  
    print("Shape after dropping duplicates: ",df.shape)  
  
    column_pairs = [(i,j) for i,j in combinations(df,2) if df[i].equals(df[j])]  
    ide_cols=[]  
    for col_pair in column_pairs:  
        ide_cols.append(col_pair[1])  
    df.drop(columns=ide_cols,axis=1,inplace=True)  
    print("Columns which have identical values: ",column_pairs," dropped!")  
    print("Shape after removing identical value columns: ",df.shape)  
    return df  
df=data_cleaning(df)
```

## Name of attacks





## Name of devices in dataset

586712	53	266	2	2	70
365782	53	31070	1	1	57
62189	21	4	2	0	14

10 rows × 80 columns

✓  
0s



```
df["DeviceName"].value_counts()
```



```
DeviceName
PC-1    692703
PC-5    529918
PC-2    445909
PC-4    288602
PC-7    286467
PC-6    191033
PC-3    170366
Name: count, dtype: int64
```

## Register in website

```

1 <section class="section register min-vh-100 d-flex flex-column align-items-center justify-content-center py-4">
2   <div class="container">
3     <div class="row justify-content-center">
4       <div class="col-lg-6 col-md-6 d-flex flex-column align-items-center justify-content-center">
5         <div class="d-flex justify-content-center py-4">
6           <a href="index.html" class="logo d-flex align-items-center w-auto">
7             <span class="d-none d-lg-block fs-1">Be Secure</span>
8           </a>
9         </div>
10        <div class="card mb-3">
11          <div class="card-body">
12            <div class="pt-4 pb-2">
13              <h5 class="card-title text-center pb-0 fs-4">Create an Account</h5>
14              <p class="text-center small">Enter your personal details to create account</p>
15            </div>
16            <form class="row g-3" id="signup-form">
17              <div class="col-6">
18                <label for="fName" class="form-label">First Name</label>
19                <input type="text" name="firstName" class="form-control" id="fName" required>
20              </div>
21              <div class="col-6">
22                <label for="lName" class="form-label">Last Name</label>
23                <input type="text" name="lastName" class="form-control" id="lName" required>
24              </div>
25              <div id="name-error" class="error"></div>
26
27              <div class="col-12">
28                <label for="yourEmail" class="form-label">Your Email</label>
29                <input type="email" name="email" class="form-control" id="email" required>
30              </div>
31              <div class="col-12">
32                <label for="yourUsername" class="form-label">Username</label>
33                <div class="input-group has-validation">
34                  <span class="input-group-text" id="inputGroupPrepend">@</span>
35                  <input type="text" name="username" class="form-control" id="username" required>
36                </div>
37              </div>
38              <div id="email-error" class="error"></div>
39
40              <div class="col-6">
41                <label for="yourPassword" class="form-label">Password</label>
42                <input type="password" name="password" class="form-control" id="password" required>
43              </div>
44              <div class="col-6">
45                <label for="yourPassword" class="form-label">Confirm Password</label>
46                <input type="password" name="password2" class="form-control" id="password2" required>
47              </div>
48              <div id="password-error" class="error"></div>
49
50              <div class="col-12">
51                <button class="btn btn-primary w-100" type="submit">Create Account</button>
52              </div>
53              <div class="col-12">
54                <p class="small mb-0 text-center">Already have an account? <a href="login.html">Log in</a></p>
55              </div>
56            </form>
57          </div>
58        </div>
59      </div>
60    </div>
61  </section>
62  <a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i
63    class="bi bi-arrow-up-short"></i></a>


```

## login in website

```
1 <body>
2
3 <main>
4 <div class="container">
5
6 <section class="section register min-vh-100 d-flex flex-column align-items-center justify-content-center py-4">
7 <div class="container">
8 <div class="row justify-content-center">
9 <div class="col-lg-4 col-md-6 d-flex flex-column align-items-center justify-content-center">
10
11 <div class="d-flex justify-content-center py-4">
12 <a href="index.html" class="logo d-flex align-items-center w-auto">
13 <span class="d-none d-lg-block fs-1">Be Secure</span>
14 </a>
15 </div><!-- End Logo -->
16
17 <div class="card mb-3">
18
19 <div class="card-body">
20
21 <div class="pt-4 pb-2">
22 <h5 class="card-title text-center pb-0 fs-4">Login to Your Account</h5>
23 <p class="text-center small">Enter your email & password to login</p>
24 </div>
25
26 <form id="login-form" class="row g-3 needs-validation" novalidate>
27
28 <div class="col-12">
29 <label for="yourEmail" class="form-label">Email</label>
30 <div class="input-group has-validation">
31 <span class="input-group-text" id="inputGroupPrepend">@</span>
32 <input type="text" name="email" class="form-control" id="email" required>
33 <div class="invalid-feedback">Please enter your Email.</div>
34 </div>
35 </div>
36 <div id="email-error" class="error"></div>
37
38 <div class="col-12">
39 <label for="yourPassword" class="form-label">Password</label>
40 <input type="password" name="password" class="form-control" id="password" required minlength="8">
41 <div class="invalid-feedback">Please enter your password!</div>
42 </div>
43 <div id="password-error" class="error"></div>
44 <div class="col-12">
45 <div class="form-check">
46 <input class="form-check-input" type="checkbox" name="remember" value="true" id="rememberMe">
47 <label class="form-check-label" for="rememberMe">Remember me</label>
48 </div>
49 </div>
50 <div class="col-12">
51 <button class="btn btn-primary w-100" type="submit">Login</button>
52 </div>
53 <div class="col-12">
54 <p class="small mb-0">Don't have account? <a href="registeration.html">Create an account</a></p>
55 </div>
56 </form>
57
58 </div>
59 </div>
60
61
62 </div>
63 </div>
64 </div>
65
66 </section>
67
68 </div>
69 </main><!-- End #main -->
70
```

## Firestore

```
1  const firebaseConfig = {
2    apiKey: "AIzaSyCB-E-G3oFizHs57jg7t4cGtI8PQVTyTGY",
3    authDomain: "ioot-1fa10.firebaseio.com",
4    projectId: "ioot-1fa10",
5    storageBucket: "ioot-1fa10.appspot.com",
6    messagingSenderId: "865570550327",
7    appId: "1:865570550327:web:d88deae7d4611fb79745b2",
8    measurementId: "G-9PLWLHKX4L"
9  };
10
11  // Initialize Firebase
12  const app = firebase.initializeApp(firebaseConfig);
```



# Chapter 7

## Conclusion

## summary:

This study aims to analyze network traffic using the CICIDS2017 dataset. The dataset was collected for the purpose of evaluating the performance of threat detection and intrusion prevention systems in a controlled environment. It contains a large collection of network traffic, including various known attacks and artificial intrusions.

The project involves the following steps:

**Dataset Exploration:** The dataset is examined to understand its structure and content. This includes verifying the available data, existing variables, and understanding the nature of the included attacks.

**Data Cleaning:** The dataset may contain noisy or incomplete data. It needs to be cleaned and processed to ensure the accuracy and completeness of the data used in the analysis.

**Information Extraction:** Important and useful information is extracted from the data, such as patterns, structures, and different classifications of network traffic.

**Performance Analysis:** The performance of threat detection and intrusion prevention systems is evaluated using the dataset. This is done by applying machine learning techniques and data analysis to determine the systems' ability to detect and prevent known attacks and artificial intrusions.



**Recommendations and Conclusions:** After the analysis, recommendations and conclusions are provided based on the derived results. These recommendations may include improvements to threat detection and intrusion prevention systems or guidelines for enhancing the security of computer networks.

## **Conclusions:**

**Effectiveness of Algorithms:** Evaluate the performance of the employed algorithms in detecting and preventing known attacks and artificial intrusions. Compare their accuracy, precision, recall, and overall effectiveness in identifying malicious network traffic.

**Identification of Threats:** Summarize the types of attacks and intrusions that were successfully detected using the algorithms. Identify the most common or significant threats present in the CICIDS2017 dataset.

**False Positives and False Negatives:** Discuss the occurrence of false positives (legitimate traffic wrongly identified as malicious) and false negatives (malicious traffic not detected) in the analysis. Assess the impact of these errors on the overall performance of the algorithms.

**Recommendations for Improvement:** Based on the findings, provide recommendations for enhancing the effectiveness of threat detection and intrusion prevention systems. This may include refining existing algorithms, incorporating additional features or data sources, or exploring new machine learning techniques.

**Dataset Limitations:** Highlight any limitations or challenges encountered during the analysis of the CICIDS2017 dataset. Discuss potential biases, data quality issues, or any other factors that may have influenced the results.

## Future work:

There are some features we plan to add it in our app in the future like:

1. Add a chat to facilitate communication between project members
2. Add some other features like color change and font size
3. Adding Arabic language to the application.
4. Make a mobile app for the application.
5. Ability to solve the attacks happened



# References

## References

- [1] <https://firebase.google.com/>
- [2] <https://www.w3schools.com/>
- [3] <https://www.kaggle.com/>
- [4] <https://scholar.google.com/>
- [5] <https://colab.research.google.com/>
- [6] <https://laravel.com/docs/11.x>
- [7] <https://www.tableau.com/data-insights/ai/algorithms>
- [8] <https://ieeexplore.ieee.org/abstract/document/8455902>
- [9] <https://ieeexplore.ieee.org/abstract/document/8985492>
- [10] <https://ieeexplore.ieee.org/abstract/document/871937>
- [11] <https://ieeexplore.ieee.org/abstract/document/906014>
- [12] [https://link.springer.com/chapter/10.1007/978-981-16-8550-7\\_47](https://link.springer.com/chapter/10.1007/978-981-16-8550-7_47)
- [13] <https://www.lucidchart.com/pages/how-to-draw-ERD>