A Project Report on

Securing SCADA Systems: Advanced Threat Mitigation and Resilient Cybersecurity Strategies for Critical Infrastructure

Submitted in the partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

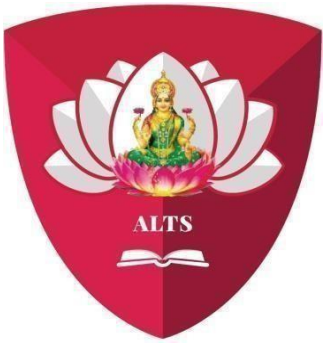
by

GANGULA BHAVITHA REDDY	212G1A3913
UPPARA BHAVITHA	212G1A3954
VIRUPAKSHI HAMPI AMARESH	222G5A3909
VADDE SUDHA RANI	212G1A3956
CHITYALA SUMANTH REDDY	222G5A3901

Under Supervision of

Mr. D. Lakshmi Narayana Reddy M. Tech., (Ph.D.)

Assistant Professor in Computer Science and Engineering



DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

ANANTHA LAKSHMI INSTITUTE OF TECHNOLOGY AND SCIENCES

Approved by AICTE, New Delhi & Affiliated to J.N.T.U. Ananthapuram, Accredited by NAAC

Near S.K. University, Itikalapalli (V), Anantapur (Dt) – 515 721.A.P.

(2021-2025)

ANANTHA LAKSHMI INSTITUTE OF TECHNOLOGY AND SCIENCES

Approved by AICTE, New Delhi & Affiliated to J.N.T.U. Ananthapuram, Accredited by NAAC

Near S.K. University, Itikalapalli(v), Anantapur (v) – 515 721.A.P.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

CERTIFICATE

This is to certify that the "Securing SCADA Systems: Advanced Threat Mitigation and Resilient Cybersecurity Strategies for Critical Infrastructure" is bonafied work carried out by following students of this institute under guidance of Guide Mr. D. Lakshmi Narayana Reddy M. Tech., (Ph. D) for the partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Artificial Intelligence & Machine Learning** from **ANANTHA LAKSHMI INSTITUTE OF TECHNOLOGY AND SCIENCES**, Itikalapalli (V), Anantapur in the academic year of 2024-2025.

GANGULA BHAVITHA REDDY	**212G1A3913**
UPPARA BHAVITHA	**212G1A3954**
VIRUPAKSHI HAMPI AMARESH	**222G5A3909**
VADDE SUDHA RANI	**212G1A3956**
CHITYALA SUMANTH REDDY	**222G5A3901**

Name of the Supervisor
Mr. D. Lakshmi Narayana Reddy M. Tech., (Ph. D.)
Assistant Professor,
Department of Computer science and Engineering,
Anantha Lakshmi Institute of Technology and Sciences, Anantapur.

Head of the Department
Dr. K. Bhargavi M. Tech., Ph.D.
Associate Professor & HOD,
Department of Computer science and Engineering,
Anantha Lakshmi Institute of Technology and Sciences, Anantapur.

Viva Voce Conducted on: _____

Internal Examiner **External Examiner**

DECLARATION

We here by declare that the work which is being presented in this dissertation entitled **"Securing SCADA Systems: Advanced Threat Mitigation and Resilient Cybersecurity Strategies for Critical Infrastructure"** submitted towards the partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in the **ARTIFICIAL INTELLIGENCE & MACHINE LEARNING, ANANTHA LAKSHMI INSTITUTE OF TECHNOLOGY AND SCIENCES** is an authentic work carried out by us during 2024-2025 under the supervision of **Mr. D Lakshmi Narayana Reddy** M. Tech., (Ph. D) **Assistant Professor, Department of Computer Science and Engineering**, Anantha Lakshmi Institute of Technology and Sciences, Itikalapalli(V), Anantapur.

The matter embodied in this dissertation report has not been submitted by us for the award of any other degree or diploma. Further, the technical details furnished in the various chapters in this thesis are purely relevant to the above project.

With Gratitude

GANGULA BHAVITHA REDDY	**212G1A3913**
UPPARA BHAVITHA	**212G1A3954**
VIRUPAKSHI HAMPI AMARESH	**222G5A3909**
VADDE SUDHA RANI	**212G1A3956**
CHITYALA SUMANTH REDDY	**222G5A3901**

ACKNOWLEDGEMENTS

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we have now the opportunity to express our gratitude for all of them.

It is with immense pleasure that we would like to express my indebted gratitude to **Mr. D. Lakshmi Narayana Reddy** M.Tech., (Ph.D.) Assistant Professor, **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**, who has guided us a lot and encouraged us in every step of Internship. We thank him/her for the stimulating guidance, constant encouragement and constructive criticism which have made possible to bring out this project work.

It is with immense pleasure that we would like to express my indebted gratitude to **Dr. K. Bhargavi** M.Tech., Ph.D., **Associate Professor & HOD**, **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**, for giving me an opportunity to work on a project that was so challenging and interesting for me. we remember with great emotion. the constant encouragement and help extended to me by her that went even beyond the realm of academics.

We wish to convey our special thanks to **Dr. B. M. G. PRASAD** M. Tech., Ph.D.**, Dean of CSE, Anantha Lakshmi Institute of Technology and Sciences** for giving the required information in doing my project work.

I wish to convey my special thanks to **Dr. N. Ramamurthy** M. Tech., Ph. D., **Principal** of **Anantha Lakshmi Institute of Technology and Sciences** for giving the required information in doing my project work. Not to forget, we thank all other faculty and non-teaching staff, and my friends who had directly or indirectly helped and supported us in completing my project in time.

We also express our sincere thanks to **Sri M. Anantha Ramudu, Chairman** and **Sri M. Ramesh Naidu, Vice Chairman** of **Anantha Lakshmi Institute of Technology and Sciences** for providing excellent facilities**.**

With Gratitude

GANGULA BHAVITHA REDDY	**212G1A3913**
UPPARA BHAVITHA	**212G1A3954**
VIRUPAKSHI HAMPI AMARESH	**222G5A3909**
VADDE SUDHA RANI	**212G1A3956**
CHITYALA SUMANTH REDDY	**222G5A3901**

ABSTRACT

SCADA systems control important services like power and water, so it's very important to keep them safe. If these systems are not secure, hackers could cause serious problems. Many SCADA systems have weaknesses, especially in how they communicate with each other. The communication often doesn’t have strong protections like encryption (to hide data) or authentication (to check who’s using the system). Instead, they just rely on being separate from the internet, which isn’t enough to stop attacks. This makes it easier for hackers to break in. It’s crucial to improve the security of SCADA systems and find ways to prevent attacks.

LIST OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Acronym	Abbreviation
LSVM	Linear Support Vector Machine
KSVM	Kernel Support Vector Machine
KNN	K-Nearest Neighbor
DT	Decision Tree Classifier
RF	Random Forest Classifier
MSE	Mean Squared Error
PCA	Principal Component Analysis
CNN	Convolutional Neural Network
SVM	Support Vector Machine
EDA	Exploratory Data Analysis
DFD	Data Flow Diagram
UML	Unified Modelling Language

CHAPTER - 1

INTRODUCTION

The past two decades have witnessed remarkable advancements in the fields of computing and communication. Any system has the potential to be considered critical when its vulnerabilities evolve into threats that can cause chaos across social structures, energy sectors, security frameworks, healthcare systems, and various other dimensions of society. The breakdown or unavailability of a system's functions can have catastrophic consequences on society, the economy, and overall stability. Traditionally, the focus of infrastructure security was primarily centered on environmental risks.

Cyberattacks, despite this, are a reality and have switched the focus to further risks and damages. The attackers attempt to exploit network and Internet weaknesses. Critical Infrastructure (CI) 's susceptibility to cyber threats has necessitated the development of modern security solutions. Inaccessibility or breakdown The cascading failures caused by a single CI can bring massive havoc and harm to society, the economy, the stability of a nation, and numerous other infrastructures. Traditional security solutions aim to accommodate developing threats that are well-known; nevertheless, Innovative, robust assaults are unavoidable Hence, it is imperative to implement adaptable security strategies in order to counteract these threats. The article delves into the realm of security issues and unresolved queries in this domain.

The persistent increase in cyber threats targeting SCADA systems is a result of factors such as advancing levels of sophistication, ongoing modernization efforts, the constant demand for real-time operations and distribution, and the intricate multi-component architecture of these systems. To enable the complex tracking of interconnected and integrated systems, it becomes essential to develop advanced SCADA systems that align with the requirements of the forthcoming architectural advancements. Commercial manufacturers have enhanced their firewall capabilities to handle SCADA protocols or created SCADA-specific firewalls. Although open-source firewalls are utilized effectively in IT networks, their application in SCADA networks has not been well examined. The authors shed light on the security problems and unresolved issues surrounding SCADA devices and demonstrate the importance of securing them.

Above and beyond, the article reviews SCADA networks and discusses the available security solutions for SCADA network attacks. Supervisory Control and Data Acquisition (SCADA) systems are utilized by critical infrastructures (CIs).

1.1 DEEP LEARNING:

This project leverages deep learning techniques for audio classification and anomaly detection using a combination of LSTM-based auto encoders and fully connected neural networks. The primary goal is to process audio signals, extract meaningful features, and classify them while simultaneously detecting abnormal patterns using reconstruction error. The deep learning model takes Mel spectrograms as input, which are generated from raw audio using Librosa and then normalizes them to improve model performance.

At the core of the model, an LSTM-based auto encoder is employed for anomaly detection. The encoder compresses the audio spectrogram into a latent representation, and the decoder attempts to reconstruct it. By comparing the original and reconstructed versions, a Mean Squared Error (MSE) loss is computed. If the reconstruction error exceeds a predefined threshold, the system flags the audio as anomalous. In parallel, a classification branch uses a soft max layer to categorize the sound into predefined classes such as "fan," "pump," "valve," or "slider." The entire architecture is trained using a multi-task learning approach, where Sparse Categorical Cross-Entropy Loss is used for classification and MSE Loss for reconstruction.

The model is optimized using the Adam optimizer, and early stopping is implemented to prevent over fitting. The training process involves batch processing and a validation split to ensure generalization. When a new audio file is uploaded, it undergoes the same preprocessing steps and is fed into the trained model. The classification output determines the type of sound, while the reconstruction error helps detect anomalies. This deep learning approach enables accurate sound recognition and anomaly detection, making it highly effective for industrial applications, predictive maintenance, and fault detection in machinery.

1.1.1 LSTM-Based Model for Anomaly Detection:

An autoencoder is a neural network designed to learn a compressed representation of input data and reconstruct it with minimal error. In this project, an LSTM-based autoencoder is used to detect anomalies in audio signals. The encoder processes Mel spectrograms of normal audio and compresses them into a low-dimensional latent space.

The decoder then reconstructs the input from this compressed representation. The difference between the original and reconstructed spectrogram is measured using Mean Squared Error (MSE). If the reconstruction error exceeds a threshold, the sound is classified as anomalous, indicating potential machine faults.

Dense Neural Network for Classification (Soft max Layer)

In this project, a Dense Neural Network (DNN) is used for audio classification, complementing an LSTM-based autoencoder for anomaly detection. The DNN is responsible for identifying different machine sounds, such as fan, pump, valve, and slider, by analyzing Mel spectrogram features extracted from audio recordings. The network consists of multiple fully connected (Dense) layers that learn high- level patterns from the input data. which enhances non-linearity and improves the model's ability to capture complex relationships within the sound features.

The final Softmax layer outputs probability scores for each sound class, allowing the model to classify the machine sound accurately. During training, the model is optimized using the Adam optimizer, and Sparse Categorical Cross-Entropy Loss is used to measure performance. Early stopping is applied to prevent overfitting and improve efficiency. By integrating the DNN for classification with the LSTM-based autoencoder for anomaly detection, this project provides a comprehensive solution for predictive maintenance and fault detection in industrial settings, enabling real-time monitoring and early detection of potential machine failures.

1.1.2 Feature Extraction for Deep Learning

Feature extraction is a crucial step in deep learning, especially for processing audio data. In this project, Mel spectrograms are used as the primary input features for the deep learning model. A Mel spectrogram is generated from raw audio using Librosa, where the waveform is transformed into a time- frequency representation that mimics human auditory perception. The Mel-frequency cepstral coefficients (MFCCs) and log-Mel spectrograms are extracted by applying Short-Time Fourier Transform (STFT), followed by a Mel filter bank to emphasize perceptually relevant frequencies. Additionally, log scaling is applied using librosa. power to db to convert the spectrogram into a logarithmic scale, making it more suitable for deep learning models.

To enhance performance, feature normalization is performed by subtracting the mean and dividing by the standard deviation, ensuring that the model learns effectively from the extracted features. These processed spectrograms serve as inputs to the LSTM-based autoencoder and Dense Neural Network, allowing for accurate classification and anomaly detection in industrial sound monitoring applications. Alongside anomaly detection, the model also classifies sounds into categories such as fan, pump, valve, and slider using a fully connected Dense Neural Network. The output of the LSTM encoder is passed through

Dense layers with Re LU activation, followed by a final Soft max layer that assigns probabilities to each class. This classification branch allows the model to not only detect anomalies but also categorize normal sounds accurately. By combining LSTM-based auto encoders for anomaly detection and Dense Neural Networks with Soft max classification, this model provides a robust solution for predictive maintenance and fault detection in industrial audio applications.

1.2 Artificial Intelligence

This project utilizes Artificial Intelligence (AI) techniques, specifically Deep Learning, to perform audio classification and anomaly detection in industrial environments. The core AI components include a Dense Neural Network (DNN) for sound classification and an LSTM-based autoencoder foranomaly detection, both of which analyze machine sounds to identify normal and abnormal patterns.

The AI system begins with feature extraction, where raw audio signals are converted into Mel spectrograms using Librosa. These spectrograms are then used as inputs for the deep learning models. The DNN, built with fully connected (Dense) layers, classifies sounds into predefined categories such as fan, pump, valve, and slider, using Softmax activation to determine the most likely class. Meanwhile, the LSTM-based autoencoder reconstructs normal machine sounds and calculates the reconstruction error. If the error exceeds a predefined threshold, the AI system detects it as an anomaly, signaling a potential machine fault. The AI models are trained using supervised learning for classification and unsupervised learning for anomaly detection, leveraging techniques such as Adam optimization, dropout regularization, and early stopping to enhance performance.

1.3 Machine Learning

Machine Learning (ML) plays a critical role in this project, particularly in audio classification and anomaly detection for industrial machines. The project utilizes Deep Learning, a subset of ML, to train models that can recognize different machine sounds and detect anomalies in real-time. ML in Audio Classification: The Dense Neural Network (DNN) is trained using supervised learning, where labeled audio samples (e.g., sounds from fans, pumps, valves, and sliders) are used to teach the model how to classify different machine sounds.

The Mel spectrograms extracted from audio recordings serve as features for training the model. Using Soft max activation and Sparse Categorical Cross-Entropy loss, the ML model learns to predict the correct machine type based on its sound patterns. ML in Anomaly Detection: The LSTM-based auto encoder applies unsupervised learning to detect abnormal machine behavior. Feature Engineering & Preprocessing: ML techniques such as data normalization, feature extraction (Mel spectrograms), and encoding help improve model performance. Librosa is used to convert raw audio signals into spectrograms, which act as meaningful features for ML models.

CHAPTER – 2

LITERATURE SURVEY

Literature Survey for Enhancing SCADA Systems Security

The literature on Supervisory Control and Data Acquisition (SCADA) systems security has expanded significantly in recent years, reflecting the increasing importance of securing critical infrastructure against cyber threats. This survey summarizes key findings and trends in the field, focusing on the vulnerabilities of existing SCADA systems and proposed solutions for enhancing security.

1. Vulnerabilities in Existing SCADA Systems

A significant body of research highlights the vulnerabilities inherent in traditional SCADA systems, particularly those relying on outdated communication protocols. For instance, **Stouffer et al. (2011)** describe the lack of built-in security features in widely used protocols such as Modbus and DNP3, which were not designed with cyber threats in mind. These protocols often lack essential security measures like encryption and authentication, making SCADA systems susceptible to various attacks, including man-in-the-middle and denial-of-service (DoS) attacks.

Igure et al. (2006) emphasize the risks associated with the reliance on "security by obscurity." Many organizations assume that isolation from public networks offers sufficient protection, yet the integration of SCADA systems with corporate networks and the internet has rendered this approach ineffective. This shift has led to increased exposure to cyberattacks, particularly as attackers have become more sophisticated.

2. Limitations in Access Control and Incident Response

Research by **Cunningham (2011)** reveals that many SCADA systems lack adequate access control mechanisms, making them vulnerable to unauthorized access, particularly in scenarios involving remote operations. This deficiency is compounded by the absence of robust incident detection and response capabilities, which often leaves SCADA systems unable to identify and mitigate threats in real-time.

Nivethan and Papa (2016) support this notion, illustrating that many SCADA systems are not equipped with modern intrusion detection systems, making them ill-prepared to respond to evolving cyber threats.

Ranathunga et al. (2016) propose the implementation of SCADA-specific firewalls and Virtual Closed Networks (VCNs) to isolate SCADA systems from public networks, significantly reducing the risk of external threats. These firewalls can be configured to allow only authorized communications, effectively creating a secure environment for critical infrastructure.

Ferencz et al. (2024) discuss the integration of automated patch management systems to ensure that SCADA devices are regularly updated, thereby addressing known vulnerabilities. This proactive approach can prevent attackers from exploiting outdated software.Furthermore, **Tapia et al. (2023)** advocate for the adoption of advanced authentication mechanisms, such as multi-factor authentication and role-based access control, to enhance the security of SCADA systems. These measures can significantly reduce the likelihood of unauthorized access and insider threats.

3. Emerging Technologies and Future Directions

Recent research has begun exploring the potential of emerging technologies to bolster SCADA security. For instance,

Alshahrani et al. (2023) highlight the use of artificial intelligence (AI) and machine learning (ML) algorithms for real-time threat detection and incident response. These technologies can analyze network traffic patterns to identify anomalies indicative of cyber threats, allowing for swift mitigation actions.

In addition, the application of blockchain technology for ensuring data integrity and security within SCADA systems is gaining attention. By leveraging decentralized ledgers, organizations can create tamper-proof records of transactions and communications, enhancing overall system security.

Conclusion

The literature on SCADA systems security underscores the critical need for enhanced protective measures in the face of evolving cyber threats. Existing systems suffer from significant vulnerabilities, particularly due to outdated protocols and insufficient security measures. Proposed solutions, such as the integration of secure communication protocols, automated patch management, and advanced access controls, present promising avenues for improving SCADA security.

As emerging technologies like AI and block chain continue to develop, they hold the potential to further fortify SCADA systems against cyber threats, ensuring the resilience and integrity of critical infrastructure in an increasingly connected world.

CHAPTER – 3
SYSTEM ANALYSIS

3.1 SYSTEMSTUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely

depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.2 EXISTING SYSTEM:

The existing SCADA systems used in critical infrastructures are fraught with vulnerabilities that expose them to significant cyber threats. One of the primary issues is the use of outdated and insecure communication protocols, such as Modbus and DNP3, which were designed without encryption or authentication mechanisms, making them easy targets for cyberattacks. Many SCADA systems have also been in operation for decades and suffer from lack of regular updates and patches. These unpatched systems remain exposed to both known and emerging threats.

Furthermore, there is an over-reliance on security by obscurity, where operators assume that simply isolating systems from public networks provides sufficient protection. However, with modern SCADA systems increasingly connected to corporate networks and the internet, this approach is no longer effective, leaving the systems open to cyber intrusions. In addition, the absence of robust access control mechanisms leads to unauthorized access vulnerabilities, particularly when remote access is involved. Finally, the limited capability to detect and respond to cyber threats makes SCADA systems highly vulnerable, as they lack the necessary tools to identify and mitigate attacks in real time.

3.3 Disadvantages of the Existing system:

Outdated Communication Protocols: Many SCADA systems use legacy protocols like Modbus and DNP3, which lack essential security features such as encryption and authentication, making them vulnerable to cyberattacks.

Lack of Regular Software Updates: SCADA systems often run on old software that hasn't been updated in years, leaving them exposed to known security vulnerabilities that can easily be exploited.

Over-Reliance on Isolation ("Security by Obscurity"): Many SCADA systems rely on the assumption that isolation from public networks will protect them. However, with increasing integration into corporate and internet networks, this approach is no longer effective.

Weak Access Control: Remote access to SCADA systems is becoming more common, but these systems frequently lack strong access control mechanisms, making them susceptible to

unauthorized access.

Inability to Detect and Respond to Threats: Existing SCADA systems generally lack advanced monitoring and intrusion detection capabilities, limiting their ability to identify and mitigate cyber threats in real-time.

3.4 PROPOSED SYSTEM:

The proposed system addresses these critical vulnerabilities by implementing modern security solutions designed for today's connected infrastructure environments. One of the main improvements is the adoption of secure communication protocols that incorporate encryption and authentication, ensuring data integrity and confidentiality across the SCADA network. In addition, the proposed system introduces a robust patch management solution, ensuring that all devices and software are regularly updated to protect against new vulnerabilities. The introduction of Virtual Closed Networks (VCN) and SCADA-specific firewalls further strengthens the system by isolating SCADA devices from external networks and filtering unauthorized access attempts.

The proposed system also features role-based access control with multi-factor authentication, ensuring that only authorized personnel can gain access to critical parts of the SCADA infrastructure, significantly reducing the risk of insider threats or unauthorized access. Finally, the integration of real-time threat detection and intrusion response mechanisms enhances the system's ability to monitor and respond to cyber threats, allowing proactive defense against potential breaches. This ensures the system is both secure and resilient in the face of evolving cyber challenges.

3.5 Advantages of Proposed System:

Enhanced Security Protocols: The proposed system incorporates modern communication protocols that utilize encryption and authentication, ensuring secure data transmission and significantly reducing vulnerability to cyberattacks.

Automated Patch Management: With the implementation of automated software updates and patch management, the proposed system ensures that all SCADA devices are consistently updated, protecting them from newly discovered vulnerabilities.

3.6 SYSTEM REQUIREMENTS

3.6.1 HARDWARE REQUIREMENTS:

System	**:**	Inteli7
Hard Disk	**:**	1TB
Monitor	**:**	14’Colour Monitor
Mouse	**:**	Optical Mouse
Ram	**:**	8 GB

3.6.2 SOFTWARE REQUIREMENTS:

Operating System	**:**	Windows 10
Coding Language	**:**	Python
Front-End	**:**	Html, Css
Designing	**:**	Html, Css, Java script.
Data Base	**:**	SQLite.

CHAPTER-4

SYSTEM DESIGN

4.1 DATA FLOW

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

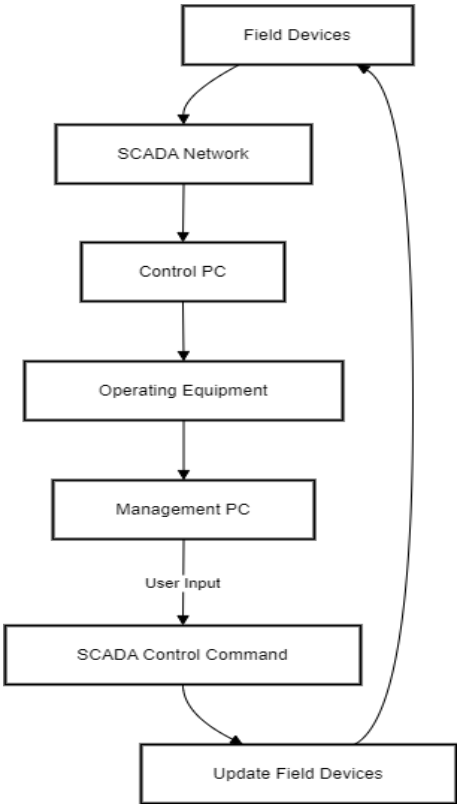


Fig: 4.1 Data Flow Diagram

4.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML .

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

4.2.1 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

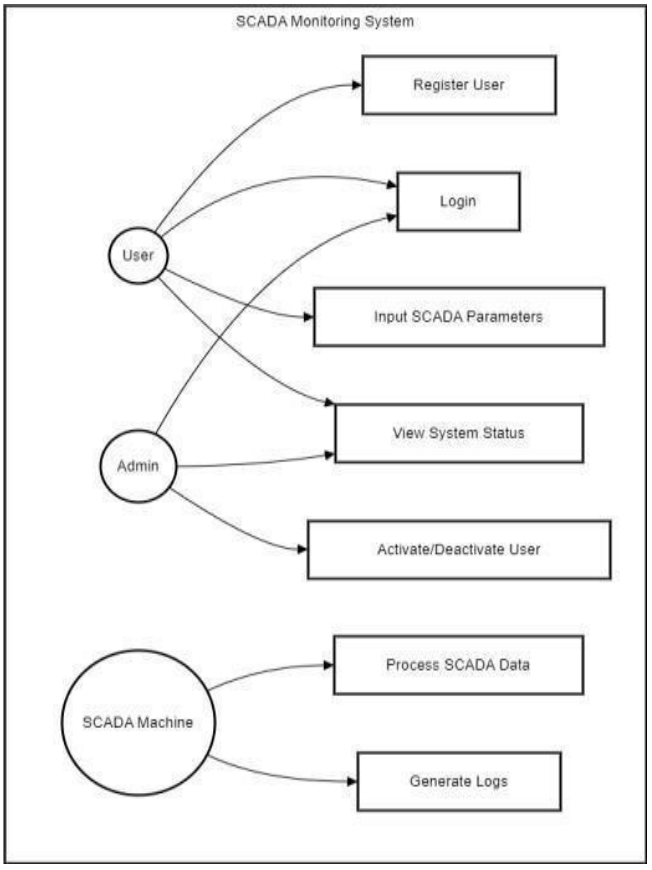


Fig: 4.2 Use Case Diagram

4.2.2 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

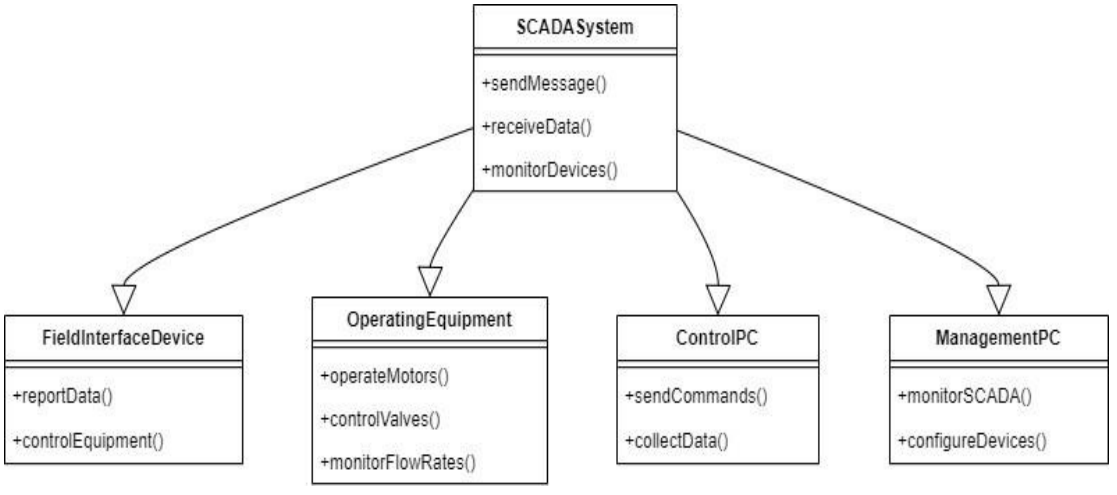


Fig: 4.3 Class Diagram

4.2.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

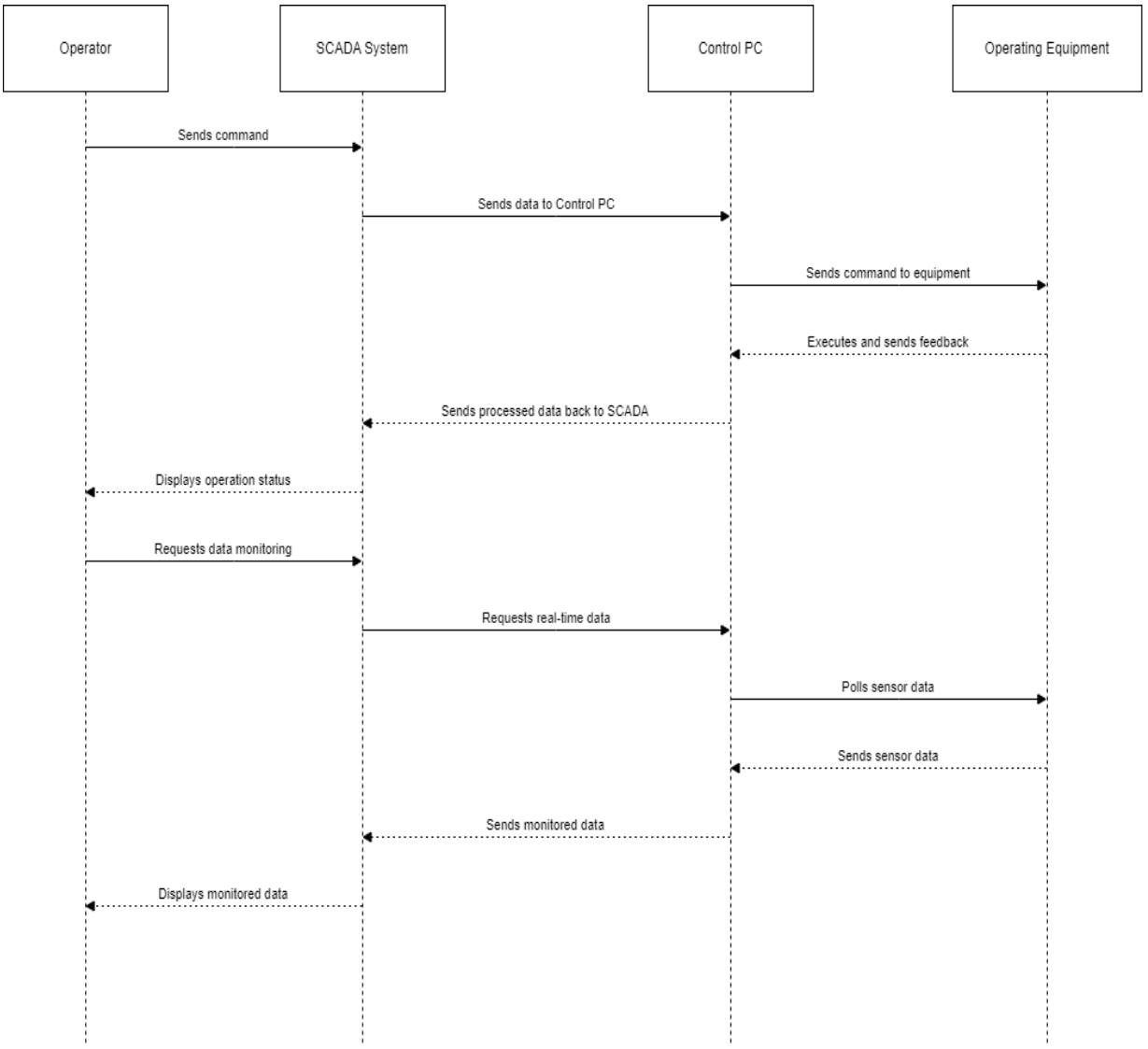


Fig: 4.4 Sequence Diagram

4.2.4 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

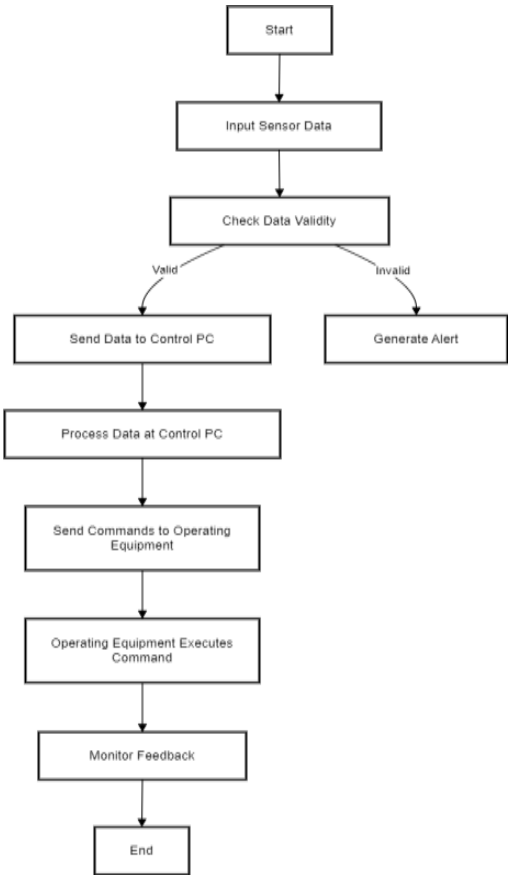


Fig: 4.5 Activity Diagram

CHAPTER-5

IMPLEMENTATION

5.1 MODULES

5.1.1 User Module:

The User Module is designed to provide operators and general users with secure and streamlined access to SCADA system functionalities. This module ensures that users can log in securely through multi-factor authentication, protecting against unauthorized access. Once logged in, users are greeted with a real-time monitoring dashboard that displays key system metrics, including sensor data, system statuses, and alerts. This enables operators to quickly assess the health of the infrastructure. Users have the capability to view both real-time and historical SCADA data, such as sensor readings, system logs, and operational metrics, providing them with insights into the performance and security of critical infrastructure.

5.1.2 Admin Module:

The Admin Module is a critical component that allows administrators to manage and secure the entire SCADA system. This module provides full control over user management, including the creation, deletion, and updating of users and roles. Admins can assign permissions to different users based on their roles, ensuring that only authorized personnel can access sensitive parts of the system. The security settings within the Admin Module are robust, allowing administrators to configure firewall rules and enforce policies to protect SCADA components from external threats. For instance, admins can set up a Virtual Closed Network (VCN) to restrict communication to only trusted devices, effectively shielding the system from cyberattacks like Distributed Denial of Service (DDoS) and unauthorized remote access attempts.

5.1.3 SCADA Module:

The SCADA Module is the core of the system, responsible for controlling, monitoring, and securing the various industrial processes. This module connects to the physical devices and sensors that operate in critical infrastructure environments, such as power grids, water treatment plants, and manufacturing facilities. It interfaces with field devices (e.g., valve actuators, motor controls, sensors) and communicates with control and management systems

to ensure the smooth operation of industrial processes. The SCADA Module is responsible for collecting and processing data from these field devices and distributing it to operators and administrators through control computers and management PCs.

5.2 SYSTEM ARCHITECTURE:

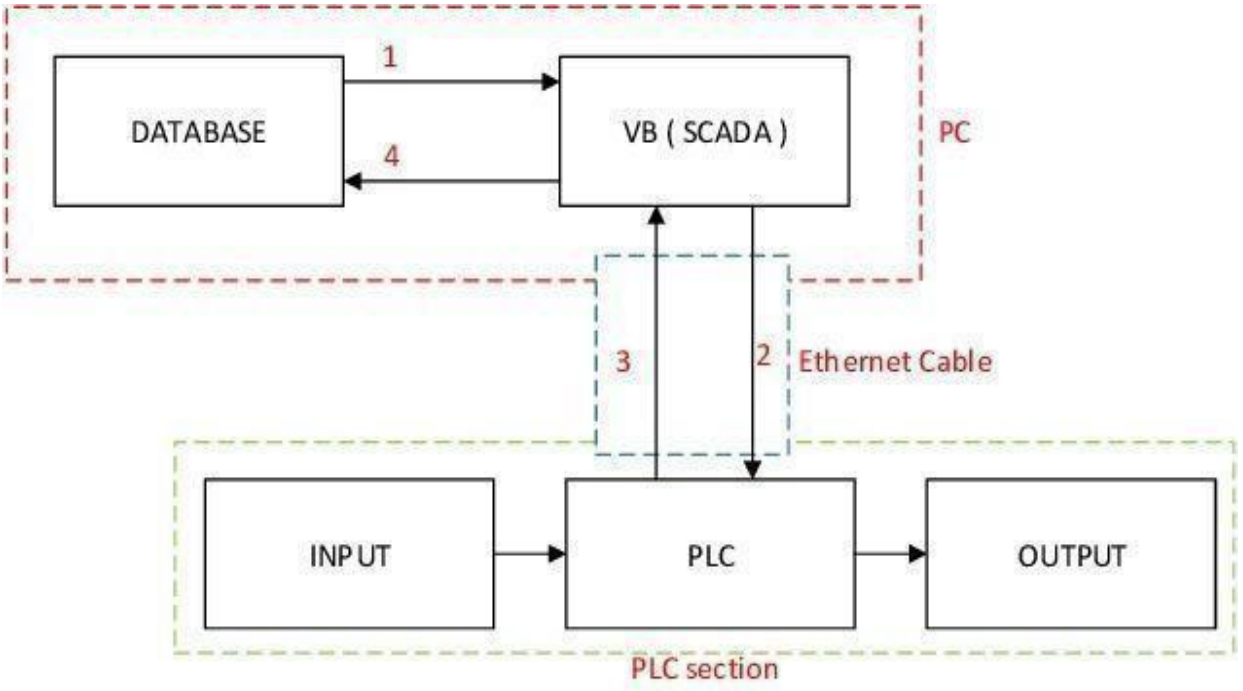


Fig 5.1 System Architecture

The architecture shows a two-step process for zero-shot audio classification:

1. Contrastive Pre training

In this stage, **paired audio and text data** are used to train two separate encoders:

Text Encoder: This module takes text descriptions (e.g., "Paddling in the water") and converts them into text embeddings.

Audio Encoder: This module processes the corresponding audio clips and transforms them into audio embeddings.

Both encoders are trained **contrastively**, meaning they learn to maximize the similarity between matching audio-text pairs and minimize it for non-matching pairs. The output of the training is a **similarity matrix** where:

2. Zero-shot Prediction

Once pretraining is complete, the model can generalize to new datasets or tasks without additional training. The pre trained encoders are employed as follows:

Text Encoder: Takes a set of class labels in textual form (e.g., "Dog barking", "Rain falling", "Siren wailing") and encodes them into the shared embedding space.

5.3 SOURCE CODE

Views code:

```
# Create your views here.
import pandas as pd
from django.shortcuts import render, HttpResponse
from django.contrib import messages
from .forms import UserRegistrationForm
from .models import UserRegistrationModel, TokenCountModel
from django.conf import settings
from django.core.files.storage import FileSystemStorage
from datetime import datetime, timedelta
from jose import JWTError, jwt
import numpy as np
import os
import matplotlib.pyplot as plt
import json
import random


SECRET_KEY =
"ce9941882f6e044f9809bcee90a2992b4d9d9c21235ab7c537ad56517050f26b"
ALGORITHM = "HS256"
def create_access_token(data: dict):
to_encode = data.copy()

# expire time of the token

expire = datetime.utcnow() + timedelta(minutes=15)
to_encode.update({"exp": expire})
encoded_jwt = jwt.encode(to_encode, SECRET_KEY, algorithm=ALGORITHM)
# return the generated token
return encoded_jwt

def verify_token(token: str):
```

```
try:
payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
return payload
except JWTError:
raise HttpResponse(
status_code=HttpResponse(status=204),
detail="Could not validate credentials",
)
# Create your views here.
def UserRegisterActions(request):
if request.method == 'POST':
form = UserRegistrationForm(request.POST)
if form.is_valid():
print('Data is Valid')
loginId = form.cleaned_data['loginid']
TokenCountModel.objects.create(loginid=loginId, count=0)
form.save()
messages.success(request, 'You have been successfully registered')
form = UserRegistrationForm()
return render(request, 'UserRegistrations.html', {'form': form})
else:
messages.success(request, 'Email or Mobile Already Existed')
print("Invalid form")
else:
form = UserRegistrationForm()
return render(request, 'UserRegistrations.html', {'form': form})

def UserLoginCheck(request):
if request.method == "POST":
loginid = request.POST.get('loginid')
pswd = request.POST.get('pswd')
print("Login ID = ", loginid, ' Password = ', pswd)
try:
check = UserRegistrationModel.objects.get(loginid=loginid, password=pswd)
```

```
status = check.status
print('Status is = ', status)
if status == "activated":
request.session['id'] = check.id
request.session['loggeduser'] = check.name
request.session['loginid'] = loginid
request.session['email'] = check.email
data = {'loginid': loginid}
token_jwt = create_access_token(data)
request.session['token'] = token_jwt
print("User id At", check.id, status)
return render(request, 'users/UserHomePage.html', {})
else:
messages.success(request, 'Your Account Not at activated')
return render(request, 'UserLogin.html')
except Exception as e:
print('Exception is ', str(e))
pass
messages.success(request, 'Invalid Login id and password')
return render(request, 'UserLogin.html', {})
def UserHome(request):
return render(request, 'users/UserHomePage.html', {})
```

Base.Html

```
def sendScadaMessage(request):
if request.method == "POST":
hostName = request.POST.get('hostName')
ip = request.POST.get('ip')
pressureValue = request.POST.get('pressureValue')
temperature = request.POST.get('temperature')
flowRate = request.POST.get('flowRate')

switchRate = request.POST.get('switchRate')
valveStatus = request.POST.get('valveStatus')
```

```
pumpStatus = request.POST.get('pumpStatus')
flowIndicator = request.POST.get('flowIndicateor')
msg = {"HostName": hostName,
"IP": ip,
"PressureValue": pressureValue,
"Temperature": temperature,
"FlowRate": flowRate,
"SwitchRate": switchRate,
"ValveStatus": valveStatus,
"PumpStatus": pumpStatus,
"FlowIndicator": flowIndicator,
"status": random.choice(['Normal', "Idle"])
}
machineOperator = json.dumps(msg)
import socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('localhost', 54321)) # Connect to the attacker instead of server
client_socket.sendall(machineOperator.encode())
return render(request, 'users/ScadaRes.html', {"data": msg})
else:
import socket
hostname = socket.gethostname()
# IPAddr = socket.gethostbyname(hostname)
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect(("8.8.8.8", 80))
IPAddr = s.getsockname()[0]
s.close()


print("Your Computer Name is:" + hostname)
print("Your Computer IP Address is:" + IPAddr)
return render(request, "users/sendScadaForm.html", {'hostName': hostname, 'ip': IPAddr})
# Function to convert cursor to a list of dictionaries
def cursor_to_dict(cursor):
```

```
# Get the column names from the cursor
columns = [description[0] for description in cursor.description]
# Fetch all rows and convert each row to a dictionary
return [dict(zip(columns, row)) for row in cursor.fetchall()]
def MachinesLogs(request):
import sqlite3
conn = sqlite3.connect('attacker.db')
cursor = conn.cursor()
data = cursor.execute('"SELECT * FROM ScadaNetwork"')
result = cursor_to_dict(cursor)
print(result, type(result))
df = pd.DataFrame(result)
return render(request, "users/scadalogs.html", {'data': df.to_html(index=False)})
```

user base:

```
{%load static%}
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale= 1.0">
<title>A Comprehensive Analysis and Solutions for Enhancing SCADA Systems Security in
Critical
Infrastructures</title>
<link rel="stylesheet" href="{%static 'css/styles.css'%}">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</head>
<body>
<header>
<h1>A Comprehensive Analysis and Solutions for Enhancing SCADA Systems Security in
Critical Infrastructures</h1>
```

```
<nav>
<ul>
<li class="active"><a href="{% url 'UserHome' %}">Home</a></li>
<li class="active"><a href="{% url 'sendScadaMessage' %}">Scada Message</a></li>
<li class="active"><a href="{% url 'MachinesLogs' %}">Machine Logs</a></li>
<li class="active"><a href="{% url 'index' %}">Logout</a></li>
</ul>
</nav>
</header>
<main>
{%block contents%}
{%endblock%}

</main>
<footer>
<div class="container">
<p>
&copy; All Rights Reserved. Designed and Developed by
<a href="#">Alex Corporations</a>
</p>
</div>
</footer>
</body>
</html>
```

CHAPTER – 6

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 TYPES OF TESTS

6.1.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.1.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.1.3 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.1.4 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two

distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

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

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered

6.1.7 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered

6.2 TESTCASES:

Sample Test Cases

Test Case ID	Test Description	Input Details	Expected Outcome	Status
TC-001	User Registration - Valid Data	Username: validUser Email: user@example.com Password: StrongPass123!	Registration successful; user status set to "pending"	Pass/Fail
TC-002	User Registration - Invalid Email Format	Username: validUser Email: user@.com Password: StrongPass123!	Registration failed; display error message for invalid email	Pass/Fail
TC-003	User Registration - Password Too Weak	Username: validUser Email: user@example.com Password: pass	Registration failed; display error message for weak password	Pass/Fail
TC-004	User Registration - Missing Required Fields	Username: <empty> Email: user@example.com Password: StrongPass123!	Registration failed; display error message for required fields	Pass/Fail
TC-005	Admin Activation - Activate Pending User	Username: validUser	User status updated to "active"; notification sent to user	Pass/Fail
TC-006	Admin Activation - Reject Pending User	Username: validUser	User status marked as "rejected"; notification sent to user	Pass/Fail
TC-007	User Login - Valid Credentials	Username: validUser Password: StrongPass123!	Login successful; redirect to user dashboard	Pass/Fail
TC-008	User Login - Invalid Password	Username: validUser Password: WrongPass123!	Login failed; display error message for incorrect credentials	Pass/Fail
TC-009	User Login - Inactive User	Username: inactiveUser Password: StrongPass123!	Login failed; display error message indicating user is inactive	Pass/Fail
TC-010	User Dashboard - Display of Values (Temperature, Pressure, Flow Rate)	After login for active user	Display temperature, pressure, flow rate, etc. in user dashboard	Pass/Fail

CHAPTER – 7

SOFTWARE ENVIRONMENT

7.1 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non- profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

```
$ python
Python 2.4.3 (#1, Nov 11 2010, 13:34:43)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the following text at the Python prompt and press the Enter −

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However in Python version 2.4.3, this produces the following result:

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows

```
$ python test.py
```

This produces the following result

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file

Live Demo

```
#!/usr/bin/python
print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows

```
$ chmod +x test.py     # This is to make file executable
$./test.py
```

This produces the following result

```
Hello, Python!
```

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Reserved Words

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example

```
if True:
   print "True"
else:
```

```
   print "False"
```

However, the following block generates an error if True:

```
print "Answer"
print "True"
else:
print "Answer"
print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python
import sys
try:
   # open file stream
   file = open(file_name, "w")
except IOError:
   print "There was an error writing to", file_name
   sys.exit()
print "Enter '", file_finish,
print "' When finished"
while file_text != file_finish:
   file_text = raw_input("Enter text: ")
   if file_text == file_finish:
      # close the file
      file.close
      break
   file.write(file_text)
   file.write("\n")
file.close()
file_name = raw_input("Enter filename: ")
if len(file_name) == 0:
print "Next time please enter something"
```

```
   sys.exit()
try:
   file = open(file_name, "r")
except IOError:
   print "There was an error reading file"
   sys.exit()
file_text = file.read()
file.close()
print file_text
```

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

```
total = item_one + \
        item_two + \
        item_three
```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

Quotation in Python

Python accepts single ('), double (") and triple ('' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

```
word = 'word'
sentence = "This is a sentence."

paragraph = """This is a paragraph. It is made up of multiple lines and
sentences."""
```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

```
#!/usr/bin/python
```

```
# First comment
print "Hello, Python!" # second comment
```

This produces the following result −

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression −

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows −

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
'''
This is a multiline
comment.
'''
```

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action −

```
#!/usr/bin/python
raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement

starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example −

```
if expression :
   suite
elif expression :
   suite
else :
   suite
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python -h
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-c cmd  : program passed in as string (terminates option list)
-d      : debug output from parser (also PYTHONDEBUG=x)
-E      : ignore environment variables (such as PYTHONPATH)
-h      : print this help message and exit
```

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing −

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value −

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo

```
#!/usr/bin/python
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );

print "tup1[0]: ", tup1[0];
print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result −

```
tup1[0]:  physics
tup2[1:5]:  [2, 3, 4, 5]
```

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Zara
dict['Age']:  7
```

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Alice']: ", dict['Alice']
```

When the above code is executed, it produces the following result −

```
dict['Alice']:
Traceback (most recent call last):
  File "test.py", line 4, in <module>
    print "dict['Alice']: ", dict['Alice'];
KeyError: 'Alice'
```

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School"; # Add new entry
print "dict['Age']: ", dict['Age']
```

```
print "dict['School']: ", dict['School']
```

When the above code is executed, it produces the following result −

```
dict['Age']:  8
dict['School']:  DPS School
```

Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
del dict['Name']; # remove entry with key 'Name'
dict.clear();     # remove all entries in dict
del dict ;        # delete entire dictionary
print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more −

```
dict['Age']:
Traceback (most recent call last):
  File "test.py", line 8, in <module>
    print "dict['Age']: ", dict['Age'];
TypeError: 'type' object is unsubscriptable
```

Note − del() method is discussed in subsequent section.

Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys –

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example −

Live Demo

```
#!/usr/bin/python
dict = {['Name']: 'Zara', 'Age': 7}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
    dict = {['Name']: 'Zara', 'Age': 7};
TypeError: unhashable type: 'list'
```

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates −

Live Demo

```
#!/usr/bin/python
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');
# Following action is not valid for tuples
# tup1[0] = 100;
# So let's create a new tuple as follows
tup3 = tup1 + tup2;
print tup3;
```

When the above code is executed, it produces the following result −

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with

putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example −

Live Demo

```
#!/usr/bin/python
tup = ('physics', 'chemistry', 1997, 2000);
print tup;
del tup;
print "After deleting tup : ";
print tup;
```

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist any more −

```
('physics', 'chemistry', 1997, 2000)
After deleting tup :
Traceback (most recent call last):
  File "test.py", line 9, in <module>
    print tup;
NameError: name 'tup' is not defined
```

7.2 DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

Django is a high-level Python web framework that promotes rapid development and clean design. It follows the Model-Template-View (MTV) architecture and includes powerful built-in features like an admin interface, authentication system, and URL routing. Django emphasizes security, scalability, and reusability, making it ideal for building complex, database-driven websites. With an active community and a rich ecosystem of third-party packages, Django supports tasks like API development, asynchronous processing, and content

management. Its ORM allows seamless database interaction using Python.

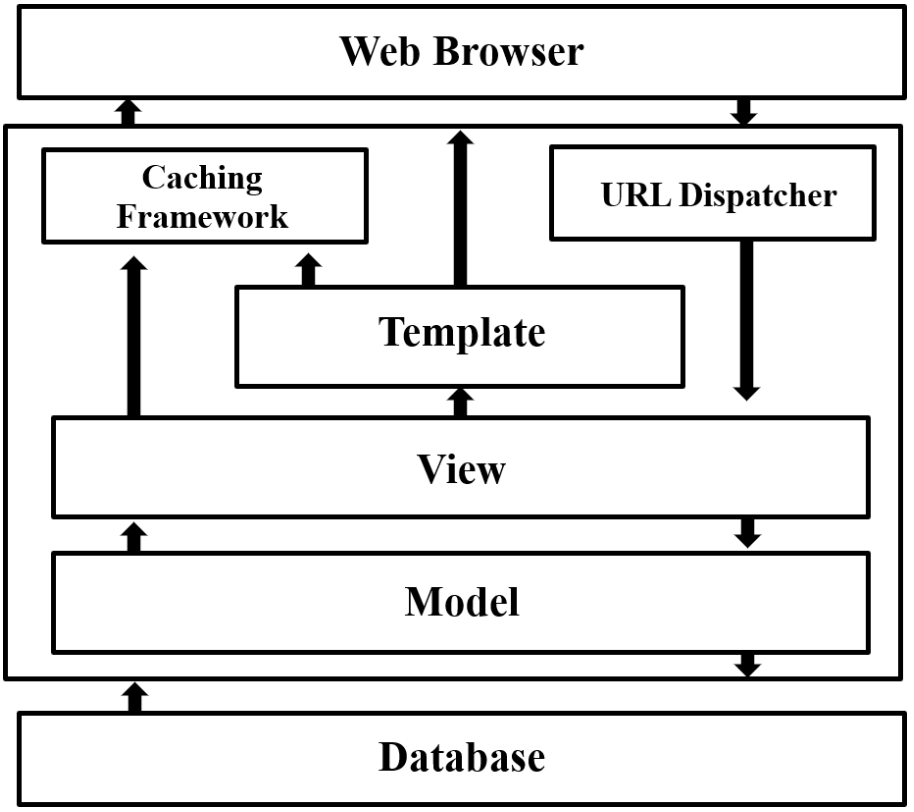


Fig 7.1 Django Framework

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models

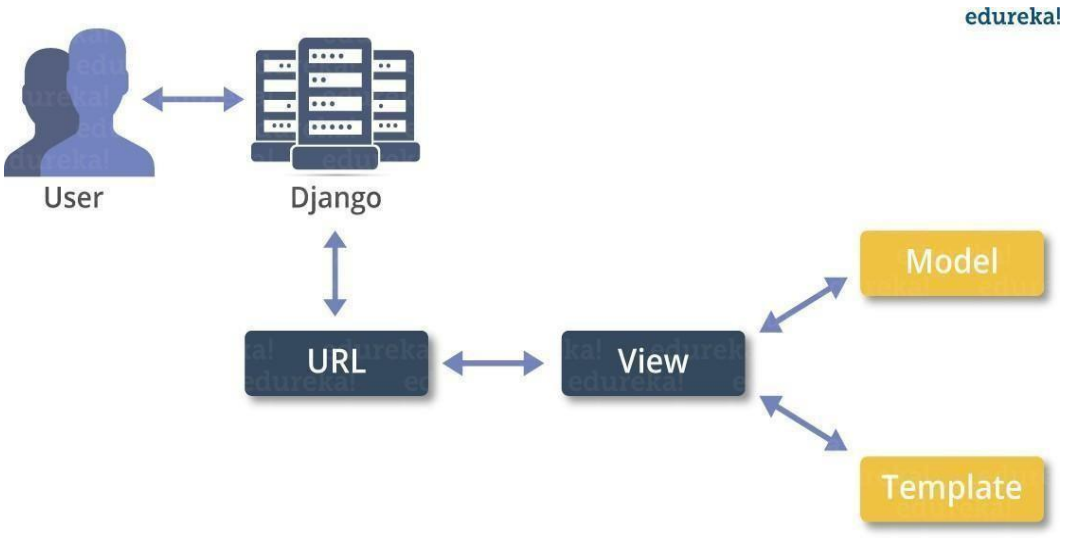


Fig 7.2 Overview of Django

Create a Project

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code –

$ django-admin startproject myproject

This will create a "myproject" folder with the following structure −

myproject/
manage.py
myproject/
init.py
settings.py
urls.py
wsgi.py

The Project Structure

The “myproject” folder is just your project container, it actually contains two elements −

manage.py − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code −

$ python manage.py help

The “myproject” subfolder − This folder is the actual python package of your project. It contains four files −

__init__.py − Just for python, treat this folder as package.

settings.py − As the name indicates, your project settings.

urls.py − All links of your project and the function to call. A kind of ToC of your project.

wsgi.py − If you need to deploy your project over WSGI.

Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set −

DEBUG = True

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to ‘True’ for a live project. However, this has to be set to ‘True’ if you want the Django light server to serve static files. Do it only in the development mode.

```
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': 'database.sql',
    'USER': '',
    'PASSWORD': '',
    'HOST': '',
    'PORT': '',
  }
}
```

Database is set in the ‘Database’ dictionary. The example above is for SQLite engine. As stated earlier, Django also supports −

MySQL (django.db.backends.mysql)
PostGreSQL (django.db.backends.postgresql_psycopg2)
Oracle (django.db.backends.oracle) and NoSQL DB
MongoDB (django_mongodb_engine)
Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME_ZONE, LANGUAGE_CODE, TEMPLATE…

Now that your project is created and configured make sure it's working −

```
$ python manage.py runserver
```

You will get something like the following on running the above code −

```
Validating models...

0 errors found
```

September 03, 2015 - 11:41:50

Django version 1.6.11, using settings 'myproject.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CONTROL-C.

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

Create an Application

We assume you are in your project folder. In our main “myproject” folder, the same folder then manage.py −

$ python manage.py startapp myapp

You just created myapp application and like project, Django create a “myapp” folder with the application structure −

myapp/

  __init__.py

  admin.py

  models.py

  tests.py

  views.py

__init__.py − Just to make sure python handles this folder as a package.

admin.py − This file helps you make the app modifiable in the admin interface.

models.py − This is where all the application models are stored.

tests.py − This is where your unit tests are.

views.py − This is where your application views are.

Get the Project to Know About Your Application

At this stage we have our "myapp" application, now we need to register it with our Django project "myproject". To do so, update INSTALLED_APPS tuple in the settings.py file of your project (add your app name) −

INSTALLED_APPS = (

  'django.contrib.admin',

```
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'myapp',
)
```

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a forms.py file in myapp folder to contain our app forms. We will create a login form.

myapp/forms.py

```
#-*- coding: utf-8 -*-
from django import forms

class LoginForm(forms.Form):
    user = forms.CharField(max_length = 100)
    password = forms.CharField(widget = forms.PasswordInput())
```

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: DateInput for dates, CheckboxInput for checkboxes, etc.

Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the request.POST dictionary.

Let's create a login view in our myapp/views.py −

```
#-*- coding: utf-8 -*-
from myapp.forms import LoginForm

def login(request):
    username = "not logged in"
    if request.method == "POST":
        #Get the posted form
```

```
    MyLoginForm = LoginForm(request.POST)


    if MyLoginForm.is_valid():
      username = MyLoginForm.cleaned_data['username']
  else:
    MyLoginForm = Loginform()
  return render(request, 'loggedin.html', {"username" : username})
```

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.

```
<html>
  <body>
    <form name = "form" action = "{% url "myapp.views.login" %}"
      method = "POST" >{% csrf_token %}
      <div style = "max-width:470px;">
        <center>
          <input type = "text" style = "margin-left:20%;"
            placeholder = "Identifiant" name = "username" />
        </center>
      </div>
      <br>

      <div style = "max-width:470px;">
        <center>
          <input type = "password" style = "margin-left:20%;"
            placeholder = "password" name = "password" />
        </center>
      </div>
      <br>

      <div style = "max-width:470px;">
        <center>

          <button style = "border:0px; background-color:#4285F4; margin-top:8%;
            height:35px; width:80%;margin-left:19%;" type = "submit"
            value = "Login" >
```

```
            <strong>Login</strong>
          </button>
        </center>
      </div>
    </form>
  </body>
</html>
```

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

{% csrf_token %}

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

```
<html>
  <body>
    You are : <strong>{{username}}</strong>
  </body>
</html>
```

Now, we just need our pair of URLs to get started: myapp/urls.py

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',
  url(r'^connection/',TemplateView.as_view(template_name = 'login.html')),
  url(r'^login/', 'login', name = 'login'))
```

When accessing "/myapp/connection", we will get the following login.html template rendered –

Setting Up Sessions

In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE_CLASSES and the INSTALLED_APPS options. This should be done while creating the project, but it's always good to know, so MIDDLEWARE_CLASSES should have –

'django.contrib.sessions.middleware.SessionMiddleware'

And INSTALLED_APPS should have −

'django.contrib.sessions'

By default, Django saves session information in database (django_session table or collection), but you can configure the engine to store information using other ways like: in file or in cache.

When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute.

Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won't see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.

For this, first lets change our login view to save our username cookie server side −

```
def login(request):
  username = 'not logged in'
  f request.method == 'POST':
    MyLoginForm = LoginForm(request.POST)

    if MyLoginForm.is_valid():
      username = MyLoginForm.cleaned_data['username']
      request.session['username'] = username
    else:
      MyLoginForm = LoginForm()

  return render(request, 'loggedin.html', {"username" : username}
```

Then let us create formView view for the login form, where we won't display the form if cookie is set −

```
def formView(request):
  if request.session.has_key('username'):
    username = request.session['username']
    return render(request, 'loggedin.html', {"username" : username})
```

```
  else:
    return render(request, 'login.html', {})
```

Now let us change the url.py file to change the url so it pairs with our new view −

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView
urlpatterns = patterns('myapp.views',
  url(r'^connection/','formView', name = 'loginform'),
  url(r'^login/', 'login', name = 'login'))
```

When accessing /myapp/connection, you will get to see the following page

CHAPTER – 8

OUTPUT SCREENSHOTS

8.1 SCREENSHOTS

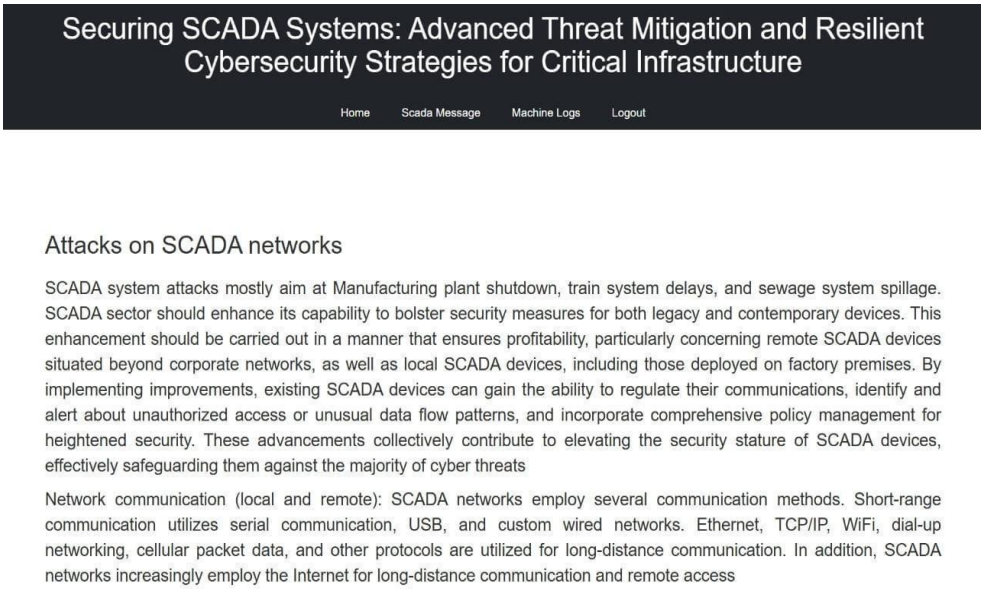


Fig: 8.1 This image appears to be a webpage discussing SCADA system security, focusing on advanced threat mitigation and cyber security strategies for critical infrastructure.

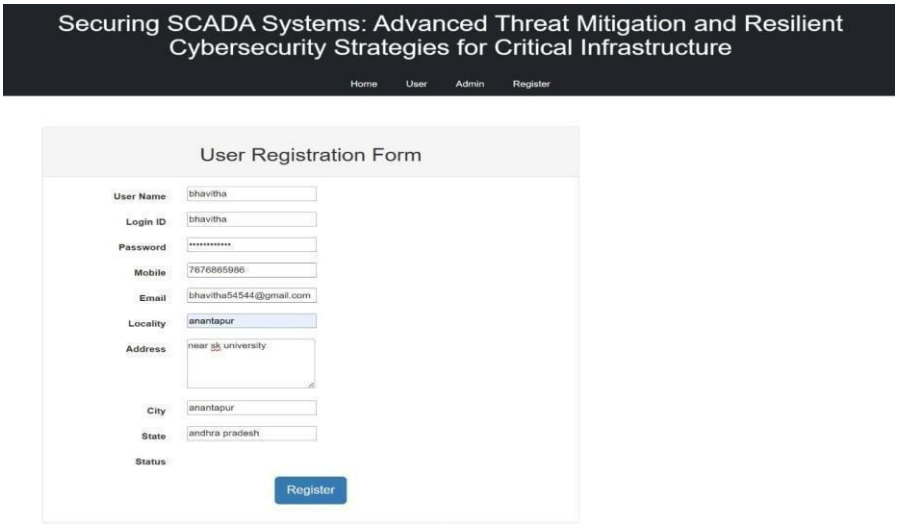


Fig: 8.2 The image shows a user registration form for a system focused on securing SCADA systems for critical infrastructure.

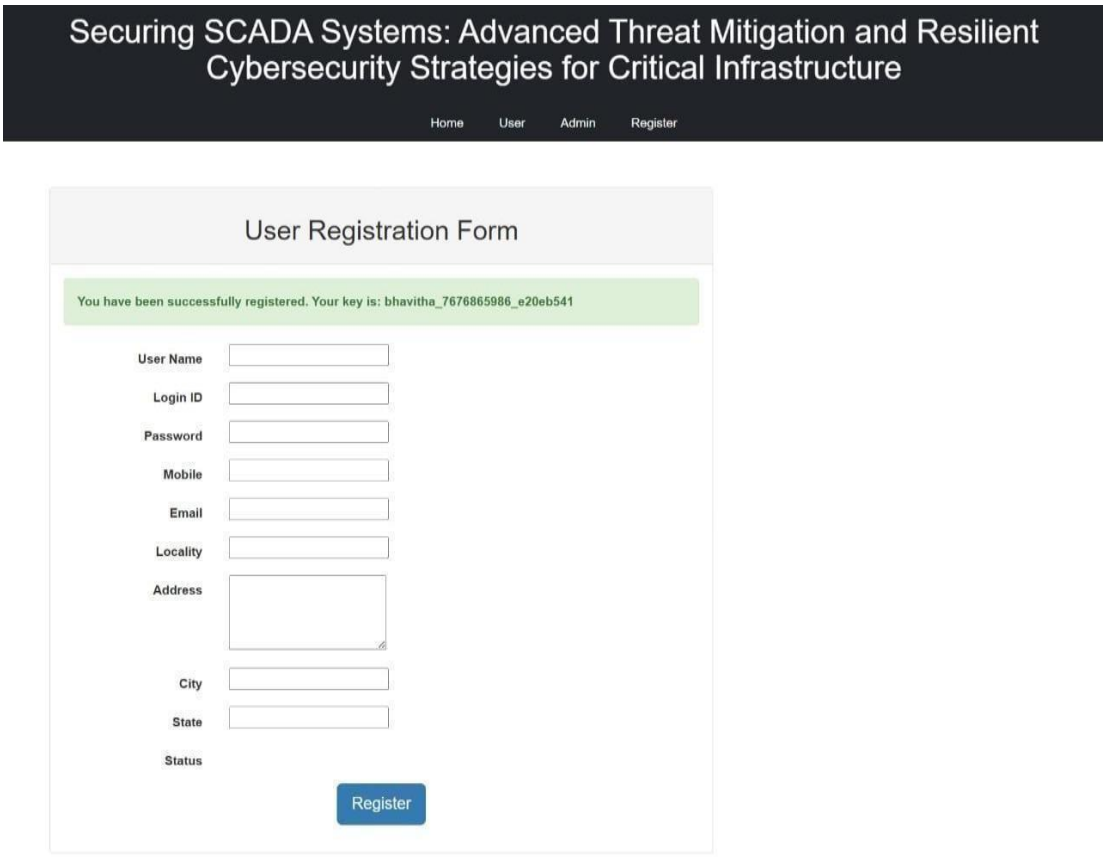


Fig: 8.3 The image displays a user registration form with a success message indicating a user has been registered and provided with a key.

Fig: 8.4 The image shows an admin login form for a system focused on securing SCADA systems for critical infrastructure.

Fig: 8.5 The webpage discusses attacks on SCADA networks and the importance of enhancing security measures for both local and remote devices, along with various network communication methods used.

Fig: 8.6 This page displays a list of registered users for a SCADA cyber security system, showing their details and account activation status.

Fig: 8.7 This page displays a list of registered users for a SCADA cybersecurity system, allowing administrators to activate user accounts.

Fig: 8.8 This is a user login page, The user can enter the password and then login into the page.

Fig: 8.9 This is a user login page where admin not activated the account of the user.

Fig: 8.10 The image shows a when user forgets password, they can reset the password by entering username.

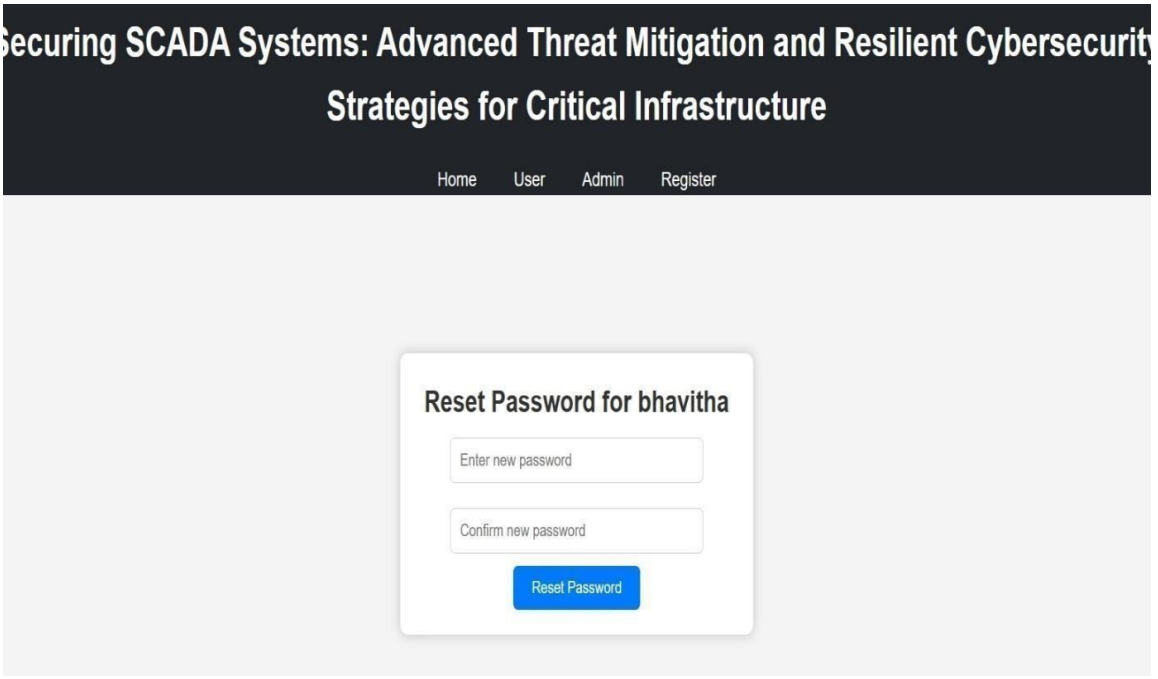


Fig: 8.11 The image shows a User can Reset the Password by entering new password

Fig: 8.12 The image shows after resetting password the password got successfully updated.

Fig: 8.13 This image displays a key-based login page for a SCADA system security platform, featuring authentication via an access key.

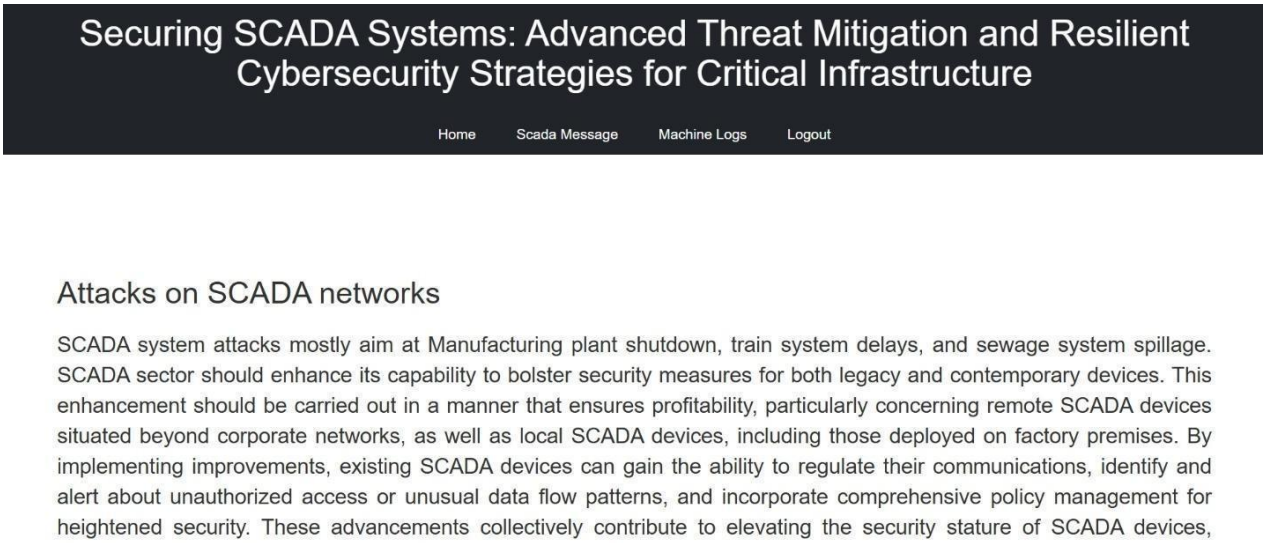


Fig: 8.14 The User Home Page after login through Password and Key.

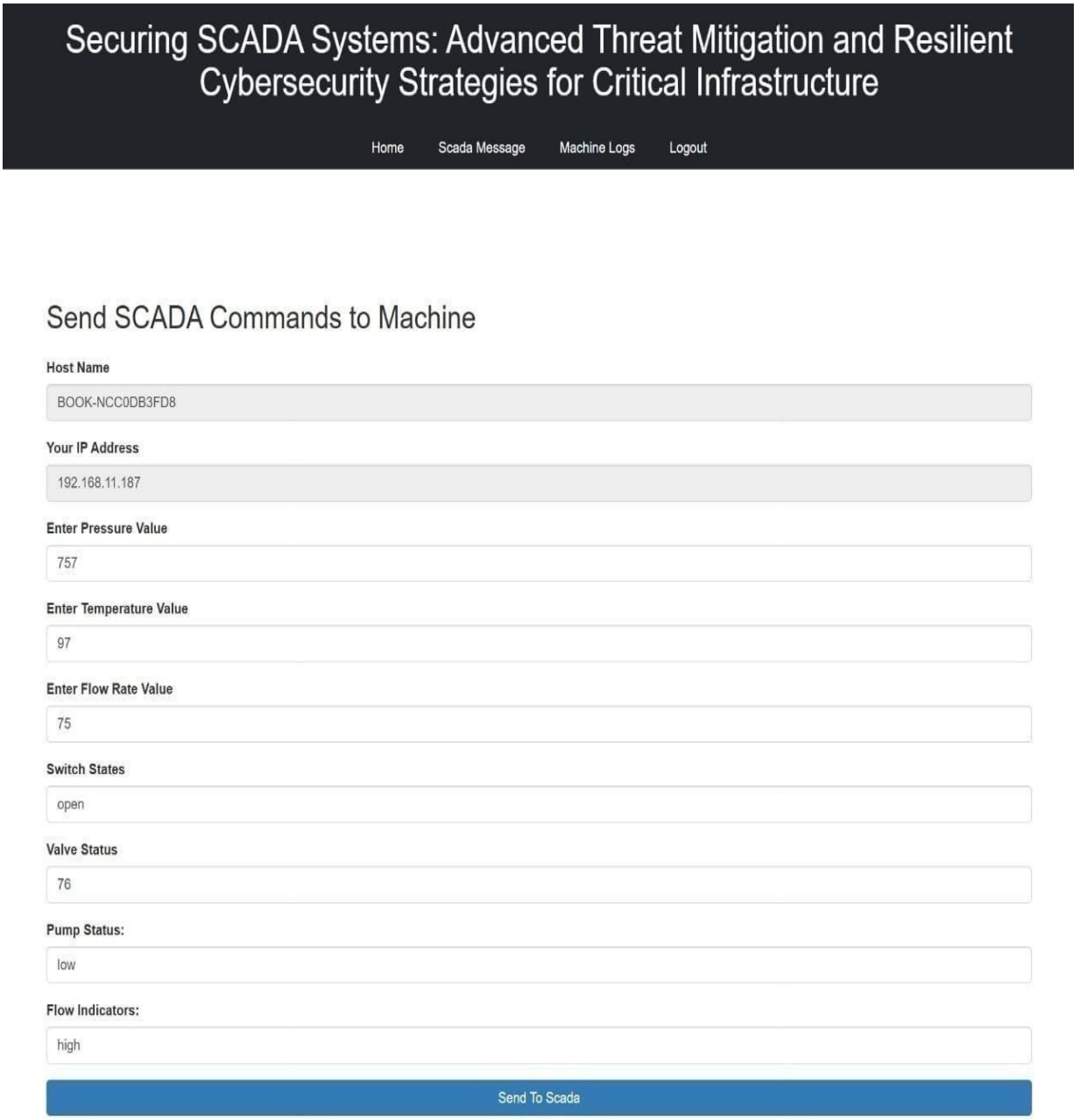
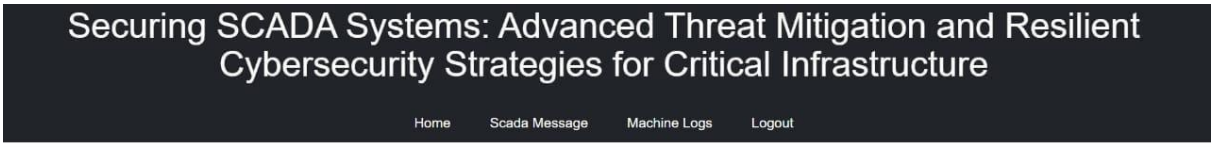


Fig: 8.15 This image shows a SCADA system interface for sending machine commands, allowing users to input parameters like pressure, temperature, and flow rate.

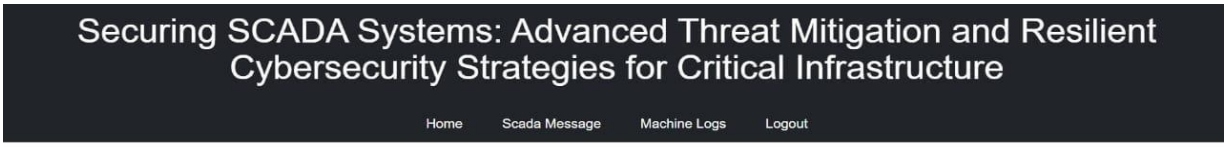


Attacks on SCADA networks

HostName	BOOK-NCC0DB3FD8
IP	192.168.11.187
PressureValue	757
Temperature	97
FlowRate	75
SwitchRate	open
ValveStatus	76
PumpStatus	low
FlowIndicator	high
status	Idle

Network communication (local and remote): SCADA networks employ several communication methods. Short-range communication utilizes serial communication, USB, and custom wired networks. Ethernet, TCP/IP, WiFi, dial-up networking, cellular packet data, and other protocols are utilized for long-distance communication. In addition, SCADA networks increasingly employ the Internet for long-distance communication and remote access

Fig: 8.16 The image shows a table displaying details of a potential attack on a SCADA network, including the host's IP address, sensor readings, and status



Scada Machine Sent Logs

HostName	IP	PressureValue	Temperature	FlowRate	SwitchRate	ValveStatus	PumpStatus	FlowIndicator	status
DESKTOP-SUHNF4D	192.168.0.10	High	Low	Minimum	on	on	Idle	Good	tampered
DESKTOP-SUHNF4D	192.168.0.10	290	80	89	On	89	High	Low	tampered
DESKTOP-SUHNF4D	192.168.0.10	890	879	87	88	98	8	8	Idle
DESKTOP-SUHNF4D	192.168.0.10	High	879	Minimum	88	89	98	Good	tampered
DESKTOP-SUHNF4D	192.168.0.10	31 bar	Low	2355 psi	On	high	High	Low	tampered
DESKTOP-SUHNF4D	192.168.0.10	890	Low	Minimum	On	98	High	Low	Idle
DESKTOP-865NABJ	192.168.1.30	76 bar	44	2779 psi	open	open	good	55	tampered
DESKTOP-A0LRJP2	192.168.90.83	69 bar	10	2332 psi	15	close	0	12	tampered
BOOK-NCC0DB3FD8	192.168.11.187	757	97	75	open	76	low	high	Idle

Network communication (local and remote): SCADA networks employ several communication methods. Short-range communication utilizes serial communication, USB, and custom wired networks. Ethernet, TCP/IP, WiFi, dial-up networking, cellular packet data, and other protocols are utilized for long-distance communication. In addition, SCADA networks increasingly employ the Internet for long-distance communication and remote access

Fig: 8.17 The image displays a log of SCADA machine data, potentially indicating system monitoring and security status, with a title suggesting a focus on securing such systems.

CHAPTER – 9

CONCLUSION AND FUTURE ENHANCEMENT

9.1 CONCLUSION

The security of Supervisory Control and Data Acquisition (SCADA) systems in critical infrastructures is of paramount importance in today's digital age, where cyber threats loom large and the integrity of processes and operations is crucial. This comprehensive analysis delves into the vulnerabilities inherent in SCADA systems, particularly concerning outdated protocols and insufficient security measures. The authors aptly highlight the pressing need to enhance the security posture of SCADA devices to thwart potential cyber-terrorist assaults and ensure the resilience of critical infrastructures.

The evolution of SCADA architectures, from monolithic systems to cloud-based connectivity, underscores the dynamic nature of technological advancements. However, this evolution also introduces new challenges, particularly in safeguarding against cyberattacks targeting SCADA-based critical infrastructure. With the rise of sophisticated cyber threats, including phishing, DDoS attacks, and man-in-the middle attacks, there is an urgent call for robust security solutions tailored to the unique requirements of SCADA systems. The implementation of adaptable security strategies, such as the utilization of SCADA firewalls employing virtual isolated networks, holds promise in mitigating cyber risks and bolstering the defense mechanisms of SCADA devices.

Additionally, the adoption of secure communication protocols and stringent authentication mechanisms are imperative to safeguard against potential intrusions and data breaches. As we navigate the complex landscape of SCADA security, collaboration between industry stakeholders, government agencies, and cybersecurity experts is essential to develop effective countermeasures and resilience strategies. By prioritizing security enhancements and embracing innovative technologies, we can fortify SCADA systems against emerging threats and uphold the integrity of critical infrastructures in the face of evolving cyber challenges. In conclusion, this study underscores the critical importance of enhancing SCADA systems' security and resilience in safeguarding critical infrastructures against cyber threats, thereby ensuring societal stability, economic prosperity, and national security in an increasingly interconnected world.

9.2 FUTURE ENHANCEMENT

Future enhancements for the proposed SCADA system can focus on several key areas to improve its security, efficiency, and adaptability to evolving technological landscapes. Here are some potential enhancements:

1. Integration of Artificial Intelligence (AI) and Machine Learning (ML): Implementing AI and ML algorithms can enhance threat detection capabilities by analyzing patterns in network traffic and identifying anomalies indicative of cyber threats.

2. Advanced Data Analytics: Incorporating advanced data analytics tools can facilitate better decision-making by providing deeper insights into operational data. This can include real-time data visualization, trend analysis, and reporting capabilities that allow operators to quickly assess system performance and respond to issues.

3. Cloud Integration: Expanding the system's architecture to leverage cloud computing can provide scalable resources for data storage and processing. This can enable better data management, enhance computational power, and facilitate remote access and monitoring while ensuring robust security measures are in place.

4. IoT Device Integration: Future enhancements could include the integration of Internet of Things (IoT) devices for enhanced monitoring and control. This would involve implementing secure communication protocols for IoT devices, allowing them to contribute real-time data to the SCADA system and improving operational visibility.

5. Cyber security Training and Awareness Programs: Developing ongoing training programs for staff and operators can enhance the overall security posture of the SCADA system. By raising awareness about cybersecurity threats and best practices.

6. Blockchain Technology for Data Integrity: Exploring the use of blockchain technology can enhance data integrity and security within the SCADA system. By creating a decentralized ledger of transactions and communications.

7. Enhanced Incident Response Plans: Developing and regularly updating comprehensive incident response plans can ensure that organizations are prepared to respond to security breaches effectively.

8. Regulatory Compliance and Standards Alignment: Ensuring that the SCADA system remains compliant with evolving industry standards and regulations will be essential. Future enhancements should include regular assessments and updates to align with best practices in cybersecurity and operational technology (OT) security frameworks.

REFERENCES

1. H. Altaleb and Z. Rajnai, “Risk assessments Methods and Cyber Vulnerabilities in SCADA systems,” Natl. Secur. Rev. Period. Mil. Natl. Secur. Serv., vol. 2, pp. 181–194, 2021.

2. J. Jaskolka and J. Villasenor, “An approach for identifying and analyzing implicit interactions in distributed systems,” IEEE Trans. Reliab., vol. 66, no. 2, pp. 529–546, Jun. 2017, doi: 10.1109/TR.2017.2665164.

3. K. Stouffer, J. Falco, and K. Scarfone, “GUIDE to industrial control systems (ICS) security,” Stuxnet Comput. Worm Ind. Control Syst. Secur., pp. 11–158, 2011.

4. K. Sayed and H. A. Gabbar, “Scada and smart energy grid control automation,” Smart Energy Grid Eng., pp. 481–514, 2017, doi: 10.1016/B978-0-12-805343-0.00018-8.

5. S. Cunningham, “Cyber security for industrial control systems,” Power Eng. (Barrington, Illinois), vol. 115, no. 11, pp. 142–146, 2011, doi: 10.2524/jtappij.69.1205.

6. V. M. Igure, S. A. Laughter, and R. D. Williams, “Security issues in SCADA networks,” Comput. Secur., vol. 25, no. 7, pp. 498 506, 2006, doi: 10.1016/j.cose.2006.03.001.

7. J. Hajda, R. Jakuszewski, and S. Ogonowski, “Security challenges in industry 4.0 plc systems,” Appl. Sci., vol. 11, no. 21, 2021, doi: 10.3390/app11219785.

8. T. Sauter and C. Schwaiger, “Achievement of secure Internet access to fieldbus systems,” Microprocess. Microsyst., vol. 26, no. 7, pp. 331–339, Sep. 2002, doi: 10.1016/S0141

   9331(02)00044-3.

9. V. M. Igure, S. A. Laughter, and R. D. Williams, “Security issues in SCADA networks,” Comput. Secur., vol. 25, no. 7, pp. 498 506, Oct. 2006, doi: 10.1016/J.COSE.2006.03.001.

10. D. Ranathunga, M. Roughan, H. Nguyen, P. Kernick, and N. Falkner, “Case Studies of SCADA Firewall Configurations and the Implications for Best Practices,” IEEE Trans. Netw. Serv. Manag., vol. 13, no. 4, pp. 871–884, 2016, doi: 10.1109/TNSM.2016.2597245.

VELAMMAL
INSTITUTE OF TECHNOLOGY
Chennai - Kolkatta Highway, Panchetti, Ponneri

5th International Conference on Artificial Intelligence,
6G Communications and Network Technologies - ICA6NT 2025

Certificate of Appreciation

This is to certify that

GANGULA BHAVITHA REDDY

Anantha lakshmi institute of technology and sciences

has participated and presented paper entitled

Securing SCADA Systems: Advanced Threat Mitigation and Resilient Cybersecurity Strategies for Critical Infrastructure

at the **5th International Conference on Artificial Intelligence, 6G Communications and Network Technologies (ICA6NT 2025)** organized by the **Department of Electronics and Communication Engineering, Velammal Institute of Technology,** Chennai held on **27th & 28th , March 2025.**

Coordinator	Coordinator	Coordinator	Convener	Conference Chair
Dr.R.Jothi Chitra	Dr. M. Sivarathinabala	Mr. K. Ragupathi	Dr. B. Sridevi	Dr. N. Balaji
Professor-ECE	Professor-ECE	Assistant Professor-ECE	Professor & Head -ECE	Principal

VELAMMAL
INSTITUTE OF TECHNOLOGY
Chennai - Kolkatta Highway, Panchetti, Ponneri

5th International Conference on Artificial Intelligence,
6G Communications and Network Technologies - ICA6NT 2025

Certificate of Appreciation

This is to certify that

UPPARA BHAVITHA

Anantha lakshmi institute of technology and sciences

has participated and presented paper entitled

Securing SCADA Systems: Advanced Threat Mitigation and Resilient Cybersecurity Strategies for Critical Infrastructure

at the **5th International Conference on Artificial Intelligence, 6G Communications and Network Technologies (ICA6NT 2025)** organized by the **Department of Electronics and Communication Engineering, Velammal Institute of Technology,** Chennai held on **27th & 28th , March 2025.**

Coordinator	Coordinator	Coordinator	Convener	Conference Chair
Dr.R.Jothi Chitra	Dr. M. Sivarathinabala	Mr. K. Ragupathi	Dr. B. Sridevi	Dr. N. Balaji
Professor-ECE	Professor-ECE	Assistant Professor-ECE	Professor & Head -ECE	Principal

VELAMMAL
INSTITUTE OF TECHNOLOGY
Chennai - Kolkatta Highway, Panchetti, Ponneri

5th International Conference on Artificial Intelligence,

6G Communications and Network Technologies - ICA6NT 2025

Certificate of Appreciation

This is to certify that

VIRUPAKSHI HAMPI AMARESH

Anantha lakshmi institute of technology and sciences

has participated and presented paper entitled

Securing SCADA Systems: Advanced Threat Mitigation and Resilient Cybersecurity Strategies for Critical Infrastructure

at the **5th International Conference on Artificial Intelligence, 6G Communications and Network Technologies (ICA6NT 2025)** organized by the **Department of Electronics and Communication Engineering, Velammal Institute of Technology,** Chennai held on **27th & 28th , March 2025.**

Coordinator	Coordinator	Coordinator	Convener	Conference Chair
Dr.R.Jothi Chitra	Dr. M. Sivarathinabala	Mr. K. Ragupathi	Dr. B. Sridevi	Dr. N. Balaji
Professor-ECE	Professor-ECE	Assistant Professor-ECE	Professor & Head -ECE	Principal

VELAMMAL
INSTITUTE OF TECHNOLOGY
Chennai - Kolkatta Highway, Panchetti, Ponneri

5th International Conference on Artificial Intelligence,
6G Communications and Network Technologies - ICA6NT 2025

Certificate of Appreciation

This is to certify that

VADDE SUDHA RANI

Anantha lakshmi institute of technology and sciences

has participated and presented paper entitled

Securing SCADA Systems: Advanced Threat Mitigation and Resilient Cybersecurity Strategies for Critical Infrastructure

at the 5th International Conference on Artificial Intelligence, 6G Communications and Network Technologies (ICA6NT 2025) organized by the Department of Electronics and Communication Engineering, Velammal Institute of Technology, Chennai held on 27th & 28th , March 2025.

Coordinator	Coordinator	Coordinator	Convener	Conference Chair
Dr.R.Jothi Chitra	Dr. M. Sivarathinabala	Mr. K. Ragupathi	Dr. B. Sridevi	Dr. N. Balaji
Professor-ECE	Professor-ECE	Assistant Professor-ECE	Professor & Head -ECE	Principal

VELAMMAL
INSTITUTE OF TECHNOLOGY
Chennai - Kolkatta Highway, Panchetti, Ponneri

IEEE VELAMMALITECH STUDENT BRANCH
IEEE ComSoc IEEE Communications Society

5th International Conference on Artificial Intelligence,

6G Communications and Network Technologies - ICA6NT 2025

Certificate of Appreciation

This is to certify that

CHITYALYA SUMANTH REDDY

Anantha lakshmi institute of technology and sciences

has participated and presented paper entitled

Securing SCADA Systems: Advanced Threat Mitigation and Resilient Cybersecurity Strategies for Critical Infrastructure

at the **5th International Conference on Artificial Intelligence, 6G Communications and Network Technologies (ICA6NT 2025)** organized by the **Department of Electronics and Communication Engineering, Velammal Institute of Technology,** Chennai held on **27th & 28th , March 2025.**

Coordinator	Coordinator	Coordinator	Convener	Conference Chair
Dr.R.Jothi Chitra	Dr. M. Sivarathinabala	Mr. K. Ragupathi	Dr. B. Sridevi	Dr. N. Balaji
Professor-ECE	Professor-ECE	Assistant Professor-ECE	Professor & Head -ECE	Principal