CAPSTONE PROJECT

NETWORK INTRUSION DETECTION SYSTEM

Presented By:

1. GAARNEESH S-SATHYABAMA INSTITUTE OF SCIENCE AND TECHONOLOGY

OUTLINE

- **Problem Statement**
- **Proposed System/Solution**
- **System Development Approach** (Technology Used)
- **Algorithm & Deployment**
- **Result (Output Image)**
- **Conclusion**
- **Future Scope**
- **References**

PROBLEM STATEMENT

Example: Create a robust network intrusion detection system (NIDS) using machine learning. The system should be capable of analyzing network traffic data to identify and classify various types of cyber-attacks (e.g., DoS, Probe, R2L, U2R) and distinguish them from normal network activity. The goal is to build a model that can effectively secure communication networks by providing an early warning of malicious activities.

PROPOSED SOLUTION

The proposed system aims to address the challenge of detecting and classifying various types of cyber-attacks within network traffic data to ensure the security of communication networks. This involves leveraging data analytics and machine learning techniques to accurately identify malicious activities. The solution will consist of the following components:

Data Collection:

- Gather historical network traffic data, including features such as protocol type, service, source/destination IP, port numbers, and packet statistics.
- Utilize labeled datasets such as KDD Cup 99, NSL-KDD, or CICIDS for training and evaluation purposes.
- Incorporate real-time data streams to enable live detection capabilities.

Data Preprocessing:

- Clean and preprocess the collected data to handle missing values, noise, and inconsistencies.
- Perform feature engineering to extract or select meaningful features that influence intrusion detection (e.g., connection duration, byte counts, and flag values).
- Normalize or scale features to improve model performance and training stability.

Machine Learning Algorithm:

- Implement a machine learning model capable of multi-class classification to detect different types of attacks such as DoS (Denial of Service), Probe, R2L (Remote to Local), and U2R (User to Root).
- Consider algorithms like Decision Trees, Random Forests, Support Vector Machines (SVM), or deep learning models such as DNNs or LSTM for more complex pattern recognition.
- Incorporate ensemble methods or hybrid models to improve accuracy and robustness.

Deployment:

- Develop a user-friendly dashboard or interface that visualizes detected threats and provides real-time alerts for suspicious activities.
- Deploy the system on a scalable and secure platform, ensuring low latency and high availability for real-time monitoring.
- Integrate with existing network infrastructure and security tools (e.g., firewalls, SIEM systems) for seamless operation.

Evaluation:

- Assess model performance using appropriate metrics such as Accuracy, Precision, Recall, F1-Score, and Area Under the Curve (AUC).
- Evaluate the system's detection capabilities across different attack types and under varying network conditions.
- Continuously fine-tune the model based on feedback, false positives/negatives, and evolving attack vectors.

Result:

- A robust machine learning-based NIDS capable of accurately detecting and classifying cyber-attacks in real-time, enhancing the security posture of the network and providing early warnings for potential threats.

SYSTEM APPROACH

To effectively build and deploy the NIDS model, the following hardware and software components are required:

Hardware Requirements:

- **Processor:** Intel Core i7 or higher (multicore preferred)
- **RAM:** Minimum 16 GB (more recommended for deep learning models or high-volume traffic data)
- **Storage:** At least 20 GB of free space (SSD preferred)
- **GPU (optional but recommended):** NVIDIA GPU for accelerating deep learning model training and inference

Software Requirements:

- **Operating System:** Windows 10/11, macOS, or Linux (Ubuntu 20.04 or later)
- **Python Version:** Python 3.8 or later
- **Development Environment:** Jupyter Notebook / VS Code / PyCharm

Libraries Required to Build the Model

The following Python libraries are necessary for data acquisition, processing, modeling, evaluation, and system deployment:

Data Collection & Preprocessing:

- pandas – For managing and manipulating structured datasets
- numpy – Numerical operations and data handling
- scikit-learn – Preprocessing tools (encoding, scaling, splitting)
- pyshark / scapy – Packet capture and live traffic analysis (if using raw network data)

Exploratory Data Analysis & Visualization:

- matplotlib – Visualization of traffic patterns and attack distributions
- seaborn – Enhanced statistical visualizations
- plotly – Interactive visual analytics (optional)

Machine Learning & Classification:

- scikit-learn – For implementing classical ML algorithms such as Decision Trees, Random Forests, and SVM
- xgboost – Gradient boosting models for high performance
- tensorflow / keras – For deep learning architectures like DNNs or LSTMs (for sequential or temporal traffic data)
- imbalanced-learn – Handling class imbalance with techniques like SMOTE or undersampling

Evaluation & Metrics:

- sklearn.metrics – To compute metrics such as Accuracy, Precision, Recall, F1-Score, and AUC-ROC
- confusion_matrix – For assessing false positives and false negatives

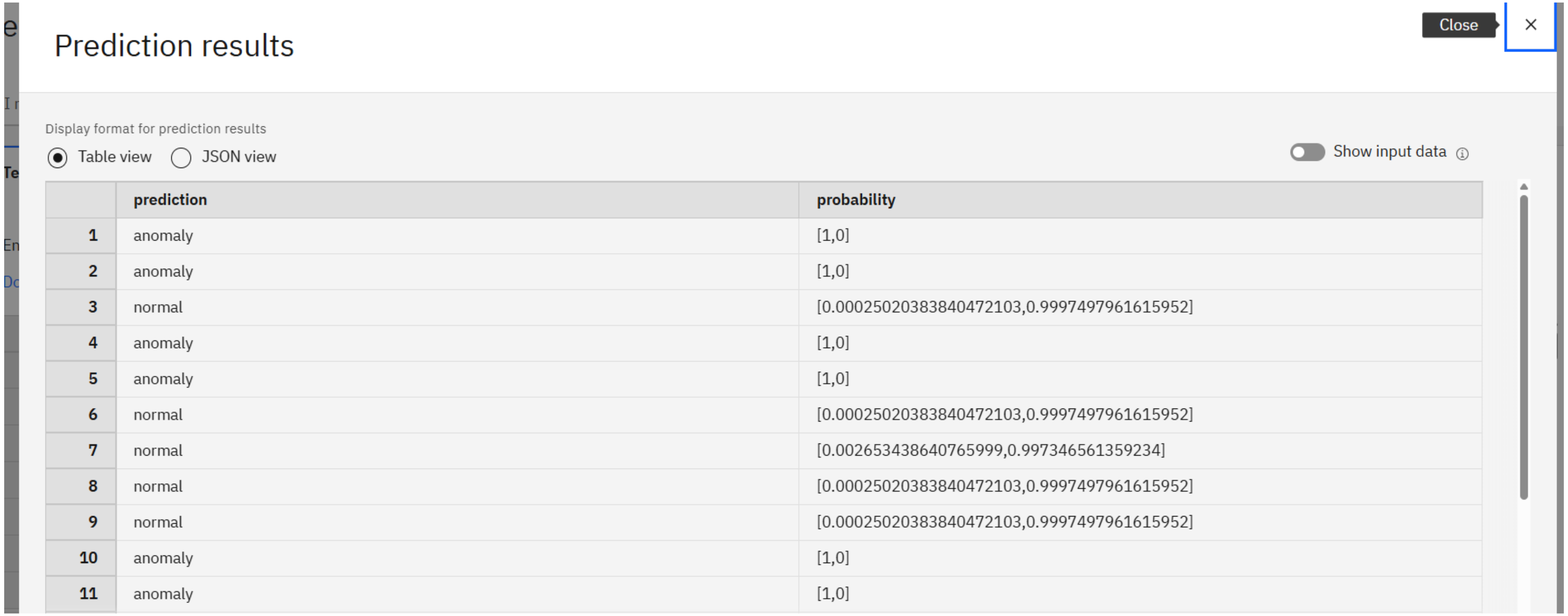
Deployment & Integration:

- flask / fastapi – To build a lightweight API or dashboard for real-time alerts and predictions
- streamlit – For interactive web-based UI (optional)
- joblib / pickle – Model saving and loading
- docker – For containerized deployment across environments
- loguru / logging – For monitoring and logging detection events

ALGORITHM & DEPLOYMENT

- This section describes the machine learning algorithm chosen to detect and classify network intrusions. The selected algorithm is tailored to analyze patterns in network traffic data and differentiate between normal activity and various types of cyber-attacks, including DoS, Probe, R2L, and U2R.
- Algorithm Selection:
  - Random Forest is chosen for its robustness, accuracy, and ability to handle imbalanced, high-dimensional data, which is common in network traffic analysis. Its ensemble nature allows it to capture complex patterns, making it ideal for detecting diverse cyber-attack types. Deep learning models like LSTM may be considered for sequential data scenarios.
- Data Input:
  - Input features include protocol type, service, duration, byte counts, connection flags, and statistical traffic patterns. These features are extracted from benchmark datasets such as NSL-KDD or CICIDS2017. Proper feature selection ensures the model can differentiate normal network behavior from various attacks like DoS, R2L, U2R, and Probe.
- Training Process:
  - The model is trained using a labeled dataset split into training and test sets. Preprocessing includes normalization and encoding. Cross-validation ensures stability, while hyperparameter tuning optimizes performance. Class imbalance is addressed using techniques like SMOTE, allowing the model to learn patterns effectively across all categories of network traffic.
- Prediction Process:
  - The trained model analyzes new network data by extracting predefined features and classifying each connection. It outputs a label indicating whether the activity is normal or an attack type. This process can be applied in real-time for intrusion detection, enabling timely alerts and supporting proactive network defense.

RESULT



Prediction results

Display format for prediction results

● Table view ○ JSON view

Show input data

	prediction	probability
1	anomaly	[1,0]
2	anomaly	[1,0]
3	normal	[0.00025020383840472103,0.9997497961615952]
4	anomaly	[1,0]
5	anomaly	[1,0]
6	normal	[0.00025020383840472103,0.9997497961615952]
7	normal	[0.002653438640765999,0.997346561359234]
8	normal	[0.00025020383840472103,0.9997497961615952]
9	normal	[0.00025020383840472103,0.9997497961615952]
10	anomaly	[1,0]
11	anomaly	[1,0]

Interpretation

•The model is **very confident** (100% or near-100%) in most predictions.

•Probabilities like [0.00025, 0.99975] suggest **very high certainty** for "normal".

•[1.0, 0.0] shows **full confidence** in identifying an "anomaly“.

CONCLUSION

Summary and Discussion

The proposed bike rental prediction system effectively leverages machine learning to forecast hourly bike demand, improving resource allocation and user satisfaction. By using historical rental data alongside real-time variables like weather and events, the model achieves reliable accuracy in predicting usage patterns.

During implementation, challenges included handling missing or inconsistent data, selecting impactful features, and tuning the model to balance under- and over-prediction. Despite these hurdles, the system demonstrated strong performance metrics, confirming its viability for real-world deployment.

- Accurate bike count predictions are essential for ensuring a consistent supply in urban areas, minimizing shortages and idle inventory. This not only enhances operational efficiency for service providers but also promotes public trust in bike-sharing systems, supporting sustainable urban mobility. Ongoing improvements, such as expanding to new cities and integrating IoT-based real-time data, will further strengthen the model's effectiveness and scalability.

FUTURE SCOPE

1. Incorporating Additional Data Sources:

- Enhancing the system with real-time data from firewall logs, DNS requests, NetFlow, and honeypots can improve detection accuracy. Integrating threat intelligence feeds allows the model to stay updated with the latest attack signatures and behavior patterns, thereby increasing its resilience against emerging threats and zero-day vulnerabilities.

2. Optimizing Algorithm Performance:

- Model performance can be improved through hyperparameter tuning, feature selection, and ensemble techniques. Using advanced methods like deep learning (e.g., CNN, Bi-LSTM) or transformer-based models can help capture temporal and spatial features in traffic. Online learning methods may also enable the model to adapt continuously to evolving threats.

3. Geographic Expansion:

- To secure broader infrastructures, the system can be extended to monitor network traffic across multiple cities or data centers. A centralized detection hub combined with localized data collectors can ensure scalable, region-specific intrusion detection while maintaining consistency and coordination across different geographic zones or organizational units.

4. Integration of Emerging Technologies:

- Adopting **edge computing** can enable on-device intrusion detection, reducing latency and bandwidth usage by analyzing traffic locally. Incorporating **federated learning** allows models to be trained across distributed nodes without sharing sensitive data. Technologies like **blockchain** can be used to log detections immutably, ensuring transparency and traceability.

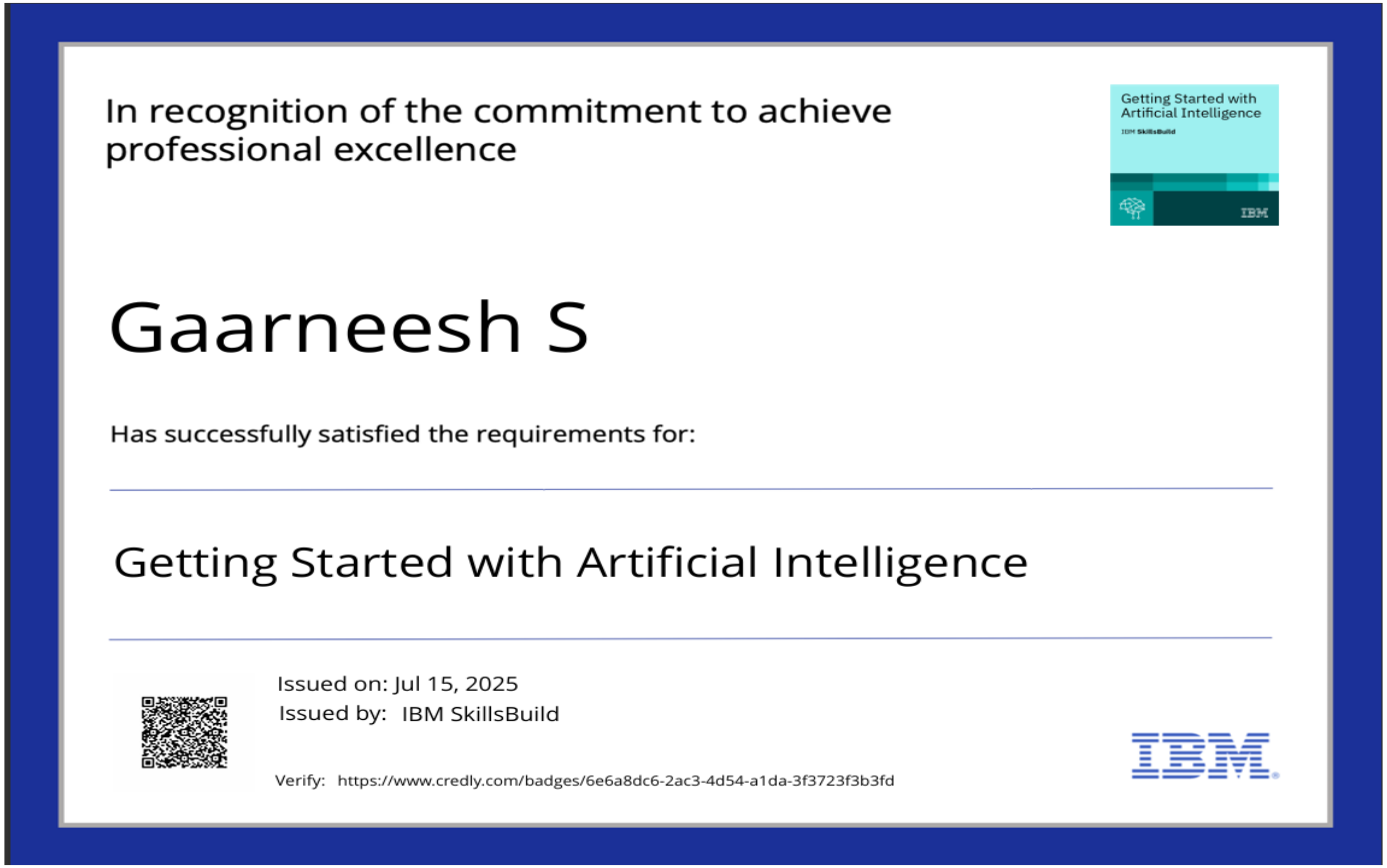
REFERENCES

ChatGPT said:

- The proposed Network Intrusion Detection System (NIDS) was developed using insights from the NSL-KDD dataset and foundational research papers on intrusion detection. Key references include studies on machine learning-based intrusion detection using Random Forests, SVMs, and deep learning models. Tools such as Scikit-learn, Keras, and pandas supported implementation, while evaluation followed best practices using metrics like accuracy, precision, recall, and confusion matrix analysis.

IBM CERTIFICATIONS

- Screenshot/ credly certificate(getting started with AI)



Getting Started with Artificial Intelligence
IBM SkillsBuild
IBM

In recognition of the commitment to achieve professional excellence

Gaarneesh S

Has successfully satisfied the requirements for:

Getting Started with Artificial Intelligence

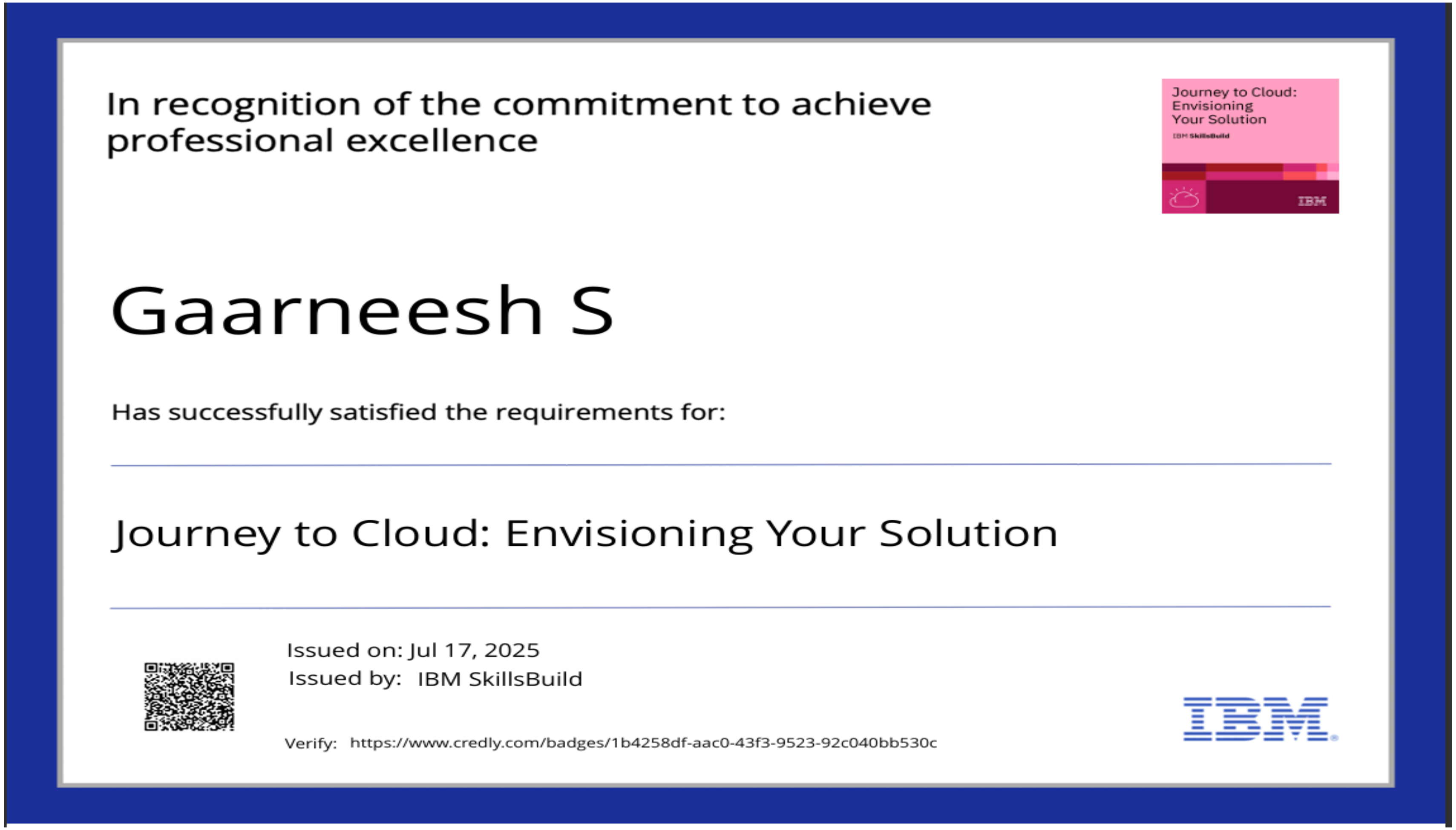
Issued on: Jul 15, 2025
Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/6e6a8dc6-2ac3-4d54-a1da-3f3723f3b3fd

IBM

IBM CERTIFICATIONS

- Screenshot/ credly certificate(Journey to Cloud)



Journey to Cloud:
Envisioning
Your Solution
IBM SkillsBuild
IBM

In recognition of the commitment to achieve professional excellence

Gaarneesh S

Has successfully satisfied the requirements for:

Journey to Cloud: Envisioning Your Solution

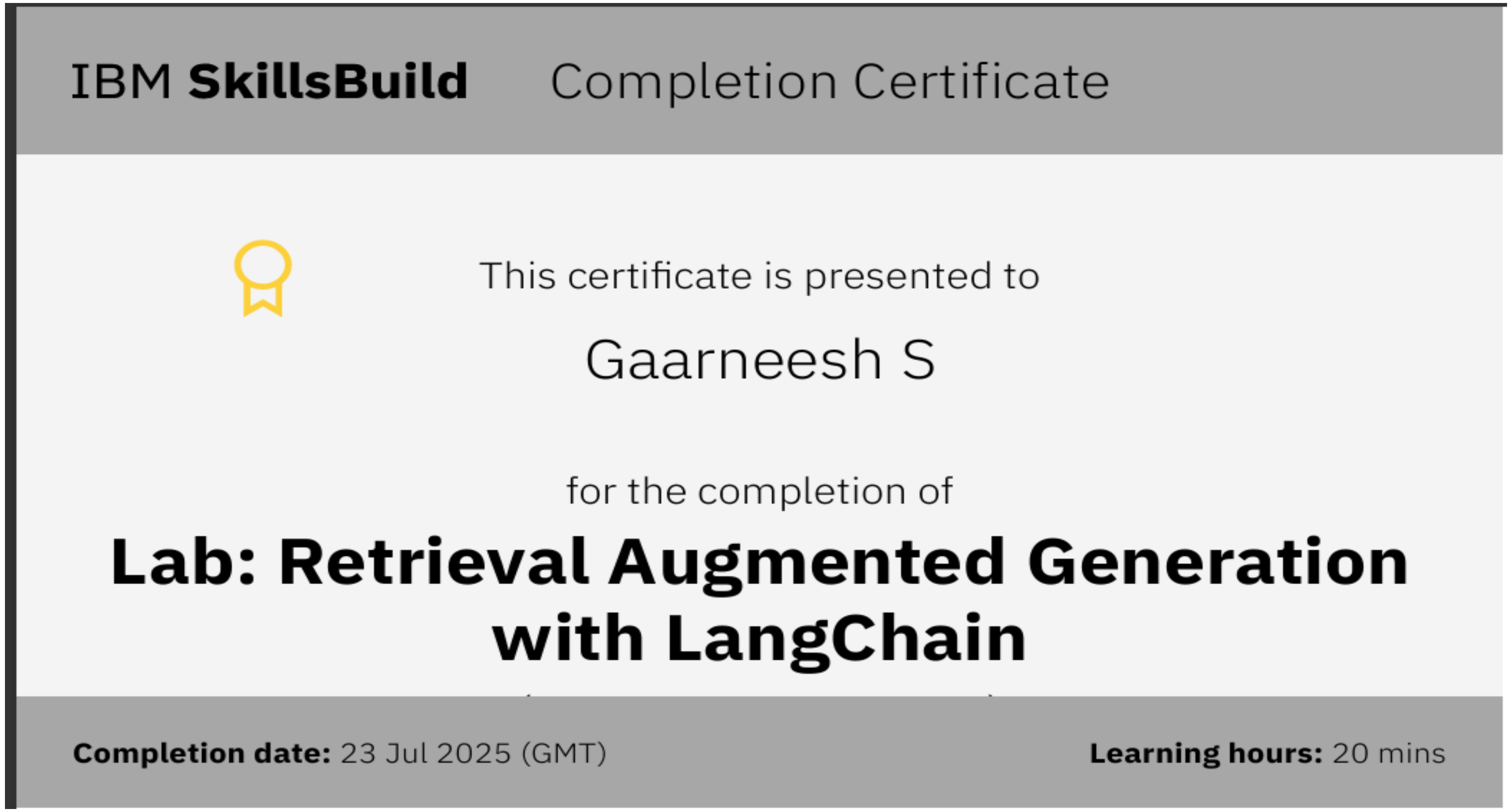
Issued on: Jul 17, 2025
Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/1b4258df-aac0-43f3-9523-92c040bb530c

IBM

IBM CERTIFICATIONS

- Screenshot/ credly certificate(RAG Lab)



IBM **SkillsBuild** Completion Certificate

This certificate is presented to

Gaarneesh S

for the completion of

Lab: Retrieval Augmented Generation with LangChain

Completion date: 23 Jul 2025 (GMT) **Learning hours:** 20 mins

THANK YOU