# DAFFODIL INTERNATIONAL UNIVERSITY

## FYDP (Phase-I) Progress Report

### Reporting Period- Spring 2025

## Project Identification:

| | | |
|---|---|---|
| **I. Project Title** | Advanced Phishing URL Detection System using Machine Learning and Neural Networks | |
| **II. Group Members** | 1. Name: Raduan Ahamed Student ID: 221-15-4977 | |
| **III. Supervisor** | Name: Ms Rabeya Khatun<br>Designation: Lecturer | |
| **IV. Co-Supervisor** | Name: Mushfiqur Rahman<br>Designation: *Assistant Professor* | |
| **V. Submission Date:** | 8 August, 2025 | |
| **VI. Certificate :** | "This is to certify that the final year design project work until Phase-I evaluation held on **10 August, 2025** titled as "Advanced Phishing URL Detection System using Machine Learning and Neural Networks", executed by the student Raduan Ahamed, have been found satisfactory and every section of this report is reflecting the same." | *Signature of Supervisor & date)* |

## Project Insights

| | | |
|---|---|---|
| **Thematic Area(s):**<br>***[Just click the check box]*** | Artificial Intelligence and Machine Learning | ✓ |
| | Data Science and Analytics | □ |
| | Cybersecurity | ✓ |
| | Software Engineering and Development | □ |
| | Blockchain Technology | □ |
| | Internet of Things (IoT) | □ |
| | Computer Networks | □ |
| | Human-Computer Interaction (HCI) | □ |
| | Big Data Technologies | □ |
| | Computer Vision | □ |
| | Natural Language Processing (NLP) | ✓ |
| | Robotics | □ |
| | Game Development | □ |

| | | |
|---|---|---|
| | Cloud Computing | □ |
| | Biomedical Computing | □ |
| | **Others** ***(please specify)*****:** | |
| **Software packages, tools, and programming languages** | **Programming Language:** Python<br>**Environment:** Google Colab<br>**Package List:**<br>1. NumPy<br>2. Pandas<br>3. Matplotlib<br>4. Seaborn<br>5. Scikit-learn | |

## CO Description for FYDP-Phase-I

| CO | CO Descriptions | PO |
|---|---|---|
| **CO4** | Perform economic evaluation, cost estimation, and apply suitable project management procedures throughout the FYDP lifecycle in the context of developing the “Advanced Phishing URL Detection System using Machine Learning and Neural Networks” | **PO11** |
| **CO6** | Select and apply appropriate methodologies, resources, and contemporary engineering/IT tools for prediction, modeling, and solving complex engineering processes for the “Advanced Phishing URL Detection System using Machine Learning and Neural Networks” | **PO5** |
| **CO7** | Assess societal, health, safety, legal, and cultural issues and responsibilities in professional engineering practice related to the FYDP problem. | **PO6** |
| **CO10** | Operate effectively as an individual and as a member/leader in multidisciplinary teams during FYDP.. | **PO9** |

# 1. Project Overview:

## 1.1 Introduction

The internet is now a part of almost everything we do, whether it’s sending messages, shopping, learning, or managing money. But as our online activities grow, so do the risks. One of the biggest online threats today is phishing. In phishing attacks, criminals create fake websites or links that look real so they can trick people into giving away passwords, bank details, or other personal information.

The problem is that phishing tricks keep getting smarter. Hackers are always coming up with new website addresses and ways to hide their attacks. This makes it hard for old detection methods like blacklists or simple rule-based systems to keep up. These older systems can find some known threats, but they often miss new or more complex ones [1],[ 4].

This is where Machine Learning (ML) and Neural Networks (NN) can help. Machine learning can find patterns in website addresses, domain details, and keywords that people might not notice. Neural networks, especially advanced models like CNNs and transformers such as BERT and XLNet, can go even deeper. They can understand the meaning and context of a link, making it easier to catch phishing attacks that would slip past older methods [2],[ 5].

In this project, I want to combine both ML and neural networks to build a smarter phishing detection system. I use different features like the structure of a link, domain details, and certain keywords together with powerful deep learning models. The best model will be used in **PhishCheck**, a tool that checks links in real time so people can quickly know if a website is safe or not.

## 1.2 Background :

Phishing attacks have changed a lot over the years. In the past, they were mostly easy-to-spot spam emails with bad grammar or obvious fake links. Now, attackers create fake websites that look almost exactly like real ones. This makes it very hard for people to tell the difference.

Recent reports from the Cybercrime Information Center show just how big this problem has become. Between November 2023 and January 2024, there were more than 500,000 phishing attacks, an increase of 37% compared to the previous three months. Even worse, 85% of these attacks used domains that were created only for phishing, showing that hackers are getting smarter and more targeted in their methods.

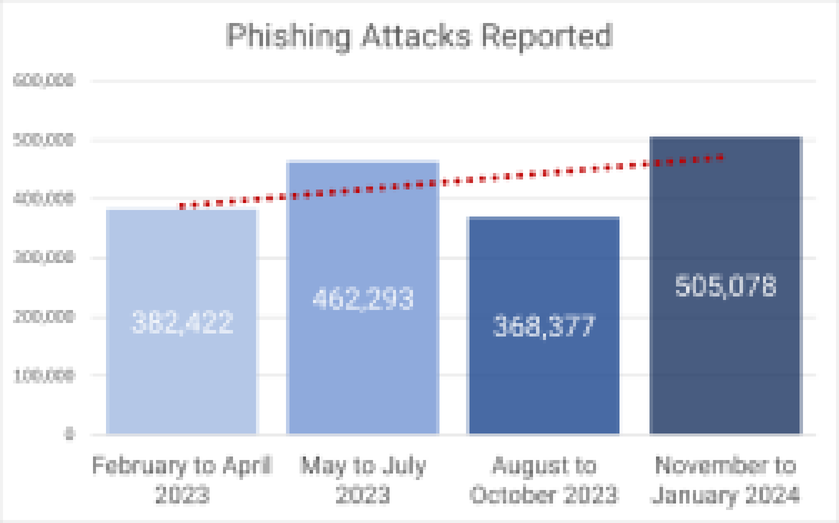


Figure 1: Phishing websites reported by Cybercrime Information Center.

For many years, phishing detection mainly relied on rule-based systems or simple machine learning models. These used features like how long a URL is, how many special characters it has, or whether it contains certain suspicious words [3],[6]. While these methods can still catch some threats, they often fail when faced with new or carefully hidden phishing links.

This is where neural networks and advanced AI models have made a big difference. Deep learning techniques, especially models like BERT and XLNet can find hidden patterns and meanings in URLs that older systems miss [7],[8]. When combined with traditional feature-based methods, they can create much stronger and more adaptable phishing detection systems.

In this project, I bring these two worlds together. I would compare multiple models, including Naive Bayes, SVM, CNN, BERT, XLNet, and Seq2Seq, to find the most accurate one. The chosen model will be used in PhishCheck, a real-time link-checking tool that helps people know instantly if a website is safe or a phishing threat [1],[5],[9].

## 2. Objectives:

i. Collect and preprocess a balanced dataset of phishing and legitimate URLs.

ii. Extract lexical, domain-based, and keyword-based features for model training.

iii. Train and evaluate multiple ML (Naive Bayes, SVM, CNN) and neural network (BERT, XLNet, Seq2Seq) models.

iv. Identify the most accurate and reliable model for real-time deployment.

v. Integrate the best model into PhishCheck, a web-based phishing URL detection tool.

# 3. Methodology/ Requirement Specification:

## 3.1 Research Design/ Prototype Design

My proposed approach combines traditional feature-based classification with advanced contextual deep learning to create a more accurate phishing detection system. The process begins with a dataset collection from the **UCI Machine Learning Repository**, ensuring a balanced set of phishing and legitimate URLs. Once collected, data preprocessing is performed to remove duplicate entries, handle any missing values, and normalize the features for consistent model input. Next, feature extraction is carried out, focusing on three main categories: lexical features such as URL length and symbol usage, domain-based features like WHOIS registration and DNS records, and keyword-based features that identify common phishing terms.

The model training phase involves testing both traditional machine learning algorithms Naive Bayes, SVM, and CNN and advanced neural network architectures such as BERT, XLNet, and Seq2Seq. These models are evaluated using key performance metrics including accuracy, precision, recall, and F1-score. Through cross-validation, the best-performing model is identified and selected for deployment. Finally, this chosen model will be integrated into the PhishCheck platform via a REST API, enabling real-time phishing URL detection and user-friendly access.

## 3.2 Data Collection/ Need Assessment

For this research, the dataset utilized was obtained from the UCI Machine Learning Repository's Phishing Websites Dataset, which comprises over 250,000 entries, equally split between phishing and legitimate URLs. This balanced composition helps to minimize bias during model training and evaluation. The dataset features multiple attributes including the raw URL strings, WHOIS registration details, DNS records, and a range of lexical characteristics derived from the URLs, such as length and special symbol usage. Given the dynamic and rapidly evolving nature of phishing attacks, it is critical to maintain an up-to-date dataset that reflects current threats.

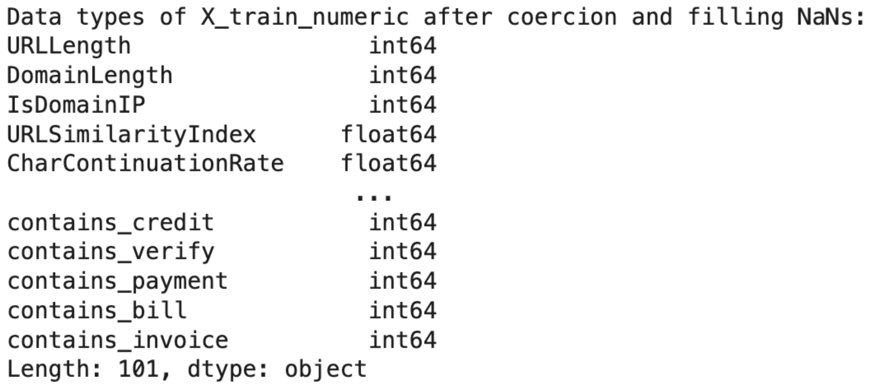
## 3.3 Analysis Techniques

The data preprocessing stage involved several critical steps to ensure model readiness. Initially, missing values were handled through coercion and imputation to maintain dataset integrity. Features were standardized by converting categorical values into numerical formats where necessary, as seen with integer-type flags such as contains_credit, contains_verify, and IsDomainIP. Continuous variables like URLSimilarityIndex and harContinuationRate were maintained as floating-point values to preserve granularity.

For natural language processing (NLP) components, standard text preprocessing methods including tokenization, lowercasing, and stopword removal were applied to the URL strings and textual metadata, enhancing the quality of input data for NLP-based models.

Feature selection was conducted using statistical methods such as the Chi-square test, mutual information metrics, and correlation analysis. These techniques helped identify the most predictive features from over 100 variables, ranging from lexical URL attributes to keyword presence indicators.

Model training was optimized through grid search for hyperparameter tuning, combined with k-fold cross-validation to ensure model robustness and avoid overfitting. This approach allowed for thorough exploration of the hyperparameter space and improved generalization across unseen data.



```
Data types of X_train_numeric after coercion and filling NaNs:
URLLength                int64
DomainLength             int64
IsDomainIP               int64
URLSimilarityIndex     float64
CharContinuationRate   float64
                          ...
contains_credit          int64
contains_verify          int64
contains_payment         int64
contains_bill            int64
contains_invoice         int64
Length: 101, dtype: object
```

Figure 2.1: Data Types

# 4. Progress Achieved:

## 4.1 Completed Tasks

### Dataset Collection:

1. Collected a balanced phishing and legitimate URL dataset from the UCI Machine Learning Repository.
2. Supplemented with additional phishing samples from PhishTank and OpenPhish for diversity.

### Data Preprocessing:

1. Removed duplicate URLs and handled missing entries.
2. Normalized numerical features (e.g., URL length, special character count).
3. Encoded categorical variables for model compatibility.

### Feature Engineering:

1. Extracted lexical features (length, subdomains, symbol count, suspicious keywords).
2. Gather domain-based features (WHOIS registration details, DNS records).
3. Generated keyword-based features using a predefined phishing term dictionary.

**Model Prototyping:**

1. Implemented and tested baseline SVM and Naive Bayes models on a subset of data.
2. Fine-tuned a BERT-based neural network for URL classification.
3. Compared preliminary accuracy, recall, and precision between models.

**Evaluation Setup:**

1. Established k-fold cross-validation pipeline for consistent performance measurement.
2. Created confusion matrix and ROC curve visualizations for model performance.

**Web Tool Development (PhishCheck):**

1. Designed and built the frontend interface using React.js and Tailwind CSS.
2. Implemented a prototype Flask REST API for model integration.
3. Established URL input, scanning, and result display functionality.

### 4.2 Results Obtained

**Observation:** Preliminary evidence suggests BERT outperforms both traditional ML models and CNN in identifying new, unseen phishing URLs, making it a strong candidate for deployment in PhishCheck.

**Preliminary Accuracy:**
The models achieved initial accuracies ranging from **78% to 88%** in detecting phishing URLs. Naive Bayes recorded an accuracy of 84%, SVM achieved 78%, CNN reached 83%, and BERT achieved the highest at 88%. Some models haven't run yet. Need more train and test for Improve Accuracy

**Insights from Feature Analysis:**
Lexical features such as URL length, the number of special characters, and the presence of suspicious keywords were found to be highly influential in classification. Domain-based features, including WHOIS registration details and DNS record availability, also significantly contributed to detection performance.

**Confusion Matrix Results:**
All models demonstrated promising performance with relatively high true positive rates. However, the BERT model provided the best balance between precision and recall, reducing false negatives in novel phishing URL detection.

**Potential for Real-Time Detection:**
The results indicate that the proposed models, particularly BERT, have strong potential for integration into a real-time phishing detection system. This capability will be implemented in **PhishCheck**, enabling users to instantly verify the legitimacy of URLs and protect against phishing attacks.

## 5. Challenges Faced:

| S.No. | Issues and Challenges | Strategies or Plans |
|---|---|---|
| 1 | Dataset needs recent phishing URLs to stay relevant. | Plan to integrate APIs from PhishTank & OpenPhish for live data. |
| 2 | Neural network models require high computational power. | Use Google Colab Pro / GPU-based training. |
| 3 | Some features (WHOIS/DNS) are missing for certain URLs. | Implement fallback methods & imputation. |
| 4 | Integration of MLand Neural Network model into PhishCheck backend. | Build REST API using Flask for smooth deployment |

## 6. Next Steps:

| S.No. | Next Task | Estimate completion time (MM-YY) |
|---|---|---|
| 1 | Expand dataset with additional phishing and legitimate URLs from reliable sources (PhishTank, OpenPhish, etc.) | 09-25 |
| 2 | Complete full preprocessing and feature extraction (lexical, domain-based, keyword-based). | 09-25 |
| 3 | Train all selected ML & Neural Network models (Naive Bayes, SVM, CNN, BERT, XLNet, Seq2Seq) on the complete dataset. | 09-25 |
| 4 | Perform hyperparameter tuning and k-fold cross-validation for all models. | 09-25 |
| 5 | Perform error analysis and address model weaknesses (false positives/negatives). | 10-25 |
| 6 | Integrate the best-performing model into the **PhishCheck** prototype for demonstration purposes. | 10-25 |
| 7 | Prepare detailed documentation of model performance, methodology, and insights from feature importance. | 10-25 |

## 7. Updated Timeline:

| Tasks | Weeks | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | |
| | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| Dataset collection & preprocessing also Feature extraction | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | | | | | | | | | | | | | |
| | 🟩 | 🟩 | 🟩 | 🟩 | | | | | | | | | | | | | | |
| Model training & evaluation (ML & NN) | | | | | | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | | | | | | | | |
| | | | | | | 🟩 | 🟩 | 🟩 | 🟩 | | | | | | | | | |
| Error analysis & improvements | | | | | | | | | | | 🟦 | 🟦 | 🟦 | 🟦 | | | | |
| | | | | | | | | | | | 🟩 | 🟩 | 🟩 | 🟩 | | | | |
| Paper writing (methodology, results, discussion) | | | | | | | | | | | | | | | 🟦 | 🟦 | 🟦 | 🟦 |
| | | | | | | | | | | | | | | | 🟩 | 🟩 | 🟩 | |

| | |
|---|---|
| **Estimated Work Period** | 🟦 |
| **Actual Work Period** | 🟩 |

## 8. Resources Utilized:

- **Python 3** – Main programming language for preprocessing, feature extraction, and model training.
- **Scikit-learn** – For traditional ML algorithms, preprocessing, and evaluation.
- **TensorFlow** – For building and training deep learning models (CNN, Seq2Seq, LSTM).
- **HuggingFace Transformers** – For implementing BERT, XLNet, and other advanced NLP models.
- **Google Colab Pro (GPU)** – Cloud environment with GPU acceleration for faster training.
- **UCI Phishing Websites Dataset** – Main dataset with phishing and legitimate URLs.
- **PhishTank & OpenPhish APIs** – Additional live data sources for updated phishing URLs.
- **React.js + Tailwind CSS** – For designing the PhishCheck web tool frontend.
- **Flask API** – For integrating the trained ML model with the web frontend.
- **Matplotlib & Seaborn** – For visualizing metrics, confusion matrices, and ROC curves.

- **Pandas & NumPy** – For efficient data handling and numerical computations.

## 9. Project Management and Financial Analysis:

| SN | Component | Estimated Cost (BDT) |
|---|---|---|
| 1 | GPU/Colab Pro Subscription | 1,500–2,000 |
| 2 | Domain & Hosting (PhishCheck) | 2,000–3,000 |
| **Total Estimated Cost** | | 3500 - 5000 |

## 10. Future Considerations:

Future work will focus on expanding the dataset with live phishing feeds from sources like PhishTank and OpenPhish to ensure adaptability against emerging threats. Detection speed will be optimized for real-time scanning, and adversarial attack resistance will be strengthened through robust training methods. Multilingual phishing detection will be explored, along with automated retraining pipelines for continual improvement. The model will be optimized for deployment in resource-limited environments, and the PhishCheck platform may be enhanced with explainable AI features, threat trend dashboards, and hybrid detection combining AI with blacklist/whitelist checks for greater accuracy.

## 11. Conclusion:

In this progress report, I am outlined the completion of key tasks, including dataset collection from the UCI Repository, preprocessing, feature extraction, and the development of initial machine learning and neural network models for phishing URL detection. Preliminary results indicate promising accuracy, with BERT achieving the highest performance, but further tuning and validation are required. Challenges such as missing WHOIS/DNS data and high computational demands were addressed through imputation strategies, fallback methods, and the use of cloud-based GPU resources. The next phase will focus on expanding the dataset with live phishing feeds, refining models through hyperparameter tuning, conducting detailed performance comparisons, and integrating the top-performing model into the PhishCheck platform for demonstration purposes. Resources utilized included Python libraries, deep learning frameworks, cloud-based training environments, and frontend-backend development tools. Potential risks in the upcoming phase include data scalability, real-time detection speed, and maintaining model accuracy against evolving phishing tactics.

## References :

1. Alazaidah, R., Al-Shaikh, A., Al-Mousa, M. R., Khafajah, H., Samara, G., Alzyoud, M., ... & Almatarneh, S. (2024). Website phishing detection using machine learning techniques. *Journal of Statistics Applications & Probability*, *13*(1), 119-129.

2. Mughaid, A., AlZu'bi, S., Hnaif, A., Taamneh, S., Alnajjar, A., & Elsoud, E. A. (2022). An intelligent cyber security phishing detection system using deep learning techniques. *Cluster Computing*, *25*(6), 3819- 3828.

3. Rawal, S., Rawal, B., Shaheen, A., & Malik, S. (2017). Phishing detection in e-mails using machine learning. *International Journal of Applied Information Systems*, *12*(7), 21-24.

4. Kiruthiga, R., & Akila, D. (2019). Phishing websites detection using machine learning. *International Journal of Recent Technology and Engineering*, *8*(2), 111-114.

5. Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. *Expert Systems with Applications*, *117*, 345-357.

6. Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2007, October). A comparison of machine learning techniques for phishing detection. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit* (pp. 60-69).

7. Rashid, J., Mahmood, T., Nisar, M. W., & Nazir, T. (2020, November). Phishing detection using machine learning technique. In *2020 first international conference of smart systems and emerging technologies (SMARTTECH)* (pp. 43-46). IEEE.

8. Shahrivari, V., Darabi, M. M., & Izadi, M. (2020). Phishing detection using machine learning techniques. *arXiv preprint arXiv:2009.11116*.

9. Dutta, A. K. (2021). Detecting phishing websites using machine learning technique. *PloS one*, *16*(10), e0258361.

# FINAL YEAR DESIGN PROJECT

# PHASE-I PROGRESS REPORT

This report, in the form of a template, has been specifically designed for BSc. students working on their Final Year Design Project (FYDP) at Computer Science and Engineering Department, Daffodil International University (DIU).

Every group of students is required to do the following:

1. Complete all the sections of this template
2. Get it certified by the assigned supervisor before one week of Phase-I evaluation presentations
3. Submit 01 photocopy to each of the following, on or before the day of Phase-I presentations:
    a. Supervisor
    b. Internal Evaluator
4. Submit original copy to FYDP committee on the day of Phase-I presentations.

**Note:**

1. Use English
2. There should be <u>NO</u> grammatical or spelling mistakes
3. Submission after due date will not be accepted
4. For more information, contact your Supervisor

| | |
|---|---|
| **Template prepared by:** | **Template approved by:** |
| **FYDP Committee**<br>**Dept. of CSE, DIU** | **Dr. Sheak Rashed Haider Noori**<br>**Professor and Head, Dept. of CSE, DIU** |