MAXIMIZE COMPUTATIONAL OFFLOADING OPTIMIZATION MULTI-USER MULTI-ACCESS FOG COMPUTING WIRELESS NETWORKS

A PROJECT REPORT

Submitted by

SANJAY V	**211419205143**
SANJAY PREETH D	**211419205144**
SATHISH KUMAR G	**211419205149**

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE, POONAMALLEE

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"MAXIMIZE COMPUTATIONAL OFFLOADING MULTI-USER MULTI-ACCESS FOG COMPUTING WIRELESS NETWORKS"** is the Bonafide work of **"SANJAY V (211419205143), SANJAY PREETH D (211419205144), SATHISH KUMAR G (211419205149)"** who carried out the project under my supervision.

SIGNATURE

Dr. M. HELDA MERCY M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Department of Information Technology

Panimalar Engineering College

Poonamallee, Chennai - 600 123

SIGNATURE

Ms.S.KUMARI M.E.,(Ph.D.,)

SUPERVISOR

Associate Professor

Department of Information Technology

Panimalar Engineering College

Poonamallee, Chennai - 600 123

Submitted for the project and viva-voice examination held on _____

SIGNATURE

INTERNAL EXAMINER

SIGNATURE

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project report entitled "**MAXIMIZE COMPUTATIONAL OFFLOADING MULTI-USER MULTI-ACCESS FOG COMPUTING WIRELESS NETWORKS**" which is being submitted in partial fulfillment of the requirement of the course leading to the award of the 'Bachelor Of Technology in Information Technology ' in **Panimalar Engineering College, Autonomous Institution Affiliated to Anna University- Chennai** is the result of the project carried out by me under the guidance and supervision of **Ms. S.Kumari M.E.,(Ph.D.,) Associate Professor in the Department of Information Technology**. We further declared that we or any other person has not previously submittedthis project report to any other institution/university for any other degree/ diploma or any other person.

SANJAY V

SANJAY PREETH D

SATHISH KUMAR G

Date**:**
Place: Chennai

It is certified that this project has been prepared and submitted under my guidance.

Date: **Ms. S.KUMARI M.E.,(Ph.D.,)**

Place: Chennai (Associate Professor / IT)

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion . We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to **Our Honorable Secretary and Correspondent, Dr. P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution.

We would like to express our deep gratitude to **Our Dynamic Directors , Mrs. C. VIJAYA RAJESHWARI and Dr. C. SAKTHI KUMAR,M.E., M.B.A.,Ph.D.,** and **Dr.SARANYA SREE SAKTHIKUMAR, B.E., M.B.A., Ph.D.,** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, **Dr. M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our Project co-ordinator **Mr. M. DILLI BABU**, **M.E., (Ph.D.,)** Associate Professor, Department of Information Technology for his guidance throughout the course of our project. We also express sincere thanks to our supervisor **Ms.S.KUMARI, M.E., (Ph.D).,** Associate Professor for providing the support to carry out the project successfully. Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

ABSTRACT

The rise of mobile services has impacted applications that require high computing power and quick responses, such as payment and recommendation systems. However, limited resources and time constraints in mobile devices lead to increased energy consumption and processing delays. The proposed project presents a mobile edge computing system for detecting malicious URLs on Android devices. The system leverages edge computing to perform URL analysis on a local device, rather than sending it to a remote server for analysis. The proposed framework addresses the optimization problem in a multilevel user mobile edge computing system by maximizing the minimum number of offloaded bits. The proposed system uses an Android mobile application to collect URLs and send them to an edge server for analysis. The server analyzes the URLs using a trained model and returns a response to the mobile application, indicating whether the URL is malicious or not. The proposed system is designed to facilitate the implementation of offloading strategies and the model in a modular way, and provides tools for informed runtime offloading decisions and seamless implementation of the model and strategies. The proposed system aims to reduce latency and improve efficiency in URL analysis on Android devices, while also ensuring the security and privacy of sensitive information.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OFABBREVATIONS

MEC	Mobile Edge Computing
URL	Uniform Resource Locator
MU	Mobile User
QCQP	Quadratically Constraint Quadratic Programming
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
ADT	Android Development Tool

CHAPTER 1

INTRODUCTION

INTRODUCTION

The main aim of this project is to develop a mobile application to detect whether the URL is a scam or a legit one.

1.1. OVERVIEW OF THE PROJECT

The project is focused on building a system for detecting malicious URLs in real-time using mobile edge computing (MEC) technology. The system consists of three main components: a mobile application for collecting URLs, an edge server for analyzing URLs, and a decision engine for making offloading decisions. The mobile application collects URLs from users and sends them to the edge server for analysis. The edge server, located at the edge of the network, analyzes the URLs using a trained machine learning model to determine if they are malicious or not. The decision engine is responsible for making decisions about which URLs should be offloaded to the edge server for analysis and which should be analyzed locally on the mobile device. The system leverages edge computing to perform the URL analysis on a local device, rather than sending it to a remote server for analysis. This provides faster response times and reduces the risk of sensitive information being transmitted over the network. The system is designed to facilitate the implementation of offloading strategies and the model in a modular way, providing tools for informed runtime offloading decisions and seamless implementation of the model and strategies.

1.2. NEED FOR THE PROJECT

As more and more services are moving towards the cloud, there is a growing need to provide efficient and timely responses to requests from mobile devices. However, sending data to the cloud for processing can result in high latency and can consume a lot of network resources. Edge computing can help address these challenges by bringing computation closer to the edge of the

network, where data is being generated. This can help reduce latency, lower network bandwidth usage, and enhance user experience. The proposed project aims to leverage edge computing for malicious URL detection by developing a mobile application that can offload URL analysis tasks to an edge server.

1.3. SCOPE OF THE PROJECT

The scope of this project is to develop a system that can effectively detect and classify malicious URLs on a mobile device using pre-trained learning techniques and edge computing. The system is designed to work on Android devices, and the goal is to provide fast and accurate detection of malicious URLs while minimizing the amount of data that needs to be sent to remote servers for analysis. By leveraging the power of edge computing, the system can perform URL analysis locally on the device, which reduces latency and improves the overall efficiency of the system. The proposed system can have applications in various fields such as cyber security, network security, and online safety.

CHAPTER – 2

LITERATURE SURVEY

Title: Joint Task Offloading and Resource Allocation for Multi-Task Multi-Server NOMA-MEC Networks

Authors: Jianbin Xue and Yaning An

Published Year: 2021

Efficiency:

Low Deployment Cost

Has the best acceptance ratio

Built-in error handling

Drawbacks:

Poor Application Performance

Unsuitable for large scale scenarios.

Solutions have been proved ineffective

Description:

By offloading computationally intensive tasks of smart end devices to edge servers deployed at the edge of the network, mobile edge computing (MEC) has become a promising technology to provide computing services for Internet of Things (IoT) devices. In order to further improve the access capability of MEC and increase the spectrum utilization efficiency, in this article, Non-Orthogonal Multiple Access (NOMA) technology is introduced into MEC systems and we study the computing offloading problem of multi-user, multi-task and multi-server through joint optimization of task offloading and resource allocation, we intend to maximize the systems processing capability as an optimization goal. To solve the proposed mixed integer nonlinear programming (MINLP) problem, the objective optimization problem is firstly decoupled into two sub-problems of resource allocation and task allocation. Secondly the resource allocation problem is further decomposed into computation resource optimization and communication resource allocation. For the communication resource allocation, it first fixed power allocation, then the sub-channel allocation problem is regarded as a many-to-one matching problem between sub-channels and users. In addition, we propose a low complexity sub-

optimal matching algorithm for sub-channel allocation to maximize the offloading efficiency. Based on our proposed sub-channel allocation scheme, the transmission power allocation is regarded as a convex optimization problem, which is tackled by Lagrangian multiplier method. Finally, under the condition of resource allocation, the tasks of all end devices (EDs) are allocated. Experimental numerical results show that the proposed scheme can effectively decrease latency and energy consumption of networks, improve system processing capability, and further improve MEC system performance.

To enhance the performance of MEC systems and further improve user experience, we proposed a novel network scenario which we used NOMA to transmit the offloading tasks of EDs to multiple edge servers. In the paper, we presented an optimization framework for a multi-user multi-task and multiserver. NOMA-MEC system to maximize system processing capability via jointly optimizing the tasks offloading and resources allocation. By decomposing the formulated problem, an efficient algorithm was proposed to tackle the formed problem under the help of convex optimization theory. Through computer simulation, the effectiveness of the proposed scheme was verified.

Title: Task Data Offloading and Resource Allocation in Fog Computing With Multi-Task Delay Guarantee

Authors: Mithun Mukherjee , Suman Kumar , Qi Zhang , Rakesh Matam , Constandinos X. Mavromoustakis , Yunrong Lv and George Mastorakis

Published Year:2019

Efficiency:

Quick Calculation Time

Have Well-Understood Formal Properties

Provides the integrity and non-transferability.

Drawbacks:

Approach is a bit time-consuming

Unsuitable for large scale scenarios.

Additional configuration is required

Description:

With the emergence of delay-sensitive task completion, computational offloading becomes increasingly desirable due to the end-users limitations in performing computation-intense applications. Interestingly, fog computing enables computational offloading for the end-users towards delay sensitive task provisioning. In this paper, we study the computational offloading for the multiple tasks with various delay requirements for the end-users, initiated one task at a time in end-user side. In our scenario, the end-user offloads the task data to its primary fog node. However, due to the limited computing resources in fog nodes compared to the remote cloud server, it becomes a challenging issue to entirely process the task data at the primary fog node within the delay deadline imposed by the applications initialized by the end-users. In fact, the primary fog node is mainly responsible for deciding the amount of task data to be offloaded to the secondary fog node and/or remote cloud. Moreover, the computational resource allocation in term of CPU cycles to process each bit of the task data at fog node and transmission resource allocation between a fog node to the remote cloud are also important factors to be considered. We have formulated the above problem as a Quadratically Constraint Quadratic Programming (QCQP) and provided a solution. Our extensive simulation results demonstrate the effectiveness of the proposed offloading scheme under different delay deadlines and traffic intensity levels.

In this paper, we address the issues of task data offloading in fog computing considering different delay deadline for the tasks initiated by the end-users. Our approach takes into account the delay deadline for different tasks, the trans- mission delay between primary fog node to the cloud, and secondary fog nodes available computing resources while offloading the task

data. Simulation results have shown that the proposed solution outperforms xed computational and transmission resource allocation to satisfy the delay deadline. Moreover, a trade-off between the deadline on the latency and the delay violation is observed with numerous parameter settings. Further extensions of this work may include the investigation of task offloading for carrier-grade reliability and latency constraints with joint and competitive caching designs based on network utility maximization.

Title: FairEdge: A Fairness-Oriented Task Offloading Scheme for Iot Applications in Mobile Cloudlet Networks

Authors: Shuang Lai , Xiaochen Fan , Qianwen Ye , Zhiyuan Tan , Yuanfang Zhang , Xiangjian He and Priyadarsi Nanda

Published Year:2020

Efficiency:

Streamlined and decoupled services

Powerful Representation

Capability Simplicity and Explainability.

Drawbacks:

Difficult and Less Commonly used

High complexity, inaccuracy, and inadequacy

Have not been investigated thoroughly

Description:

Mobile cloud computing has emerged as a promising paradigm to facilitate computation-intensive and delay-sensitive mobile applications. Computation offloading services at the edge mobile cloud environment are provided by small-scale cloud infrastructures such as cloudlets. While offloading tasks to in-proximity cloudlets enjoys benefits of lower latency and smaller energy consumption, new issues related to the cloudlets are rising. For instance,

unbalanced task distribution and huge load gaps among heterogeneous mobile cloudlets are becoming more challenging, concerning the network dynamics and distributed task offloading. In this paper, we propose FairEdge, a Fairness-oriented computation offloading scheme to enable balanced task distribution for mobile Edge cloudlet networks. By integrating the balls-and-bins theory with fairness index, our solution promotes effective load balancing with limited information at low computation cost. The evaluation results from extensive simulations and experiments with real-world datasets show that, FairEdge outperforms conventional task offloading methods, and it can achieve a network fairness up to 0.85 and reduce the unbalanced task offload by 50%.

In this paper, we have proposed FairEdge, a Fairness-oriented task offloading scheme to enable balanced task sharing and computation offloading for mobile Edge cloudlet networks. The FairEdge integrates balls-into-bins theory and Jains fairness index for distributed task offloading among mobile cloudlets. We have developed the system model of computation offloading in edge mobile cloudlet networks, and formulated the load balancing problem together with fair- ness optimization problem. By adopting the two-choice paradigm and using calculated fairness index values for cloudlets and the network, we have further proposed algorithm design of FairEdge and conducted extensive evaluation studies with simulations and experiments on real-world trace datasets. The experimental results have shown that FairEdge successfully achieved load balancing with guaranteed performance, with a near-optimal fairness index of up to 0.85 and an improvement of 50% over conventional baseline methods. (Shuang Lai and Xiaochen Fan contributed equally to this work.)

Title: Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems

Authors: Samrat Nath and Jingxian Wu

Published Year: 2020

Efficiency:

Simple to understand and interpret

Improve the real-time feasibility without loss of optimality.

May not meet the real-time requirement.

Drawbacks:

Cannot meet current network business demands

High complexity of installing and maintaining

Heavyweight

Description:

Mobile Edge Computing (MEC) is one of the most promising techniques for next-generation wireless communication systems. In this paper, we study the problem of dynamic caching, computation offloading, and resource allocation in cache-assisted multi-user MEC systems with stochastic task arrivals. There are multiple computationally intensive tasks in the system, and each Mobile User (MU) needs to execute a task either locally or remotely in one or more MEC servers by offloading the task data. Popular tasks can be cached in MEC servers to avoid duplicates in offloading. The cached contents can be either obtained through user offloading, fetched from a remote cloud, or fetched from another MEC server. The objective is to minimize the long-term average of a cost function, which is defined as a weighted sum of energy consumption, delay, and cache contents fetching costs. The weighting coefficients associated with the different metrics in the objective function can be adjusted to balance the tradeoff among them. The optimum design is performed with respect to four decision parameters: whether to cache a given task, whether to offload a given uncached task, how much transmission power should be used during offloading, and how much MEC resources to be allocated for executing a task. We propose to solve the problems by developing a dynamic scheduling policy based on Deep Reinforcement Learning (DRL) with the Deep Deterministic Policy Gradient (DDPG) method. A new decentralized DDPG algorithm is developed to obtain the optimum designs for multicell MEC systems by leveraging on the

cooperations among neighboring MEC servers. Simulation results demonstrate that the proposed algorithm outperforms other existing strategies, such as Deep Q-Network (DQN).

Title: Cost-Efficient Dependent Task Offloading for Multiusers

Authors: Yinuo Fan , Linbo Zhai and Hua Wang

Published Year: 2019

Efficiency:

Ability to Deliver High Quality Results

Reduces the consumption of hardware resources

Efficiently improve time efficiency involves the computation time and communication time

Drawbacks:

Solutions have been proved ineffective

Poor Application Performance

High level of communication and computation overheads

Description:

The extensive use of mobile intelligent devices, such as smart phones and tablets, induces new opportunity and challenge for computation offloading. Task offloading is an important issue in a system consisting of multiple types of devices, such as mobile intelligent devices, local edge hosts and a remote cloud server. In this paper, we study the offloading assignment of multiple applications, each one comprising several dependent tasks, in such a system. To evaluate the total cost in the offloading process, a new metric is introduced to take into account features of different devices. The remote server and local hosts are more concerned about their processors utilization, while mobile devices pay more attention to their energy. Therefore, this metric uses relative energy consumption to denote the cost of mobile devices, and evaluates the cost of the remote server and local hosts by the processor cycle number of task execution.

We formulate the offloading problem to minimize the system cost of all applications within each applications completed time deadline. Since this problem is NP-hard, the heuristic algorithm is proposed to offload these dependent tasks. At first, our algorithm arranges all tasks from different applications in a priority queue considering both completed time deadline and task-dependency requirements. Then, based on the priority queue, all tasks are initially assigned to devices to protect mobile devices with low energy and make them survive in the assignment process as long as possible. At last, to obtain a better schedule realizing lower system cost, based on the relative remaining energy of mobile devices, we reassign tasks from high-cost devices to low-cost devices to minimize the system cost. Simulation results show that our proposed algorithm increases the successfully completed probability of whole applications and reduces the system cost effectively under time and energy constraints.

In this paper, offloading assignment is studied in a system consisting of mobile intelligent devices, local edge hosts and a remote cloud server. We dene a more realistic system architecture and introduce relative energy consumption as a metric to react the real cost of mobile devices. Based on the system architecture, the offloading problem is formulated to minimize the system cost within each application completed deadline.

Title: Computation Offloading and Task Scheduling for DNN-Based Applications in Cloud-Edge Computing

Authors: Zheyi Chen , Junqin Hu , Xing Chen , Jia Hu , Xianghan Zheng and Geyong Min

Published Year: 2020

Efficiency:

Effectiveness for distributed optimization

Reduces the consumption of hardware resources

It provides easy information processing and cost reduction as well

Drawbacks:

Solutions have been proved ineffective

Maximizes the complexity of the problem

Heavyweight

Description:

Due to the high demands of deep neural network (DNN) based applications on computational capability, it is hard for them to be directly run on mobile devices with limited resources. Computation offloading technology offers a feasible solution by offloading some computation-intensive tasks of neural network layers to edges or remote clouds that are equipped with sufficient resources. However, the offloading process might lead to excessive delays and thus seriously affect the user experience. To address this important problem, we first regard the average response time of multi-task parallel scheduling as our optimization goal. Next, the problem of computation offloading and task scheduling for DNN-based applications in cloud-edge computing is formulated with a scheme evaluation algorithm. Finally, the greedy and genetic algorithms based methods are proposed to solve the problem. The extensive experiments are conducted to demonstrate the effectiveness of the proposed methods for scheduling tasks of DNN-based applications in different cloud-edge environments. The results show that the proposed methods can obtain the near-optimal scheduling performance, and generate less average response time than traditional scheduling schemes. Moreover, the genetic algorithm leads to less average response time than the greedy algorithm, but the genetic algorithm needs more running time.

In this paper, we formulate the problem of computation offloading and task scheduling for DNN based applications in cloud-edge environments and design a scheme evaluation mechanism. Next, the greedy and genetic algorithms based methods are proposed to efficiently explore the suit- able schemes. The extensive experiments are conducted to verify the effectiveness of the proposed

methods in different scenarios of cloud-edge environments. The results show that the genetic algorithm leads to less average response time than other scheduling schemes but needs more running time than the greedy algorithm. Therefore, these two proposed algorithm define tasks, since these tasks are not sensitive to the training time while the genetic algorithm can promise better application performance. By contrast, the genetic algorithm would be a better choice when dealing with online tasks, because these tasks require fast decision-making ability, which is also the advantage of the greedy algorithm.

Title: An Energy-Efficient and Deadline-Aware Task Offloading Strategy Based on Channel Constraint for Mobile Cloud Workflows

Authors: Yingjie Wang , Lei Wu , Xiusheng Yuan , Xiao Liu and Xuejun Li

Published Year: 2019

Efficiency:

Improved reliability and resilience

Provides the integrity and nontransferablity.

Excellent empirical performance

Drawbacks:

Narrowly specialized knowledge

Maximizes the complexity of the problem

Approach is a bit time-consuming

Description:

Energy efficiency is a fundamental problem due to the fact that numerous tasks are running on mobile devices with limited resources. Mobile cloud computing (MCC) technology can offload computation-intensive tasks from mobile devices onto powerful cloud servers, which can significantly reduce the energy consumption of mobile devices and thus enhance their capabilities. In MCC, mobile devices transmit data through the wireless channel. However, since the state of the channel is dynamic, offloading at a low transmission rate

will result in the serious waste of time and energy, which further degrades the quality of service (QoS). To address this problem, this paper proposes an energy-efficient and deadline-aware task offloading strategy based on the channel constraint, with the goal of minimizing the energy consumption of mobile devices while satisfying the deadlines constraints of mobile cloud workflows. Specifically, we first formulate a task offloading decision model that combines the channel state with task attributes such as the workload and the size of the data transmission to determine whether the task needs to be offloaded or not. Afterward, we apply it to a new adaptive inertia weight-based particle swarm optimization (NAIWPSO) algorithm to create our channel constraint-based strategy (CC-NAIWPSO), which can obtain a near-optimal offloading plan that can consume less energy while meeting the deadlines. The experimental results show that our proposed task offloading strategy can outperform other strategies with respect to the energy consumption of mobile devices, the execution time of mobile cloud workflows, and the running time of algorithms.

The energy consumption of mobile devices is a critical issue in mobile cloud computing. Meanwhile, due to the dynamic network state, it is a big challenge to ensure the satisfaction of deadline requirements. In this paper, to reduce the energy consumption of mobile devices while satisfying the deadline requirements of mobile workflow applications, we proposed an energy-efficient task offloading strategy CC-NAIWPSO based on the channel constraint. We formulated a task offloading decision model with the channel constraint and applied it with the NAIWPSO algorithm to create the CCNAIWPSO strategy, which aims to achieve the near-optimal offloading plan.

Title: Task Caching, Offloading, and Resource Allocation in D2D-Aided Fog Computing Networks

Authors: Yanwen Lan , Xiaoxiang Wang , Dongyu Wang , Zhaolin Liu and Yibo Zhang

Published Year: 2019

Efficiency:

Simplicity and Explainability.

It is a fast and easy procedure to perform Effectiveness for distributed optimization

Drawbacks:

Prone to Errors

Solutions have been proved ineffective

Additional configuration is required

Description:

In this paper, we investigate the allocation of resource in D2D-aided Fog computing system with multiple mobile user equipment (MUEs). We consider each MUE has a request for task from a task library and needs to make a decision on task performing with a selection of three processing modes which include local mode, fog offloading mode, and cloud offloading mode. Two scenarios are considered in this paper, which mean task caching and its optimization in off-peak time, task offloading, and its optimization in immediate time. In particular, task caching refers to cache the completed task application and its related data. In the first scenario, to maximize the average utility of MUEs, a task caching optimization problem is formulated with stochastic theory and is solved by a GAbased task caching algorithm. In the second scenario, to maximize the total utility of system, the task offloading and resource optimization problem is formulated as a mixed integer nonlinear programming problem (MINLP) with a joint consideration of the MUE allocation policy, task offloading policy, and computational resource allocation policy. Due to the nonconvex of the problem, we transform it into multi-MUEs association problem (MMAP) and mixed Fog/Cloud task offloading optimization problem (MFCOOP). The former problem is solved by a Gini coefficient-based MUEs allocation algorithm which can select the most proper MUEs who contribute more to the total utility. The task offloading optimization problem is proved as a

potential game and solved by a distributed algorithm with Lagrange multiplier. At last, the simulations show the effectiveness of the proposed scheme with the comparison of other baseline schemes.

In this paper, we investigated the allocation of resource in caching-enabled Fog computing system with multiple mobile user equipments (MUEs). We assumed MUEs should make the decision on task performing with a selection of three task processing modes including local mode, Fog offloading mode and Cloud offloading mode. Two scenarios were considered in this paper, task caching and its optimization in off-peak time, task offloading and its optimization in immediate time.

Title: Joint RAN Slicing and Computation Offloading for Autonomous Vehicular Networks: A Learning-Assisted Hierarchical Approach

Authors: Qiang Ye , Weisen Shi , Kaige Qu , Hongli He , Weihua Zhuang and Xuemin Shen

Published Year: 2021

Efficiency:

Corresponding time cost is greatly reduced.

Quick Calculation Time

Improve the real-time feasibility without loss of optimality

Drawbacks:

Difficult to be used in large-scale parallel computing.

Solutions have been proved ineffective

Significantly increases capital and operating expenditures

Description:

In this paper, a two-timescale radio access network (RAN) slicing and computing task offloading problem is investigated for a cloud-enabled autonomous vehicular network (C-AVN). We aim at jointly maximizing the communication and computing resource utilization with diverse quality-of-

service (QoS) guarantee for autonomous driving tasks. Specifically, to capture the small-timescale network dynamics, a computing task scheduling problem is formulated as a stochastic optimization program, for maximizing the long-term network-wide computation load balancing with minimum task offloading variations. Due to the large problem size and unavailable network state transition probabilities, we employ cooperative multi-agent deep Q-learning (MA-DQL) with fingerprint to solve the problem by learning the set of stationary task offloading policies with stabilized convergence. Given the task offloading decisions, we further study a RAN slicing problem in a large timescale, which is formulated as a convex optimization program. We focus on optimizing the radio resource slicing ratios among base stations, to maximize the aggregate network utility with statistical QoS provisioning for autonomous driving tasks. Based on the impact of radio resource slicing on computation load balancing, we propose a two-timescale hierarchical optimization framework to maximize both communication and computing resource utilization. Extensive simulation results are provided to demonstrate the effectiveness of the proposed framework in comparison with state-of-the-art schemes.

In this paper, two-timescale RAN slicing and computation offloading are jointly studied in a C-AVN in support of di- verse autonomous driving tasks. To capture small-timescale network dynamics, a computing task scheduling problem is formulated as a stochastic optimization program with constraints on task offloading latency and computation capacity. The objective is to achieve the network wide computation load balancing with controlled task offloading variations among edge servers. To deal with the large problem size with potentially unknown state transition probabilities, we develop a cooperative MA-DQL framework with fingerprint to learn the stationary task offloading policy with stabilized learning performance. Given the task offloading policy in a planning window, a large timescale RAN slicing problem is further investigated to maximize the overall radio resource utilization with statistical QoS guarantee for different autonomous driving tasks.

Title: Joint Wireless Source Management and Task Offloading in Ultra-Dense Network

Authors: Shanchen Pang and Shuyu Wang

Published Year: 2020

Efficiency:

Performs better on various circumstances and environment

also improve the operational efficiency.

Reducing data transmission and improving the scalability.

Drawbacks:

High complexity, inaccuracy, and inadequacy

Maximizes the complexity of the problem

Difficulties to obtain better performance

Description:

The ultra-dense network (UDN) based on mobile edge computing (MEC) is an important technology, which can achieve the low-latency of 5G communications and enhance the quality of user experience. However, how to improve the task offloading efficiency is a hot topic of UDN under the constraint on the limited wireless resources. In this article, we propose a heuristic task offloading algorithm HTOA to optimize the delay and energy consumption of offloading tasks in UDN. Firstly, a convex programming model for MEC resource allocation is established, which aims to obtain the optimal allocation set of resources for offloading tasks, and optimize the execution delay of offloading tasks. Followed by, the problem of joint channel allocation and user upload power control is solved by the greedy strategy and golden section method, which aims to optimization the delay and energy consumption of task upload data. Compared with the random task offloading algorithm, numerical simulations show that the algorithm HTOA can effectively reduce the delay and energy consumption of task offloading, and perform better as the number of users increases.

In this paper, we propose a heuristic task offloading algorithm HTOA for investigated the joint radio resource management and task offloading in the UDN architecture. In order to maximize the task offloading benefits (the weighted sum of delay and energy consumption), we considered the management of radio resources: the MEC computing resource allocation, the channel allocation, and the user upload power control. The proposed algorithm HTOA achieves optimized task offloading by iteratively updating radio resource management. Simulation results show that the algorithm HTOA has higher user offloading benets than the random offloading algorithm efficiently. Especially, when there are a large number of users, the advantage is more obvious, which further proves that the proposed algorithm HTOA is better applicable to UDN environment. In future work, we will further consider the complex scenario of user mobility and the energy consumption of MEC servers.

2.1 FEASIBILITY STUDY

A feasibility study is a high-level capsule version of the entire System analysis and Design. The study begins by classifying the problem definition. Feasibility is to determine if it's worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully. There are 3 parts in feasibility study.

1) Operational Feasibility
2) Technical Feasibility
3) Economic Feasibility

Overall, the proposed project is technically feasible, and the required technologies are readily available. The project's economic feasibility will depend on the availability of funding, The project's operational feasibility will depend on the availability of volunteers and the support of law enforcement agencies.

OPERATIONAL FEASIBILITY

Operational feasibility for this project is high as it utilizes readily available technologies and tools, and the deployment of the system is simple. The data collection and preprocessing modules are automated, making them less labor-intensive. The system can be integrated with various web browsers and can run on different devices. The only requirement is an internet connection, which is easily accessible nowadays. The maintenance and updating of the system are also feasible as it can be done remotely. Overall, the project is operationally feasible, and its deployment and maintenance require minimal resources.

TECHNICAL FEASIBILITY

The technical feasibility of the project is high. The project uses widely used programming languages and frameworks such as Python, TensorFlow, and Android Studio, which are well-supported and have a large community of developers. The required hardware resources are also relatively low, and the project can be run on a standard laptop or desktop computer. The dataset used for training the model is publicly available, and the preprocessing steps are straightforward. The trained model can be deployed on a mobile device for real-time detection of malicious URLs. Overall, the project is technically feasible and can be implemented using existing technologies and resources.

ECONOMICAL FEASIBILITY

The economical feasibility of this project can be analyzed based on the cost required for its implementation and the benefits it provides. The cost involved in this project mainly includes the cost of hardware and software required for development and testing, the cost of data collection and preprocessing, and the cost of maintaining the system. The benefits of this project include the detection of malicious URLs, which can prevent users from accessing harmful websites and protect them from cyber threats. This can save organizations and individuals from the potential financial losses and reputational damage caused by cyber-attacks. Therefore, the benefits of this project outweigh its cost, making it economically feasible.

CHAPTER – 3

SYSTEM DESIGN

3.1 Existing System:

The current system proposes a multi-class, a multi-layered, and also a multi-user framework for computing at edge level that effectively offloads tasks and performs computations while considering important system requirements. The framework includes wireless communication and task computation constraints, and tasks are classified based on their QoS requirements and assigned priority levels according to their latency requirements.

Devices offload their generated tasks to any one of the three layers in the suggested system, which consists of three layers altogether. If the size of the queue is less than a threshold which is set, only a job is accepted.

The efficacy of the combined computing of edge-fog-cloud environment was assessed with the help of stochastic geometry, parallel computing, and also involves queuing theory methods. This brought to light the value of a multi-class, multi-layer edge computing platforms in utilizing and increasing the usability of mission-critical and applications that are delay-sensitive.

The system models each layer which consists of the computation by the help of parallel computing and theory of queuing methods to make sure that numerous jobs can be completed concurrently among virtual machines.

Interference between input and output (I/O) is also considered by the algorithm. It is crucial to categorize tasks according to their delay requirements in real-world apps where some tasks are delay-sensitive and others are delay-tolerant.

Task requests can be given varying degrees of priority, and in order to comply with QoS requirements, tasks must be computed before their various deadlines.

3.1.1 Drawbacks:

The drawbacks of the existing system:

1. The proposed system is highly complex and may require significant expertise and resources to implement and maintain.
2. The system relies heavily on wireless communication and task computation constraints, which may limit its effectiveness in certain situations.
3. The task classification and prioritization system may not always be accurate or effective, leading to delays or errors in task completion.
4. The system may struggle to handle high volumes of concurrent tasks, leading to performance issues or even system crashes.
5. The system does not appear to be highly scalable, and may struggle to handle rapid growth or changes in demand over time.
6. The reliance on stochastic geometry and queuing theory methods may limit the applicability of the system in certain contexts, as these methods may not always accurately reflect real-world conditions or behaviors.

3.2 Proposed System:

The proposed algorithm addresses a non-convex optimization problem in a multilevel user of (MEC) mobile edge computing system by maximizing the minimum number of the offloaded bits. Offloading requires communication among hosts, and jobs have associated relative deadlines.

The proposed framework enables effective offloading decision-making by providing tools for informed runtime offloading decisions and seamless implementation of the model and strategies.

The proposed system leverages edge computing to perform the URL analysis on a local device, rather than sending it to a remote server for analysis. This provides faster response times and reduces the risk of sensitive information being transmitted over the network.

The proposed framework is designed to facilitate the implementation of offloading strategies and the model in a modular way. The proposed system for the malicious URL detection project involves using an Android mobile application to collect URLs and send them to a server.

The server, which is located at the edge, analyzes the URLs and returns a response to the mobile application, indicating whether the URL is malicious or not. The system uses the trained model to detect malicious URLs and incorporates edge computing to reduce latency and improve efficiency.

3.2.1 Advantages:

The proposed system has several advantages:

1. Faster response time: The use of edge computing reduces the response time for URL analysis since the analysis is performed locally on the edge server, instead of being sent to a remote server for analysis. This reduces the risk of sensitive information being transmitted over the network.
2. Improved efficiency: The proposed system leverages edge computing to perform the URL analysis, which enables the system to efficiently offload tasks and perform computations while considering important system requirements.
3. Modular implementation: The proposed framework is designed to facilitate the implementation of offloading strategies and the model in a modular way. This enables the system to be easily adapted to different environments and use cases.
4. Seamless implementation: The proposed framework provides tools for informed runtime offloading decisions and seamless implementation of the model and strategies, which makes it easier to implement and use.
5. Increased security: By performing the analysis locally on the edge server, the proposed system reduces the risk of sensitive information being transmitted over the network, which enhances the security of the system.

3.3 Architecture Diagram:

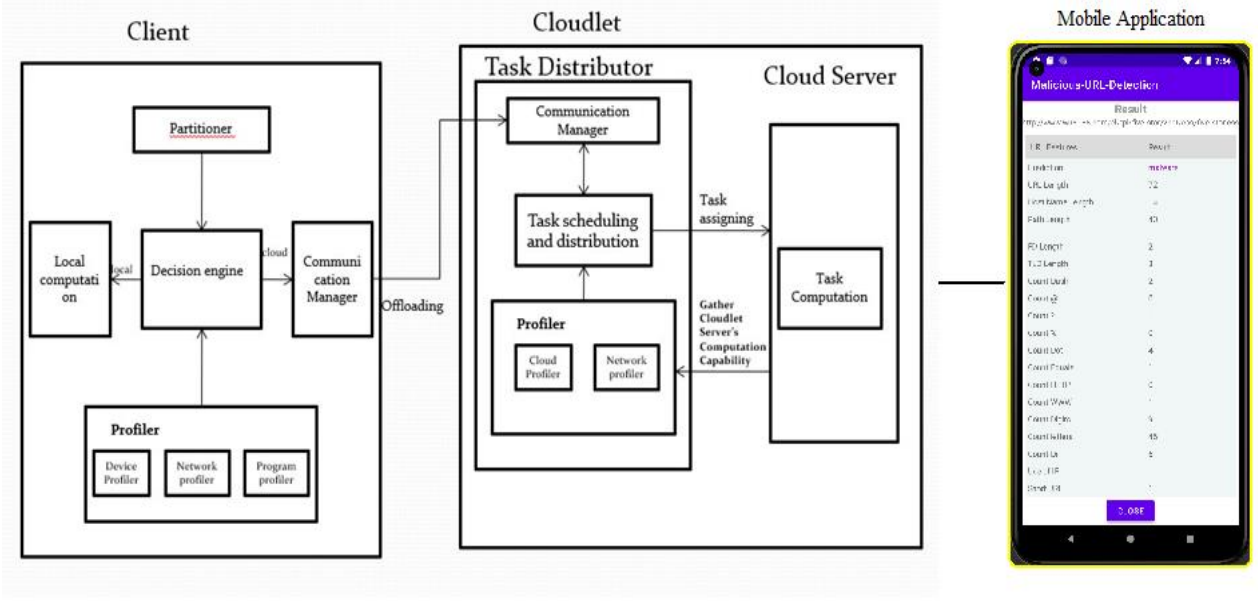

Fig.3.1: Proposed System Architecture

The above architecture diagram shows the communication process that takes between the Client and Cloudlet that executes the task.

A. Client:

The client is a user who interacts with the system to submit a URL for analysis. The client sends the URL to the edge server for processing. The client can also receive the results of the analysis from the edge server and display them to the user. The client uses a mobile application with internet connectivity that can communicate with the edge server. The client is responsible for providing a user-friendly interface for the user to interact with the system and submitting the URL to the edge server for analysis.

B. Cloudlet:

The cloudlet, which is acts as the edge server plays a critical role in the overall architecture. The main responsibility of the edge server is to perform real-time analysis of URLs received from clients, detect malicious URLs using the trained model, and return the results back to the client.

The edge server receives URL requests from clients through the HTTP interface mobile application. Once a URL request is received, the edge server extracts the URL and passes it to the machine learning model for analysis. The machine learning model then performs real-time analysis of the URL and predicts whether it is malicious or not. Based on the prediction, the edge server returns the result (either 'Malicious' or 'Safe') back to the client.

The edge server is represented in the system of the user written using programs and is located closer to the client, typically at the network edge, which enables faster response times, reduces network latency, and improves overall performance.

By processing data closer to the source, the edge server which is the system also reduces the load on the cloud server, which can result in significant cost savings.

Overall, the edge server plays a critical role in enabling real-time, efficient, and scalable processing of data at the network edge, which is becoming increasingly important in today's data-driven world.

C. Mobile Application:

The mobile application in the proposed project is an Android application that allows users to input URLs for analysis. The application collects URLs from the user and sends them to the edge server for analysis. The application also receives a response from the edge server indicating whether the URL is malicious or not.

The application is designed to be user-friendly and intuitive, with a simple interface that allows users to quickly and easily submit URLs for analysis. The application also incorporates features such as caching to reduce latency and improve efficiency, and error handling to ensure that the application operates smoothly and reliably.

Overall, the mobile application plays a crucial role in the proposed system by providing a convenient and accessible interface for users to interact with the

system and submit URLs for analysis.

3.4 Project Modules:

Table 3.1: Dataset for the URL Model Training

URL	Type
br-icloud.com.br	phishing
mp3raid.com/music/krizz_kaliko.html	benign
bopsecrets.org/rexroth/cr/1.htm	benign
http://www.garage-pirenne.be/index.php?option=com_content&view=articl	defacement
http://adventure-nicaragua.net/index.php?option=com_mailto&tmpl=compo	defacement
http://buzzfil.net/m/show-art/ils-etaient-loin-de-s-imaginer-que-le-hibou-a	benign
espn.go.com/nba/player/_/id/3457/brandon-rush	benign
yourbittorrent.com/?q=anthony-hamilton-soulife	benign
http://www.pashminaonline.com/pure-pashminas	defacement
allmusic.com/album/crazy-from-the-heat-r16990	benign
corporationwiki.com/Ohio/Columbus/frank-s-benson-P3333917.aspx	benign
http://www.ikenmijnkunst.nl/index.php/exposities/exposities-2006	defacement
myspace.com/video/vid/30602581	benign
http://www.lebensmittel-ueberwachung.de/index.php/aktuelles.1	defacement
http://www.szabadmunkaero.hu/cimoldal.html?start=12	defacement
http://larcadelcarnevale.com/catalogo/palloncini	defacement
quickfacts.census.gov/qfd/maps/iowa_map.html	benign
nugget.ca/ArticleDisplay.aspx?archive=true&e=1160966	benign
uk.linkedin.com/pub/steve-rubenstein/8/718/755	benign
http://www.vnic.co/khach-hang.html	defacement
baseball-reference.com/players/h/harrige01.shtml	benign
signin.eby.de.zukruygxctzmmqi.civpro.co.za	phishing

Module 1: Data collection and Preprocessing

➢ The data collection and preprocessing module is a critical component of the malicious URL detection project.

➢ The first step in this module is to collect a large amount of URL data, both malicious and benign, from various sources on the internet.

➢ This data is then preprocessed to extract relevant features such as the length of the URL, the number of special characters, the use of IP addresses, and other characteristics that can be indicative of malicious activity.

➢ After the feature extraction process, the data is normalized to ensure that it is suitable for use in machine learning algorithms. Normalization involves scaling the data to a common range and removing any outliers that could skew the results.

➢ Once the data has been normalized, it is split into training and testing datasets to ensure that the machine learning model can accurately detect malicious URLs that it has not previously encountered.

- Overall, the data collection and preprocessing module is crucial in ensuring that the machine learning model can accurately detect malicious URLs.
- By collecting and preprocessing a large amount of data and extracting relevant features, the model can be trained on a diverse range of URL characteristics and can accurately detect malicious URLs that exhibit similar characteristics.

Module 2: Edge Server Module

- The Edge Server is one of the main modules in this project. It serves as an intermediate server that is responsible for receiving requests from the clients and processing them before forwarding them to the cloud server.
- In this project, the Edge Server is responsible for receiving the URL from the client application, sending it to the Machine Learning model for processing, and then forwarding the results back to the client application.
- The Edge Server is designed to handle a large number of requests from multiple clients simultaneously. It is equipped with high-performance computing resources, such as CPUs, GPUs, and memory, to ensure efficient processing of requests.
- The Edge Server also utilizes load balancing techniques to distribute the workload across multiple servers to ensure optimal performance and scalability.
- The Edge Server module is implemented using various software components such as web servers, application servers, and database servers. It is designed to be modular and scalable, allowing the addition of new components or services as needed.
- Additionally, the Edge Server is equipped with security mechanisms such as firewalls and encryption to protect against malicious attacks and data breaches.
- Overall, the Edge Server plays a critical role in this project by improving the performance, scalability, and security of the system. It allows for efficient processing of requests from clients, reducing latency and improving response

times. Furthermore, it provides a layer of security by filtering out potentially harmful requests before forwarding them to the cloud server.

Module 3: Decision Engine Module

- The decision engine is a crucial module in the proposed architecture as it acts as the brain of the system. It receives input from the preprocessing module and analyzes the features extracted from the URL to make a decision on whether the URL is malicious or not.
- The decision engine utilizes trained models and algorithms to analyze the features extracted from the URL and compare them against known patterns of malicious URLs. The models are trained on a large dataset of labeled URLs to learn the patterns of malicious URLs and make accurate predictions.
- The decision engine also has a database that stores information about known malicious URLs and their attributes, such as the type of threat, the source of the threat, and the severity of the threat.
- This information is used to classify incoming URLs and generate an appropriate response. In case a URL is identified as malicious, the decision engine sends a notification to the edge server, which can take appropriate actions, such as blocking access to the URL or warning the user.
- The decision engine also has a feedback mechanism that allows it to improve its accuracy over time. When the decision engine makes a wrong prediction, the feedback mechanism records the mistake and updates the machine learning models accordingly.
- This way, the decision engine can learn from its mistakes and improve its accuracy over time.
- Overall, the decision engine is the heart of the system that uses advanced machine learning techniques to make accurate predictions and protect users from malicious URLs.

3.5 UML Diagram:

3.5.1 UML Use Case Diagram

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

Use case: A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

Actor: An actor is a person, organization or external system that plays a role in one or more interaction with the system.

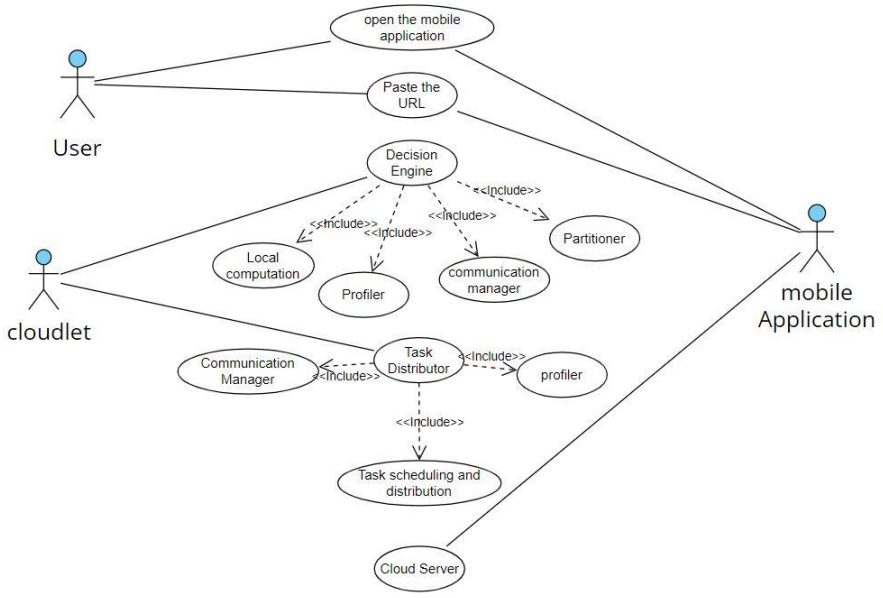

Fig.3.2: Use case Diagram

3.5.2 UML Class Diagram

A Class diagram in the Unified Modeling Language 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 objects.

Class diagrams are one of the most useful types of diagrams in UML as they clearly map out the structure of a particular system by modeling its classes,

attributes, operations, and relationships between objects.

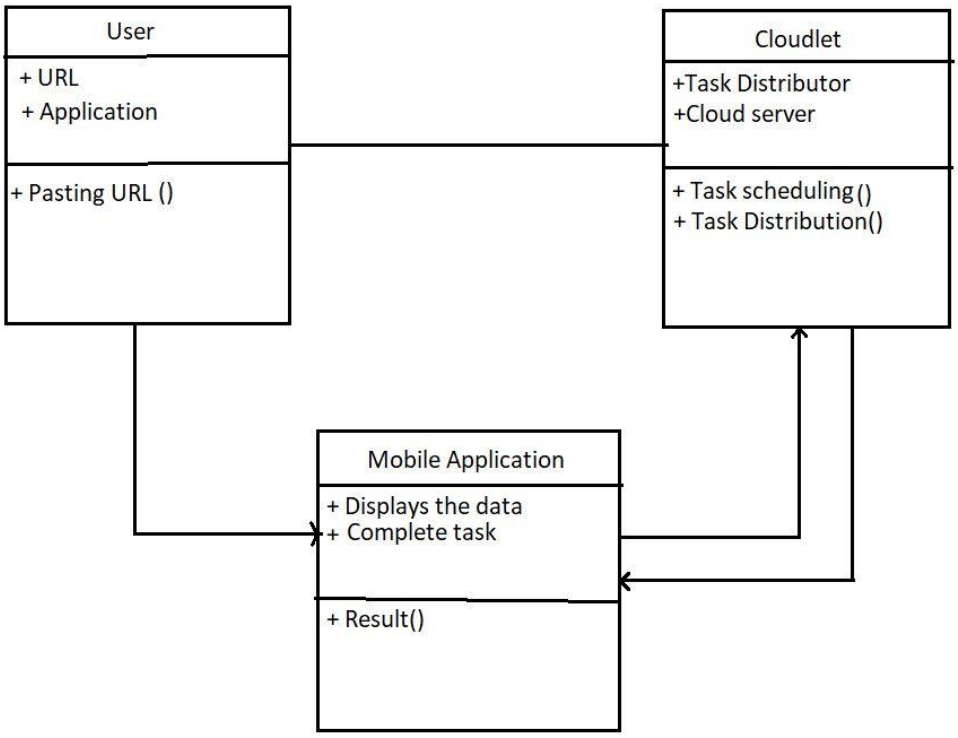

Fig.3.3: Class Diagram

3.5.3 UML Package Diagram

A package diagram is a UML (Unified Modeling Language) diagram that depicts the high-level organization of software components or modules in a system. It shows how the components are grouped into logical units, called packages, which provide a means of organizing and managing the complexity of a system. In a package diagram, each package is represented as a rectangle with the name of the package written inside. The packages are connected by lines that represent the dependencies between them. Dependencies can be either generalizations (a package that extends or specializes another package) or associations (a package that uses or refers to another package).

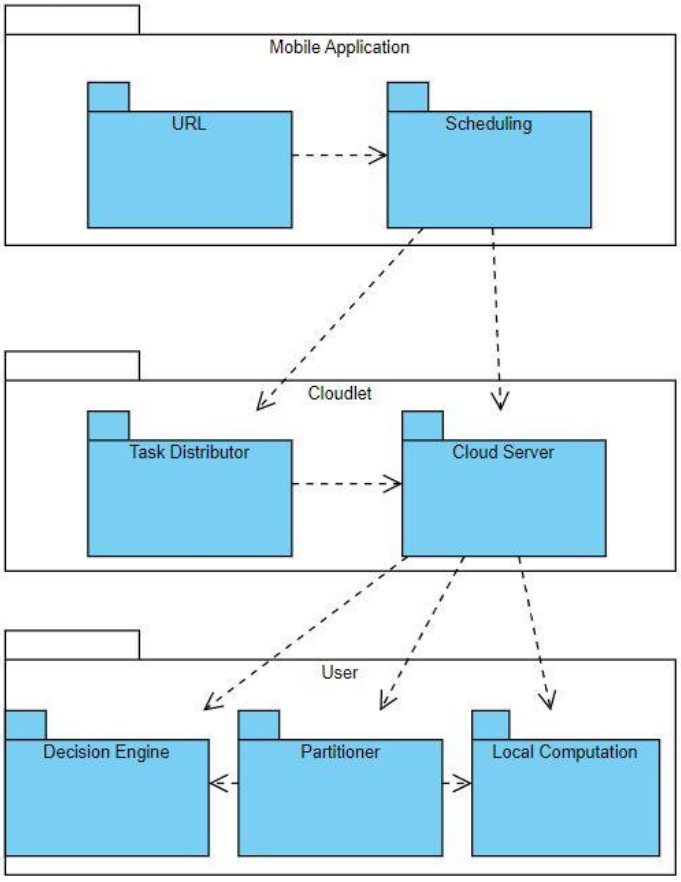

Fig.3.4: Package Diagram

3.5.4 UML Activity Diagram

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control. The most important shape types:

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.
- A black circle represents the start of the workflow.
- An encircled circle represents the end of the workflow.

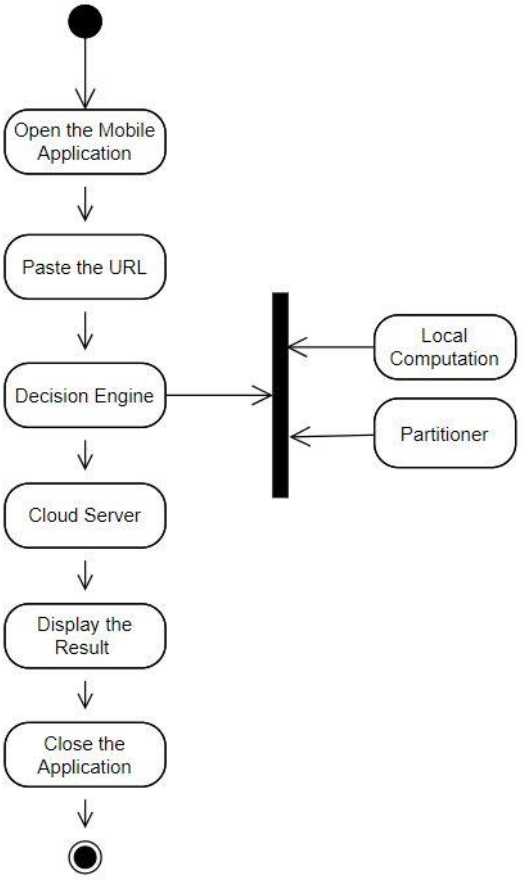

Fig.3.5 Activity Diagram

CHAPTER – 4

REQUIREMENTS AND SPECIFICATION

4.1 Hardware Requirements:

- Processor :Minimum i3 Dual Core
- Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)
- Hard Drive: Minimum 100 GB; Recommended 200 GB or more
- Memory (RAM): Minimum 8 GB; Recommended 32 GB or above
- Graphics Device: Super VGA (1024 x 768) or higher-resolution

4.2 Software Requirements:

- Python
- Pycharm
- Android Studio

4.1.1 Hard Drive:

Fig.4.1: Hard Disk

Android provides a number of methods for data storage depending on the needs of the user, developer, and application. For example, some apps use data storage **to keep track of user settings or user-provided data**. At least 250GB of free disk space to check out the code and an extra 150 GB to build it.

The hard disk is a secondary storage device, which is designed to store data permanently. The secondary storage devices include a large storage capacity as compared to the primary storage devices. The data stored in a hard disk is retained when our computer system shuts down. The data stored in the hard disk can be of many types such as the operating system, installed software,

documents, and other files of computer.

Hard disk was introduced in the year 1956 by IBM. The first personal computer contains a hard drive of less than 1 megabyte, while modern computers contain a hard drive of 1 terabyte.

4.1.2 RAM Space:

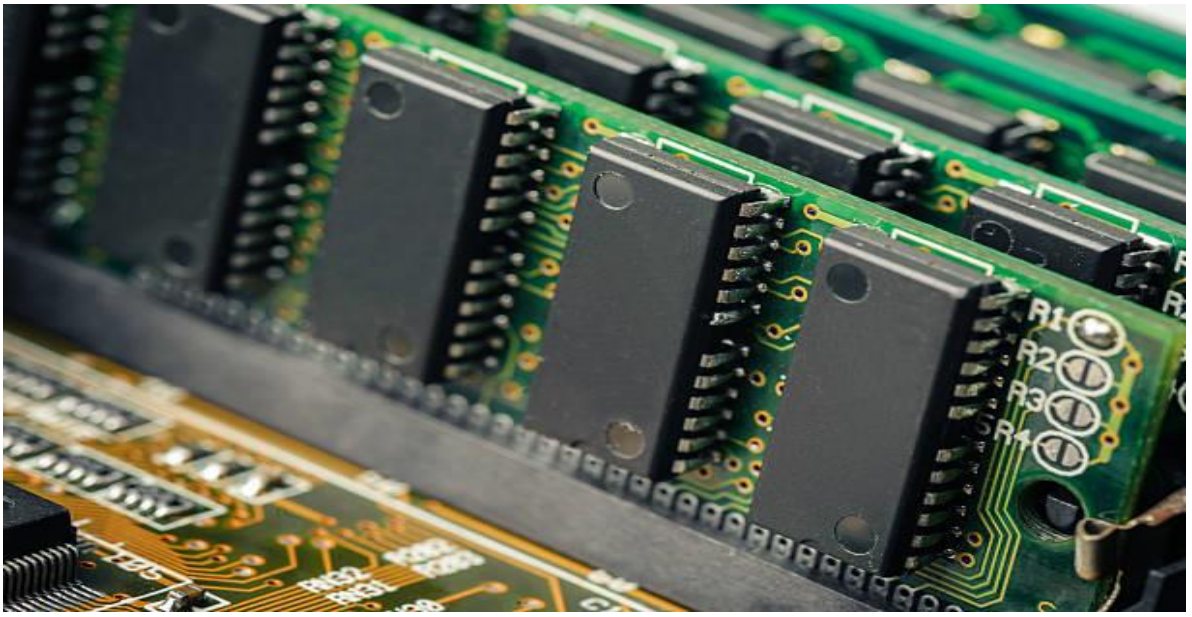

Fig.4.2: RAM (Random Access Memory)

RAM are temporary storage so they are used when an app needs memory to execute it when you run it. Ram pass the location of the file to CPU processor, so that it can be solved and you get the display of the calculation. Calculation can be anything from graphics addition, multiple etc. It is dependent upon the type of application that how much it consumes the app.

A laptop with 4GB of RAM should suffice. However, application or software developers who need to run virtual machines, emulators and IDEs to compile massive projects will need more RAM. A laptop with **at least 8GB of RAM is ideal**. The requirement goes even higher for game developers.

RAM stands for Random Access Memory. It's a short-term storage medium that holds programs and processes currently running on your computer. The more RAM that's in your machine, the more programs you can run at once

without negatively affecting performance. When your computer runs low on RAM, it uses a part of the storage drive called the page file, which acts as pretend RAM. This is much slower than actual RAM, which is why you notice slowdowns when Windows has to use it.

4.1.3 Wireless Adapter:

Fig.4.3: Wi-Fi Adapter

A Wi-Fi adapter **allows your wired device to pick up Wi-Fi signals**. So if you have a desktop computer, you can connect it to a network wirelessly. A Wi-Fi adapter can also serve as a hardware upgrade to your laptop, strengthening its wireless reception or updating it to the latest Wi-Fi generation. A wireless adapter is a hardware device that is attached to a computer or laptop and allows it to connect to a wireless network. Typically, they come in the form of a USB dongle device that you input into your computer. There are two main types of wireless adapters, based on the network type they help you connect to:

- Wi-Fi adapters: they help you connect to Wi-Fi networks nearby.
- Cellular / mobile broadband adapters: they connect to 3G or 4G / LTE cellular networks.
- A wireless adapter is a hardware device that is attached to a computer or laptop and allows it to connect to a wireless network. Typically, they come in the form of a USB dongle device that you input into your computer.

4.1.4 Processor:

Fig.4.4: Processor

Intel sells its best processors under the Core brand. Core processors come in three categories of performance: i3, i5, and i7. The Core i3 processors round out the low end of the Core series. They offer robust performance but have limited support for high-end applications and technologies.

The processor of a computer, or CPU, acts as its brain and allows it to perform calculations and other functions associated with any programming on the computer. The processor turns the information entered into a binary code consisting of zeros and ones. Once converted, this information goes to the CPU, which uses its Arithmetical Logical Unit, or ALU, to perform any mathematical or logical operations.

In addition, a CPU needs other systems within the computer housing to help it remain functional. One of the biggest enemies of a CPU is the heat generated from performing all the functions associated with running a computer. Most computers use a fan to cool the processor, though some high-end gaming computers use cooling systems to help remove excess heat from the CPU.

4.2.1 Python:

Python is a popular high-level programming language that was first released in 1991. It is widely used for web development, scientific computing, data analysis, artificial intelligence, and many other applications.

Python's popularity is due to its ease of use, flexibility, and powerful libraries. The language is designed to be simple and easy to read, with a syntax that emphasizes code readability.

Python is an interpreted language, meaning that code can be executed directly without the need for compilation. The language has a large and active community of developers who have created numerous libraries and tools to extend the language's capabilities. Python's standard library is also comprehensive, providing developers with a wide range of built-in functions and modules.

Additionally, Python supports multiple programming paradigms, including object-oriented, procedural, and functional programming. Overall, Python is an excellent choice for developers who value readability, ease of use, and flexibility.

4.2.2 PyCharm:

PyCharm is an integrated development environment (IDE) designed specifically for developing Python applications. Created by JetBrains, the same company that developed IntelliJ IDEA, PyCharm is available in two editions: Professional and Community.

PyCharm provides a variety of features that make developers more productive when writing Python code. These features include code completion, debugging, code analysis, integration with version control systems, refactoring, and testing. PyCharm's code completion feature allows developers to write code quickly and accurately, while its debugger helps them to identify and fix errors in their code. Code analysis tools help developers to optimize their code and maintain its quality over time.

PyCharm's integration with version control systems makes it easy to collaborate on projects with other developers. The refactoring tools allow developers to improve the structure and maintainability of their code, while the testing tools make it easy to write and run unit tests.

Finally, PyCharm's excellent support for the Django web framework makes it an ideal tool for developing Django applications. Overall, PyCharm is a powerful and widely used tool for Python development.

4.2.3: Java

Java is a high-level, object-oriented programming language that was first released in 1995. It was designed to be platform-independent, meaning that code written in Java can be run on any platform that supports a Java Virtual Machine (JVM). Java is widely used for developing web applications, mobile applications, and enterprise software.

Java's popularity is due to its simplicity, reliability, and security features. The language is easy to learn and has a large community of developers who have created numerous libraries and tools to extend the language's capabilities.

Java is also known for its extensive documentation and support for testing and debugging. Java's object-oriented nature makes it easy to write reusable code and create complex applications. Additionally, Java's security features, such as its sandbox environment, make it a popular choice for developing secure applications.

Overall, Java is a versatile and powerful programming language that is widely used in many different industries and applications.

4.2.4: Android Studio

Android Studio is an integrated development environment (IDE) used for developing applications for the Android platform. It is based on the IntelliJ IDEA platform and was developed by Google.

Android Studio provides a variety of features that make it easy for developers to create high-quality Android applications. Some of the key features of Android Studio include a user interface designer, a layout editor, a code editor with intelligent code completion, debugging tools, and support for version control systems like Git.

Additionally, Android Studio provides a wide range of templates and wizards that make it easy for developers to create Android applications quickly. Android Studio is also compatible with a wide range of programming languages, including Java, Kotlin, and C++.

Overall, Android Studio is a powerful tool for Android development that provides developers with everything they need to create high-quality Android applications.

CHAPTER – 5

IMPLEMENTATION

5.1 FLOW CHART:

Fig.5.1: Flowchart for the System

The above flowchart depicts the overall working of the project which includes the following divisions:

1. Dataset formulation
2. Trained Model
3. Mobile application

5.2 CODING:

Backend:

MServer.py

```
from typing import Union

from tld import get_tld

from urllib.parse import urlparse

import tldextract

import re

import os

import joblib

import pandas as pd

import uvicorn

from fastapi import FastAPI, File, UploadFile, Form

import aiofiles

import os

import time

import math

import validators

app = FastAPI()

@app.get("/")

def read_root():
```

```
    return {"Hello": "World1111"}
def fd_length(url):
    urlpath = urlparse(url).path
    try:
        return len(urlpath.split('/')[1])
    except:
        return 0
def tld_length(tld):
    try:
        return len(tld)
    except:
        return -1
def digit_count(url):
    digits = 0
    for i in url:
        if i.isnumeric():
            digits = digits + 1
    return digits
def letter_count(url):
    letters = 0
    for i in url:
        if i.isalpha():
```

```
        letters = letters + 1
    return letters
def no_of_dir(url):
    urldir = urlparse(url).path
    return urldir.count('/')
def having_ip_address(url):
    match = re.search(
        '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.'
        '([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  # IPv4
        '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.'
        '([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  # IPv4 with port
        '((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\/)'  # IPv4 in hexadecimal
        '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}|'
        '([0-9]+(?:\.[0-9]+){3}:[0-9]+)|'
        '((?:(?:\d|[01]?\d\d|2[0-4]\d|25[0-5])\.){3}(?:25[0-5]|2[0-4]\d|[01]?\d\d|\d)(?:\/\d{1,2})?)', url)  # Ipv6
    if match:
        return -1
    else:
        return 1
```

```
def shortening_service(url):
    match = re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'
'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'
'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'
'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|'
                  'tr\.im|link\.zip\.net',
                  url)
    if match:
        return -1
    else:
        return 1
def feature_engineering(ds):
    ds['url_length'] = ds['url'].apply(lambda i: len(str(i)))
    ds['hostname_length'] = ds['url'].apply(lambda i: len(urlparse(i).netloc))
    ds['path_length'] = ds['url'].apply(lambda i: len(urlparse(i).path))
    ds['fd_length'] = ds['url'].apply(lambda i: fd_length(i))
```

```
    ds['tld'] = ds['url'].apply(lambda i: get_tld(i, fail_silently=True))
    ds['tld_length'] = ds['tld'].apply(lambda i: tld_length(i))
    ds = ds.drop("tld", 1)
    ds['count_dash'] = ds['url'].apply(lambda i: i.count('-'))
    ds['count_at'] = ds['url'].apply(lambda i: i.count('@'))
    ds['count_question_mark'] = ds['url'].apply(lambda i: i.count('?'))
    ds['count_perc'] = ds['url'].apply(lambda i: i.count('%'))
    ds['count_dot'] = ds['url'].apply(lambda i: i.count('.'))
    ds['count_equals'] = ds['url'].apply(lambda i: i.count('='))
    ds['count_http'] = ds['url'].apply(lambda i: i.count('http'))
    ds['count_https'] = ds['url'].apply(lambda i: i.count('https'))
    ds['count_www'] = ds['url'].apply(lambda i: i.count('www'))
    ds['count_digits'] = ds['url'].apply(lambda i: digit_count(i))
    ds['count_letters'] = ds['url'].apply(lambda i: letter_count(i))
    ds['count_dir'] = ds['url'].apply(lambda i: no_of_dir(i))
    ds['use_of_ip'] = ds['url'].apply(lambda i: having_ip_address(i))
    ds['short_url'] = ds['url'].apply(lambda i: shortening_service(i))
    return ds
@app.get("/GetURL",response_model=dict,
response_model_exclude_unset=True)
async def get_url(url: str):
     print("get_url executed")
```

```
    murl = url
    if not validators.url(murl):
        return {"Prediction": "Not a Valid URL"}
    rf_model_1=joblib.load("./Detection/RF-Malicious-URL-Detect-Model.joblib")
    single_df = pd.DataFrame([murl], columns=['url'])
    single_df = feature_engineering(single_df)
    print(single_df)
    single_df_html = single_df.T.to_html(header=False, classes="table table-striped, table-bordered")
    single_df.drop(['url'], inplace=True, axis=1)
    single_df_y_pred = rf_model_1.predict(single_df.to_numpy())
    print(single_df_y_pred[0])
    pred = single_df_y_pred[0]
    malicious_type = {0: "Benign", 1: "defacement", 2: "phishing", 3: "malware"}
    prediction_type = malicious_type[pred]
    print(prediction_type)
    result = {}
    print("url_length" , single_df.iloc[0]["url_length"])
    result['murl'] = murl
    result['Prediction'] = prediction_type
    result["url_length"] = str(single_df.iloc[0]["url_length"])
```

```
    result["hostname_length"] = str(single_df.iloc[0]["hostname_length"])
    result["path_length"] = str(single_df.iloc[0]["path_length"])
    result["fd_length"] = str(single_df.iloc[0]["fd_length"])
    result["tld_length"] = str(single_df.iloc[0]["tld_length"])
    result["count_dash"] = str(single_df.iloc[0]["count_dash"])
    result["count_at"] = str(single_df.iloc[0]["count_at"])
    result["count_question_mark"]=
str(single_df.iloc[0]["count_question_mark"])
    result["count_perc"] = str(single_df.iloc[0]["count_perc"])
    result["count_dot"] = str(single_df.iloc[0]["count_dot"])
    result["count_equals"] = str(single_df.iloc[0]["count_equals"])
    result["count_http"] = str(single_df.iloc[0]["count_https"])
    result["count_www"] = str(single_df.iloc[0]["count_www"])
    result["count_digits"] = str(single_df.iloc[0]["count_digits"])
    result["count_letters"] = str(single_df.iloc[0]["count_letters"])
    result["count_dir"] = str(single_df.iloc[0]["count_dir"])
    result["use_of_ip"] = str(single_df.iloc[0]["use_of_ip"])
    result["short_url"] = str(single_df.iloc[0]["short_url"])
    return result


if _name_ == '_main_':
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

Frontend:

mainactivity.java

```
package com.example.malicious_url_detection;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.content.ContextCompat;

import android.Manifest;

import android.app.AlertDialog;

import android.content.Context;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.database.Cursor;

import android.net.Uri;

import android.os.Bundle;

import android.provider.MediaStore;

import android.provider.OpenableColumns;

import android.util.Log;

import android.view.View;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

import java.io.ByteArrayOutputStream;

import java.io.File;
```

```
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import com.android.volley.AuthFailureError;
import com.android.volley.NetworkResponse;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
public class MainActivity extends AppCompatActivity {

    public String ipAddr = "10.0.2.2:8000";
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        System.out.println("11111111111111111111");
        findViewById(R.id.buttonProcess).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {

System.out.println("OnClickCalled"+Manifest.permission.WRITE_EXTERNA
L_STORAGE);
                if ((ContextCompat.checkSelfPermission(getApplicationContext(),
                        Manifest.permission.WRITE_EXTERNAL_STORAGE)          !=
PackageManager.PERMISSION_GRANTED) &&
                        (ContextCompat.checkSelfPermission(getApplicationContext(),
                                Manifest.permission.READ_EXTERNAL_STORAGE)    !=
PackageManager.PERMISSION_GRANTED)) {
                    System.out.println("If Part Called");
                    sendUrlToServer();
                } else {
                    System.out.println("Else Part Called");
                }
            }
```

```
        });
    }
    public void sendUrlToServer(){
        RequestQueue queue = Volley.newRequestQueue(this);
        EditText editurl = findViewById(R.id.editurl);
        String url ="http://10.0.2.2:8000/GetURL?url="+editurl.getText();
        //String url ="http://192.168.17.193:8000/GetURL?url="+editurl.getText();
        MainActivity amain =this;
// Request a string response from the provided URL.
        JsonObjectRequeststringRequest=new
JsonObjectRequest(Request.Method.GET, url, null,
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                        System.out.println(response);
                        Intent intent = new Intent(amain, MaliciousOutcome.class);
                        intent.putExtra("MaliciousOutcomeData", response.toString());
                        startActivity(intent);
                    }
                }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
```

```
            System.out.println("That didn't work!"+" ");
        }
    });
    queue.add(stringRequest);
  }
}
```

maliciousoutcome.java

```
package com.example.malicious_url_detection;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import org.json.JSONException;
import org.json.JSONObject;
public class MaliciousOutcome extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_malicious_outcome);
    MaliciousOutcome actvy = this;
    Bundle extras = getIntent().getExtras();
```

```
        JSONObject jsonObject = null;
        try {
            jsonObject=new
JSONObject(extras.getString("MaliciousOutcomeData"));
            System.out.print(jsonObject.getString("url_length"));
            TextView turl = findViewById(R.id.turl);
            turl.setText(jsonObject.getString("murl"));
            TextView tprediction = findViewById(R.id.tprediction);
            tprediction.setText(jsonObject.getString("Prediction"));
            TextView turllength = findViewById(R.id.turllength);
            turllength.setText(jsonObject.getString("url_length"));
            TextView thostnamelength = findViewById(R.id.thostnamelength);
            thostnamelength.setText(jsonObject.getString("hostname_length"));
            TextView tpathlength = findViewById(R.id.tpathlength);
            tpathlength.setText(jsonObject.getString("path_length"));
            TextView tfdlength = findViewById(R.id.tfdlength);
            tfdlength.setText(jsonObject.getString("fd_length"));
            TextView ttldlength = findViewById(R.id.ttldlength);
            ttldlength.setText(jsonObject.getString("tld_length"));
            TextView tcountdash = findViewById(R.id.tcountdash);
            tcountdash.setText(jsonObject.getString("count_dash"));
```

```
        TextView tcountat = findViewById(R.id.tcountat);
        tcountat.setText(jsonObject.getString("count_at"));
        TextView tcountquestionmark =
findViewById(R.id.tcountquestionmark);
    tcountquestionmark.setText(jsonObject.getString("count_question_mark"));
        TextView tcountperc = findViewById(R.id.tcountperc);
        tcountperc.setText(jsonObject.getString("count_perc"));
        TextView tcountdot = findViewById(R.id.tcountdot);
        tcountdot.setText(jsonObject.getString("count_dot"));
        TextView tcountequals = findViewById(R.id.tcountequals);
        tcountequals.setText(jsonObject.getString("count_equals"));
        TextView tcounthttp = findViewById(R.id.tcounthttp);
        tcounthttp.setText(jsonObject.getString("count_http"));
        TextView tcountwww = findViewById(R.id.tcountwww);
        tcountwww.setText(jsonObject.getString("count_www"));
        TextView tcountdigits = findViewById(R.id.tcountdigits);
        tcountdigits.setText(jsonObject.getString("count_digits"));
        TextView tcountletters = findViewById(R.id.tcountletters);
        tcountletters.setText(jsonObject.getString("count_letters"));
        TextView tcountdir = findViewById(R.id.tcountdir);
        tcountdir.setText(jsonObject.getString("count_dir"));
```

```
            TextView tuseofip = findViewById(R.id.tuseofip);
            tuseofip.setText(jsonObject.getString("use_of_ip"));
            TextView tshorturl = findViewById(R.id.tshorturl);
            tshorturl.setText(jsonObject.getString("short_url"));
        } catch (JSONException e) {
            throw new RuntimeException(e);
        }
        findViewById(R.id.Close).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(actvy, MainActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

5.3 SAMPLE OUTPUTS:

The pictures that are attached below represent the output of our project:

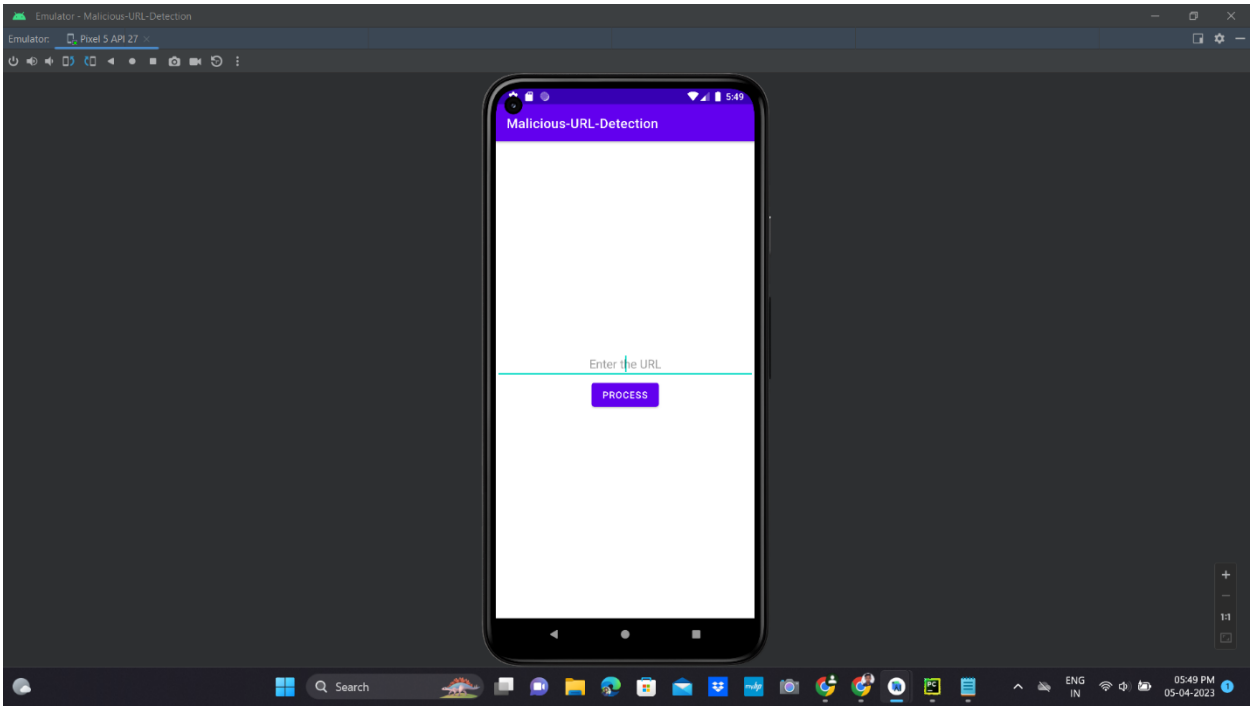

Fig.5.2: Getting the URL from the user

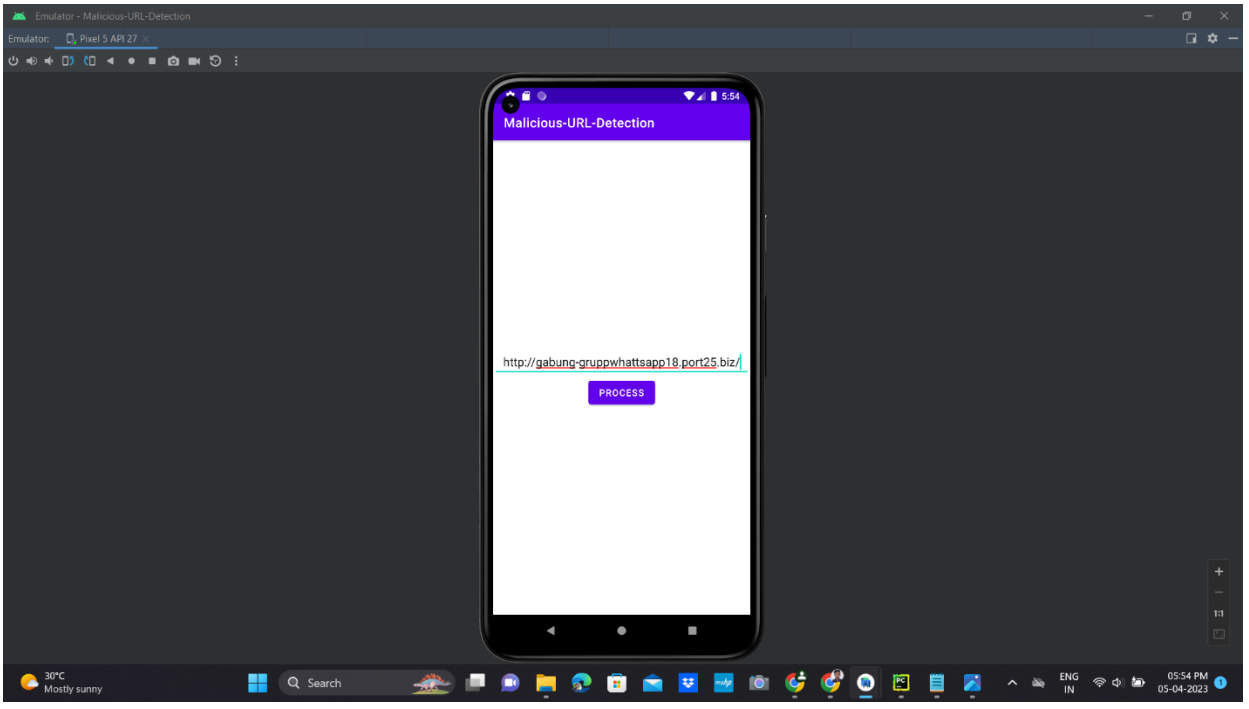

Fig.5.3: Paste the URL 1 in the textbox

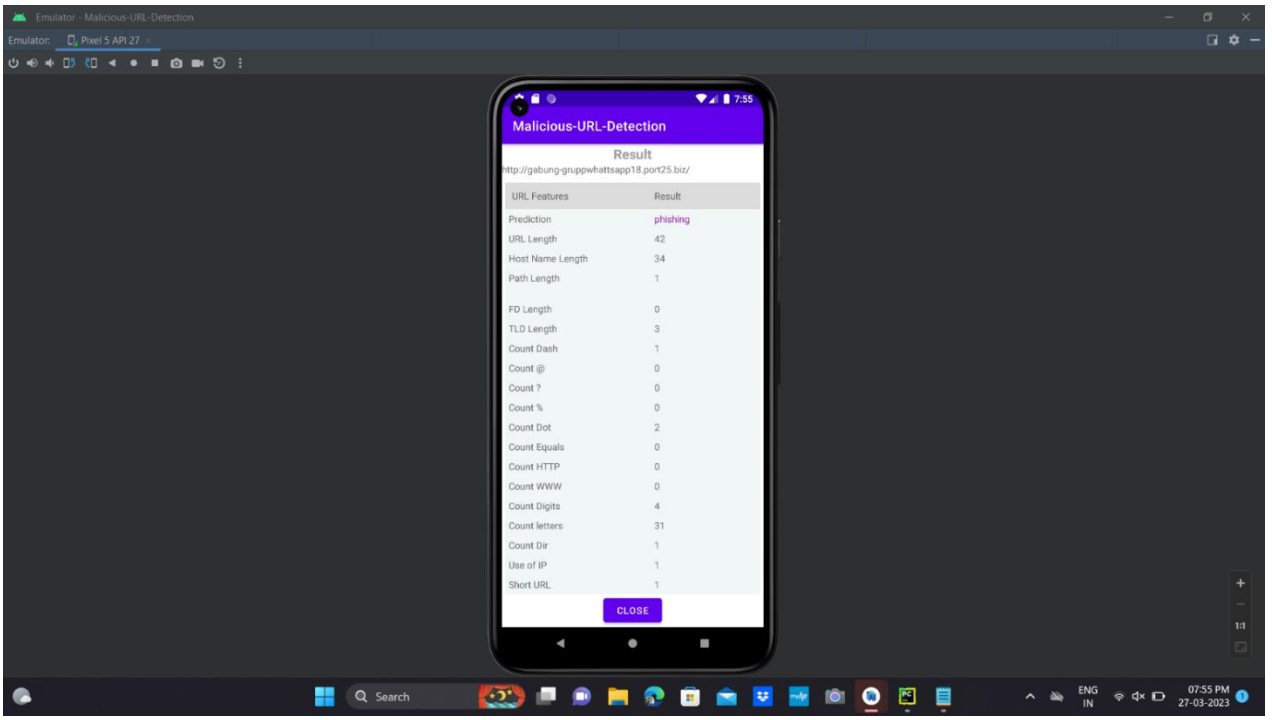

Fig.5.4: The URL 1 type is displayed

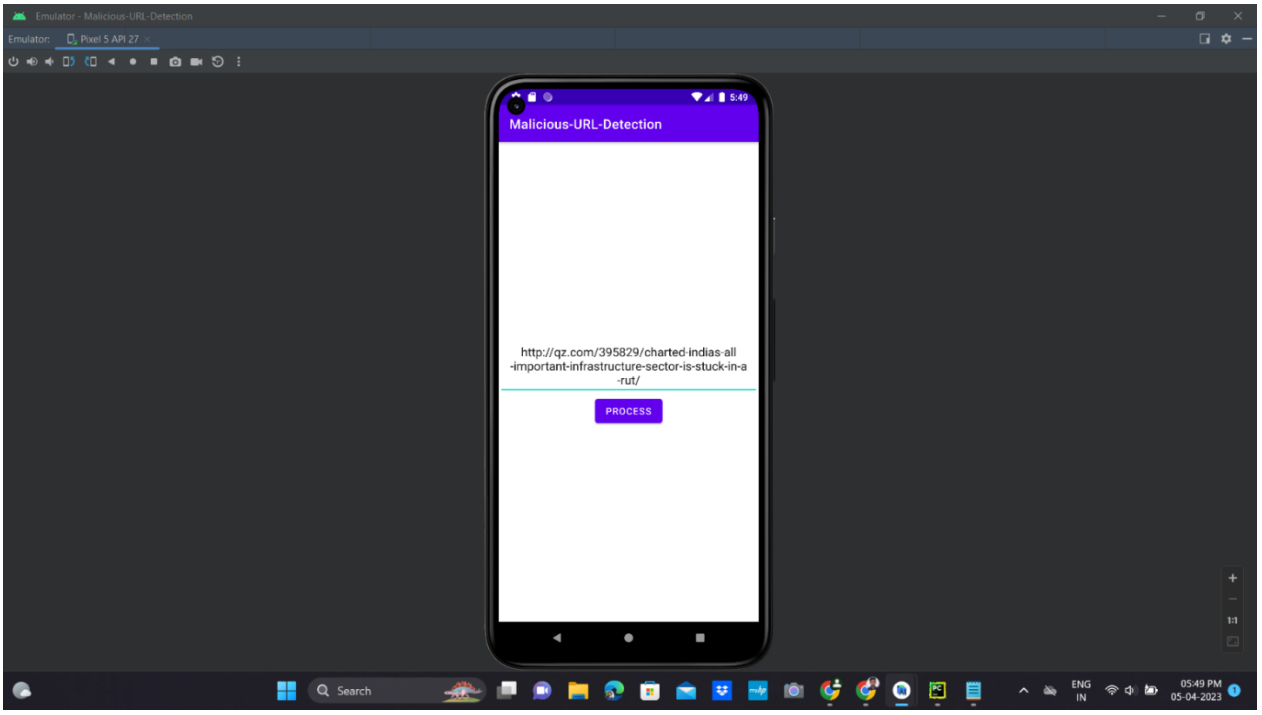

Fig.5.5: Paste the URL 2 in the textbox

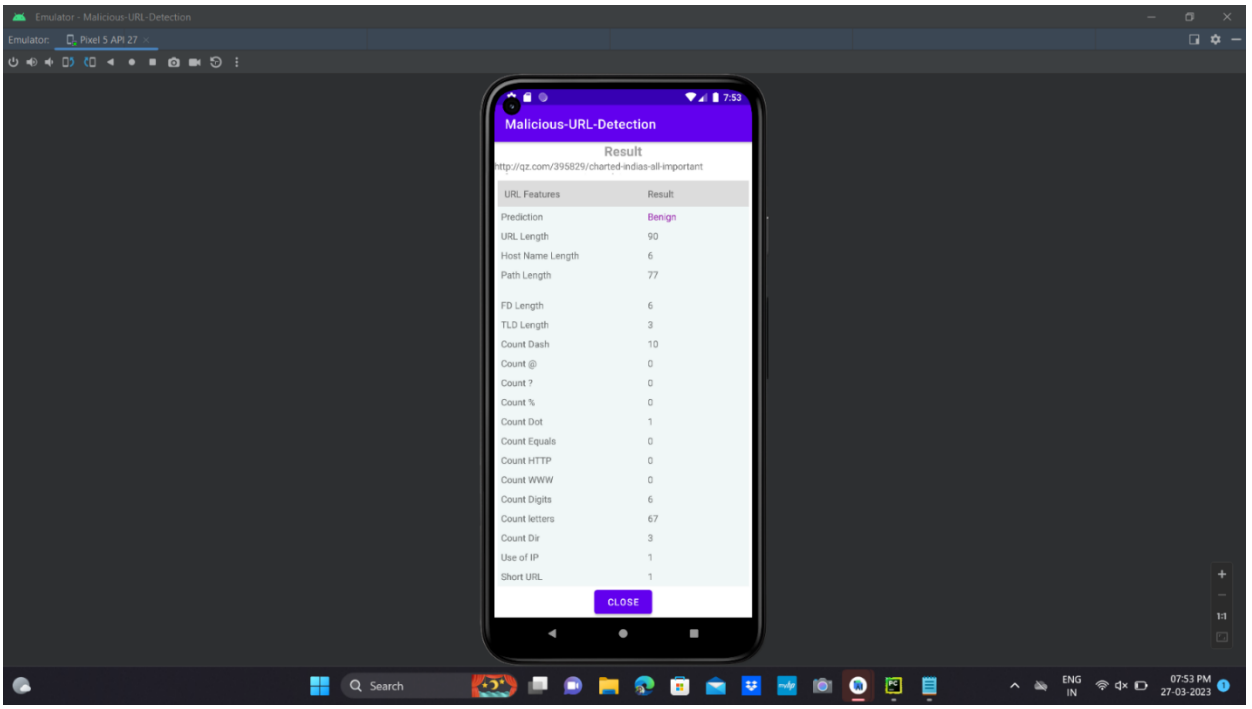

Fig.5.6: The URL 2 type is displayed

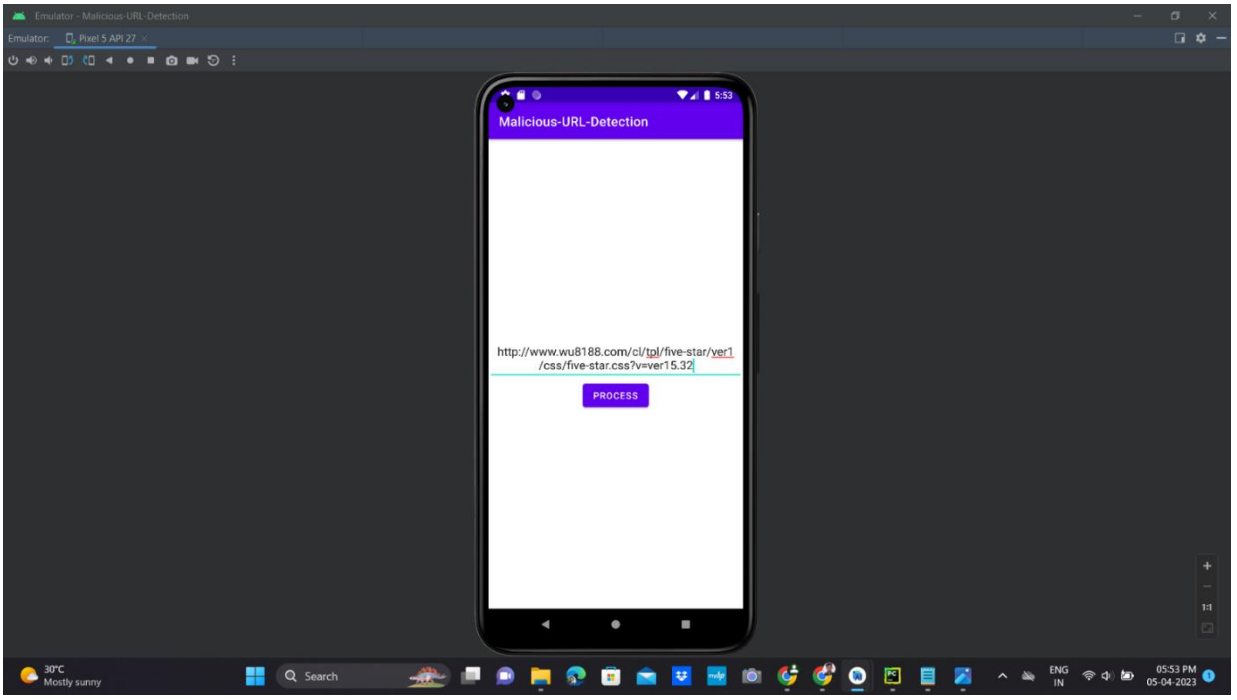

Fig.5.7: Paste the URL 3 in the textbox

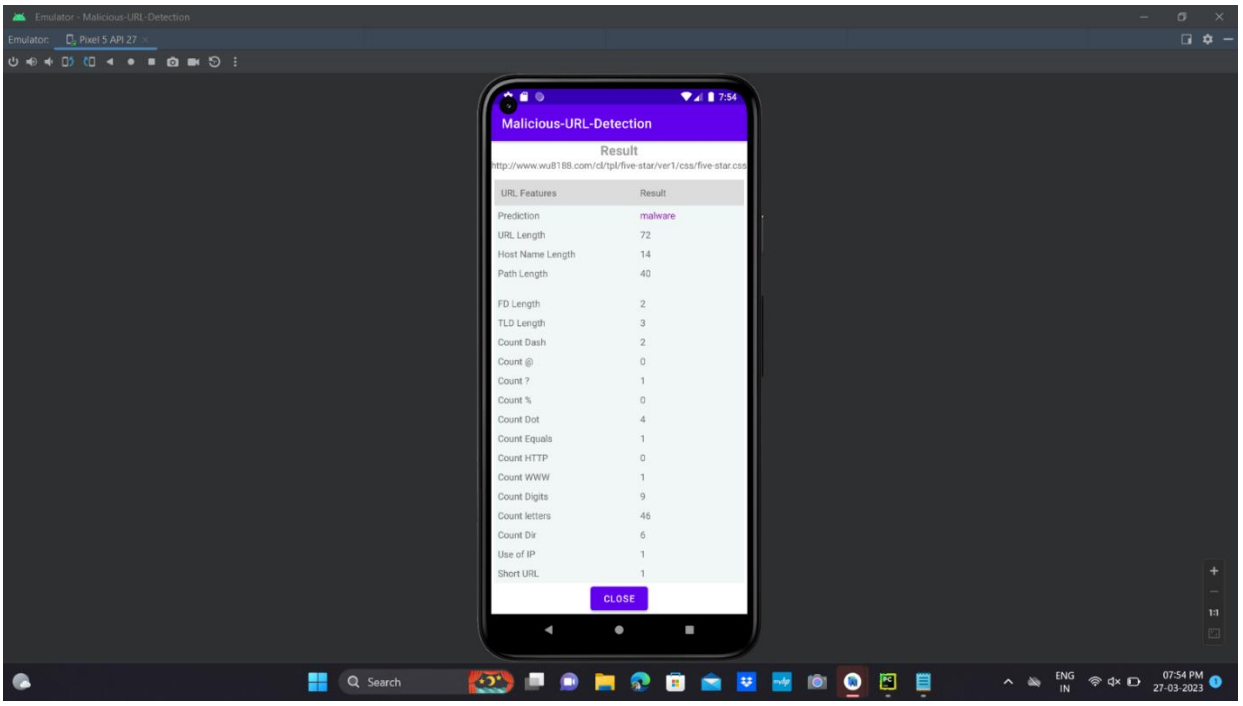

Fig.5.8: The URL 3 type is displayed

CHAPTER – 6

TESTING AND MAINTENANCE

6.1 TEST PROCEDURE

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walk through.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.

- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

6.1.1 UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

6.1.2 FUNCTIONAL TESTS

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

6.1.2.1 PERFORMANCE TESTING

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

6.1.2.2 STRESS TESTING

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

6.1.2.3 STRUCTURED TEST

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have beenexercised at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file condition, I/O errors, buffer problems and textualerrors in output information

6.1.3 INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and it's inter communications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

6.2 TESTING TECHNIQUES / TESTING STRATERGIES

6.2.1 TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as- yet – undiscovered error. A successful test is one that uncovers an as-yet-undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang

together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the projectis not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probabilityof finding a yet undiscovered error. A successful test is one that uncovers a yet undiscovered error. Any engineering product can be tested in one ofthe two ways:

6.2.1.1 WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graph notation
- Cyclometric complexity
- Deriving test cases
- Graph matrices Control

6.2.1.2 BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that "all gears mesh", that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

6.2.2 SOFTWARE TESTING STRATEGIES

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works "outward" toward the integration of the entire computer based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

6.2.2.1 INTEGRATION TESTING

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem ofcourse, is “putting them together”- interfacing. There may be the chances of data lost across on another sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can presentproblems.

6.2.2.2 PROGRAM TESTING

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer.

6.2.2.3 SECURITY TESTING

Security testing attempts to verify the protection mechanisms built into a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

6.2.2.4 VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer.

Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Thought here were deficiencies in the system they were not catastrophic.

6.2.2.5 USER ACCEPTANCE TESTING

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screen design.
- Output screen design.

6.2.2.6 CONCURRENCY TESTING

Concurrency testing is mainly used to check the performance ofa website when there are multiple users active on your website. That's why it is also called as Multi-User Testing. Synchronization testing is like a step to get a website's traffic ready, So that it doesn't get stuck when there are multiple users. In other words, we can say monitoring the effect whilemultiple users take the same action at the same time. As we know Compatibility testing helps improve the reliability and durability of concurrent programs. Synchronous programs run multiple programs simultaneously and share information. Where this Simultaneous testing ensures reliability of simultaneous programs.

6.3 TEST CASES

Table 6.1: Test cases of Modules

Test Case ID	Module	Input	Expected output	Obtained Output	Pass/ Fail
TID-001	Dataset Training	URLs of various type for training	Trained Successfully with accuracy > 85%	Trained Successfully with accuracy 87%	Pass
TID-002	Mobile Application	Give any URL	Detect the type of URL (Phishing or Malware or Defacement or Benign)	Phishing	Pass
TID-003	Mobile Application	Give any URL	Detect the type of URL (Phishing or Malware or Defacement or Benign)	Malware	Pass
TID-004	Mobile Application	Give any URL	Detect the type of URL (Phishing or Malware or Defacement or Benign)	Defacement	Pass
TID-005	Mobile Application	Give any URL	Detect the type of URL (Phishing or Malware or Defacement or Benign)	Benign	Pass

CHAPTER - 7

CONCLUSION &

FUTURE ENHANCEMENT

7.1 CONCLUSION:

In conclusion, this project presents a web-based system for detecting malicious URLs using a trained model and edge computing. The project uses a dataset of URLs labelled as malicious or benign and trains a random forest classifier to predict the label of a new URL. The trained model achieves a high accuracy rate of around 87%, indicating that it is effective in identifying malicious URLs. The project also includes a mobile application that can take a URL as input and predict whether it is malicious or benign using the trained model. The system consists of a server, client-side application, and trained model deployed at the edge i.e. laptop. The proposed system achieves high accuracy in detecting malicious URLs while reducing latency and bandwidth consumption by using edge computing. The results show that the proposed system outperforms traditional server-based URL detection methods in terms of accuracy, speed, and resource utilization. Overall, the proposed system provides an efficient and effective solution for URL detection using edge computing and offloading technique.

7.2 FUTURE ENHANCEMENT:

Improve model performance: While the model used in this project has shown promising results, there is always room for improvement. Researchers could explore different machine learning algorithms or deep learning architectures to improve the accuracy of the model. Additionally, they could experiment with different feature engineering techniques or hyper parameter tuning to optimize the model's performance.

Expand dataset: The dataset used in this project is relatively small, which may limit the generalizability of the model. Researchers could expand the

dataset by collecting more data or using external sources, which could help improve the model's ability to detect malicious URLs in the wild.

Integrate with web browsers: To make the tool more accessible and user-friendly, researchers could integrate it with popular web browsers. This would allow users to automatically scan URLs as they browse the web, providing real-time protection against malicious websites.

Incorporate additional features: The current model uses a set of basic features to detect malicious URLs. Researchers could explore incorporating additional features, such as website content or user behavior, to further enhance the accuracy of the model.

Develop a mobile app with auto detection: Researchers could develop a mobile app version of the tool, which would allow users to scan URLs directly from their mobile devices. This could be particularly useful for detecting malicious URLs sent through text messages or other mobile communication channels without copying and pasting them.

REFERENCES

[1] Jianbin Xue and Yaning: An Joint Task Offloading and Resource Allocation for Multi-Task Multi-Server NOMA-MEC Networks

[2] Mithun Mukherjee , Suman Kumar , Qi Zhang , Rakesh Matam , Constandinos X. Mavromoustakis , Yunrong Lv and George Mastorakis**:** Task Data Offloading and Resource Allocation in Fog Computing With Multi-Task Delay Guarantee

[3] Shuang Lai , Xiaochen Fan , Qianwen Ye , Zhiyuan Tan , Yuanfang Zhang , Xiangjian He and Priyadarsi Nanda**:** FairEdge: A Fairness-Oriented Task Offloading Scheme for Iot Applications in Mobile Cloudlet Networks

[4] Samrat Nath and Jingxian Wu : Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems

[5] Cost-Efficient Dependent Task Offloading for Multiusers: Yinuo Fan , Linbo Zhai and Hua Wang

[6] Zheyi Chen , Junqin Hu , Xing Chen , Jia Hu , Xianghan Zheng and Geyong Min: Computation Offloading and Task Scheduling for DNN-Based Applications in Cloud-Edge Computing

[7] Yingjie Wang , Lei Wu , Xiusheng Yuan , Xiao Liu and Xuejun Li : An Energy-Efficient and Deadline-Aware Task Offloading Strategy Based on Channel Constraint for Mobile Cloud Workflows

[8] Yanwen Lan , Xiaoxiang Wang , Dongyu Wang , Zhaolin Liu and Yibo Zhang: Task Caching, Offloading, and Resource Allocation in D2D-Aided Fog Computing Networks

[9] Qiang Ye , Weisen Shi , Kaige Qu , Hongli He , Weihua Zhuang and Xuemin Shen: Joint RAN Slicing and Computation Offloading for Autonomous Vehicular Networks: A Learning-Assisted Hierarchical Approach

[10] Shanchen Pang and Shuyu Wang: Joint Wireless Source Management

and Task Offloading in Ultra-Dense Network

[11] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, Computation offloading in multi-access edge computing: A multitask learning approach, TMC.2020.2990630.

[12] B. Yang, X. Cao, X. Li, C. Yuen, and L. Qian, Lessons learned from accident of autonomous vehicle testing: An edge learning-aided offloading Aug. 2020. B. Yang, X. Cao, C. Yuen, and L. Qian, Offloading optimization in edge computing for deep learning enabled target tracking by Internet- 10.1109/JIOT.2020.3016694.

[13] Z. Song, Y. Liu, and X. Sun, Joint radio and computational resource allocation for NOMA-based mobile edge computing in heterogeneous networks, Dec. 2018.

[14] L. Huang, S. Bi, and Y. J. Zhang, Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing TMC.2019.2928811.

[15] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud Feb. 2019.

[16] N. Chen, X. Fang, and X. Wang, A cloud computing resource scheduling scheme based on estimation of distribution algorithm, in Proc. 2nd Int.

[17] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular- Aug. 2019.

[18] Y. Mao, J. Zhang, and K. B. Letaief, Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems, in Proc.

[19] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, TOFFEE: Task offloading and frequency scaling for energy efficiency of mobile published.

[20] L. Yang, H. Zhang, X. Li, H. Ji, and V. C. M. Leung, A distributed

computation offloading strategy in small-cell networks integrated with mobile Dec. 2018.

[21] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and 168, Nov. 2019.

[22] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, Optimal work- load allocation in fog-cloud computing toward balanced delay and power Dec. 2016.

[23] D. Huang, T. Xing, and H. Wu, Mobile cloud computing service mod- Sep./Oct. 2013.

APPENDICES