

ANDROID OFFLOADING USING CLOUD

A major project report submitted in partial fulfilment of the requirement
for the degree of

**Bachelor of Technology in
Computer Science and Engineering**

Submitted By

Shivansh Goyal (201250)

Ishan Mehta (201245)

Under the guidance and supervision of

Dr. Hari Singh Rawat



**Department of Computer Science & Engineering and
Information Technology**

**Jaypee University of Information Technology, Wagnaghat
173234, Himachal Pradesh, INDIA**

CERTIFICATE

This is to certify that the work which is being presented in the project report titled “ANDROID OFFLOADING USING CLOUD” in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by “**Shivansh Goyal(201250)**”, “**Ishan Mehta(201245)**.” during the period from August 2023 to May 2024 under the supervision of **Dr. Hari Singh Rawat**, Assistant Professoor(SG) Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Submitted by:

Mr. Shivansh Goyal
(201250)

Mr. Ishan Mehta
(201245)

The above statement made is correct to the best of my knowledge.

Dr. Hari Singh Rawat

Assistant Professor(SG)

**Department of Computer Science & Engineering and Information Technology Jaypee
University of Information Technology, Waknaghat,**

DECLARATION

We hereby declare that the work presented in this report entitled '**ANDROID OFFLOADING USING CLOUD**' in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Wakanaghat is an authentic record of our own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Hari Singh Rawat** (Assistant Professor (SG), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Shivansh Goyal

(201250)

Ishan Mehta

(201245)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Hari Singh Rawat

Assistant Professor (SG)

Department of Computer Science & Engineering and Information Technology Jaypee University of Information Technology

ACKNOWLEDGEMENT

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible for us to complete the project work successfully.

We are really grateful and wish our profound indebtedness to Supervisor **Dr. Hari Singh Rawat**, (Assistant professor (SG), Department of CSE Jaypee University of Information Technology, Wagnaghat). Deep Knowledge & keen interest of our supervisor in the field of “**Cloud**” to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

We would like to express our heartiest gratitude to **Dr. Hari Singh Rawat** (Assistant Professor (SG), Department of CSE) for their kind help to finish my project.

We would also generously welcome each one of those individuals who have helped us straight forwardly or in a roundabout way in making this project a win. In this unique situation, We might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking.

Finally, We must acknowledge with due respect the constant support and patience of our parents.

Shivansh Goyal
(201250)

Ishan Mehta
(201245)

Table of contents

CERTIFICATE	I
DECLARATION	II
ACKNOWLEDGMENT	III
LIST OF ABBREVIATIONS	VI
LIST OF FIGURES	VII
ABSTRACT	VIII
1 INTRODUCTION	1
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Objectives	3
1.4 Significance and motivation of the project report	4
2 LITERATURE SURVEY	6
2.1 Overview of relevant literature	6
2.1.1 A summary of the relevant papers	6
2.2 Key gaps in the literature	17
3 System Development	18
3.1 Requirements and Analysis	18
3.1.1 Functional Requirements	18
3.1.2 Non-Functional Requirements	18
3.1.3 Hardware Requirements	20
3.1.4 Software Requirements	21
3.2 Project Design and Architecture	23
3.2.1 Methodology	24
3.3 Implementation	31
3.4 Key Challenges	38

4	Testing	39
4.1	Testing Strategy	39
5	Results and Evaluation	40
5.1	Results	40
6	Conclusions and Future Scope	43
6.1	Conclusion	43
6.2	Future Scope	44
	REFERENCES	45

List of Abbreviations

Abbreviation	Name
MCC	Mobile cloud computing
OCR	Optical character recognition
API	Application programming interface
SDK	Software Development Kit
RAM	Random Access memory
UI	User Interface
CPU	Central processing unit
JDK	Java Development Kit
JRE	Java Runtime Enviroment
VS	Visual Studio

List of Figures

Fig. No.	Figure	Page No.
1	Project Design	23

ABSTRACT

In the contemporary landscape dominated by smartphones, the performance demands imposed by resource-intensive applications pose a significant challenge. This project addresses this challenge through the strategic synergy of Android and concept of cloud computing, aiming to optimize the operational efficiency of applications through offloading computational tasks.

The envisioned system orchestrates a seamless dialogue between Android devices ,orchestrating a discerning approach to task allocation. By intelligently discerning between tasks suitable for local processing and those meriting cloud-based computation, the project seeks to enhance the overall responsiveness and computational capacity of Android applications.

Integral to this endeavor are robust communication protocols, sophisticated task partitioning algorithms, and a steadfast commitment to data security and privacy. Additionally, the project undertakes a meticulous analysis of network latency and bandwidth considerations, seeking to strike an optimal equilibrium between computational offloading and communication overhead.

This research not only contributes to the evolving domain of mobile computing but also offers practical insights for developers, researchers, and industry stakeholders vested in harnessing the potential of cloud-based offloading within the Android ecosystem. By furnishing a comprehensive examination of the benefits and challenges inherent in Android offloading, this project establishes a foundation for future advancements in mobile computing paradigms.

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

In a digital age dominated by smartphones, the search for optimal performance within resource-intensive applications is more pertinent than ever. This project introduces a dynamic collaboration between Android devices and concept of cloud computing, offering a strategic solution to enhance computational efficiency.

As our reliance on smartphones grows, so does the need for innovative approaches to improve device performance. This project positions itself at the forefront of mobile computing and cloud technology, aiming to improve the execution of resource-demanding applications through the distribution of computational tasks.

The central idea revolves around the relationship between Android devices and cloud infrastructure, wherein computational offloading becomes a strategic method to alleviate local hardware constraints. By offloading computational tasks we can run high resource demanding application.

The core idea of this project is that the computational tasks will be offloaded to some other device , virtual device or cloud . That task will be done there and result will get back to original device.

This project addresses the immediate challenges presented by resource-intensive applications in today's world.

1.2 PROBLEM STATEMENT

Within the dynamic landscape of mobile computing, smartphones have seamlessly integrated into our daily lives. Despite their extensive use and continually improving features, Android devices face intrinsic limitations in terms of computational power, memory, and battery life. Applications that require significant resources, such as real-time data processing, machine learning, or intricate graphics rendering, often encounter performance issues when executed solely on mobile hardware. This not only results in slower response times and increased latency, negatively impacting user experience, but also hastens battery drain, reducing the device's operational duration.

A viable solution to these challenges is computational offloading, where intensive tasks are shifted from the mobile device to more powerful remote servers.

The successful implementation of computational offloading in Android applications has the potential to revolutionize mobile computing by overcoming the inherent limitations of mobile devices. This project seeks to address the critical challenges of latency, security, energy consumption, and integration, paving the way for more powerful and efficient mobile applications.

1.3 OBJECTIVES

Strategic Framework Development:

Create an efficient framework for offloading computational tasks from Android, emphasizing seamless integration and strategic distribution.

Intelligent Task Management:

Develop sophisticated task partitioning algorithms to intelligently distribute workloads between local and cloud processing, optimizing overall system performance.

Network Latency Mitigation:

Investigate and apply strategies to mitigate the impact of network latency, striking an optimal balance between computational distribution and communication overhead.

Scalable System Architecture:

Design a scalable system architecture for seamless integration with diverse Android devices, accommodating varying computational demands and evolving workload requirements.

User-Centric Experience Enhancement:

Focus on improving the end-user experience by minimizing response times, extending battery life, and optimizing resource utilization through user-centric offloading strategies.

Developing a Modular Framework:

Building a modular and scalable framework that can be easily integrated into various Android applications without significant modifications to existing codebases.

1.4 SIGNIFICANCE AND MOTIVATION OF THE PROJECT

Performance Boost:

Swift response times and optimized system performance through intelligent offloading of computational tasks .

Battery Efficiency:

Potential extension of device battery life by smartly shifting computational burdens to the cloud or nearby device.

Scalability and Adaptability:

A flexible system architecture accommodating diverse Android devices and evolving computational demands.

Security and Privacy Focus:

Robust measures to fortify data security and privacy during offloading, enhancing overall system trustworthiness.

Network Optimization:

Efficient network resource utilization through strategic management of latency and communication overhead.

User-Centric Excellence:

Enhanced user experience with reduced response times, prolonged battery life, and personalized offloading strategies.

Research Contribution:

Valuable insights contributing to the evolution of Android offloading technologies and methodologies.

Technological Innovation:

Pioneering solutions addressing challenges in mobile computational offloading.

Real-World Adaptation:

Rigorous testing under diverse scenarios ensures reliability in varied network conditions.

CHAPTER 02: LITERATURE SURVEY

2.1 Overview of relevant Literature

A Survey of Computation Offloading for Mobile Systems[1]

The paper discusses computation offloading in mobile systems, detailing the historical research evolution, the dual aims of improving performance and saving energy, and key factors influencing these goals. It emphasizes conditions favoring each purpose, such as heavy computation, fast servers, and minimal data exchange for performance, and low mobile system power, efficient data transmission, and idle power consideration for energy savings.

The role of enabling technologies like wireless networks, mobile agents, and cloud computing is highlighted, with a conclusion emphasizing offloading's crucial role in balancing demanding applications with mobile resource limitations.

Proposed Method:

1. Introduces a condition for performance improvement based on task characteristics, mobile system speed, server speed, data exchange, and bandwidth.
2. Emphasizes energy constraints in mobile systems and the potential of offloading to extend battery life.
3. Discusses the advantages and challenges of static and dynamic offloading decisions during program development or execution.
4. Investigate decision-making in computation offloading for improved performance and energy savings.

Algorithm Used:-

$$\frac{w}{s} > \frac{d}{B} + \frac{w}{s}$$

Here-

W - is the amount of computation

d - input data

B – Bandwidth

S – server speed

Offloading Android Applications to the Cloud without Customizing Android[2]

The research paper introduces a novel framework designed for Android smartphones to address resource constraints by transferring compute-intensive tasks to a cloud-based Android virtual machine. This innovative framework segregates front-end and back-end tasks, leveraging Android's architecture and AIDL for redirection. It builds upon earlier research on Virtual Smartphone over IP and incorporates a deployment strategy comprising a developer's assistant tool, a user's service offloader, and cloud-based Virtual Smartphones.

Evaluation metrics encompass total response time, energy consumption, and remaining battery life. Noteworthy limitations involve static offloading decisions and the absence of support for NDK and system services. The proposed framework enhances the authors' previous work by improving GUI responsiveness and enabling offline execution.

Proposed Method

1. Design a robust framework architecture that effectively separates front-end and back-end tasks. Utilize Android's architecture and Android Interface Definition Language (AIDL) for efficient redirection of tasks between the local device and the cloud-based virtual machine.
2. Develop a sophisticated task offloading mechanism that determines which tasks should be offloaded to the cloud-based virtual machine. Consider factors such as task complexity, resource availability on the local device, and network conditions for optimal decision-making.
3. Improve upon the authors' previous work by specifically focusing on enhancing GUI responsiveness and introducing support for offline execution. Implement mechanisms to ensure a seamless user experience even when the device is not connected to the cloud.
4. Define and incorporate performance metrics to evaluate the effectiveness of the framework. Key metrics include total response time, energy consumption, and remaining battery life. Develop algorithms to dynamically adjust task offloading decisions based on real-time performance feedback.

An Android-based Application for Computation Offloading in Mobile Cloud Computing[3]

The research paper centers on addressing the challenge of computation offloading within the context of mobile cloud computing (MCC). Computation offloading pertains to the delegation of tasks from devices with limited resources to more robust mobile devices. The authors introduce an Android-centric application named ClientFramework, designed to facilitate the offloading of tasks from client devices. Additionally, they create an OffloadingServer application, serving as an intermediary between service providers and receivers. This application manages the processing of offloaded requests and the transmission of results back to the originating devices.

Proposed Method:-

Android-Based Applications:-

- **ClientFramework:** Developed for mobile devices, it allows users to offload computation-intensive tasks. Users can act as either Service Providers or Service Receivers but not both simultaneously.
- **OffloadingServer:** Developed in Java, it acts as an interface between Service Providers and Service Receivers. It processes offloaded requests, manages service offers, and allocates tasks to available Service Providers.

System Components:-

- **Service Providers:** Devices with high computing capacity that can be used to perform computationally intensive tasks for Service Receivers.
- **Service Receivers:** Devices with limited computing power that offload tasks to Service Providers for more efficient execution.
- **Offloading Server:** Facilitates communication between Service Providers and Service Receivers, manages offloading requests, and allocates tasks using the proposed auction-based mechanism.

Task Computation time:

$$T = \frac{c}{w} + \frac{(d_{in} + d_{out})}{v}$$

Here:

d_{in} – Input data size

d_{out} – Output data size

The greedy offloading mechanism

Input:

- Task offloading requests (r_i) and service offers (s_i) with processing power, bid (b_i), and available time.

Auction Mechanism:

- Greedy auction at discrete time intervals.

Sorting:

- Sort r_i in descending order of c_i/d_i (computation/deadline).
- Sort s_i in ascending order of b_i/w_i (bid/processing speed).

Resource Allocation:

- Process requests in the order of arrival.
- Allocate based on conditions:
 1. r_i completes on s_j before r_i 's deadline.
 2. r_i completes on s_j before s_j 's available time ends.
 3. Offloading r_i to s_j is faster than local completion.

Bid's Role:

- Bid (b_i) influences sorting and helps prioritize cost-effective service offers.

Output:

- Return the final allocation of service offers to task offloading requests.

A transparent code offloading technique for Android devices[4]

The research paper introduces a method for seamlessly integrating code offloading functionalities into Android devices within the domain of Mobile Cloud Computing (MCC). The objective is to improve the performance of compute-intensive applications and mitigate the energy consumption of mobile devices. Here is an overview of the main highlights outlined in the paper:

Introduction:

- Mobile devices, despite being resource-poor, are expected to offer performance and functionality comparable to desktops.
- Compute-intensive applications, requiring powerful processors and abundant resources, face limitations on mobile devices.
- Mobile Cloud Computing (MCC) addresses these constraints by offloading local computation to more resource-rich computers, such as cloud infrastructure or remote servers.

Proposed Technique:

- The paper introduces a transparent code offloading technique for Android devices. Unlike some existing techniques, this proposal does not require modifications to the Android system firmware or application source code.
- The Xposed Framework is utilized to transparently modify Android Framework methods, allowing tasks to be executed remotely without altering the applications.

Towards Computational Offloading in Mobile Device Clouds[5]

The research paper focuses on the concept of Mobile Device Clouds (MDCs) and explores the benefits of computational offloading among a set of mobile devices within this environment. Here is a breakdown of the key components and contributions of the paper:

Emulation Testbed:

- Implements a testbed to quantify potential gains in execution time and energy consumption through offloading to an MDC.
- Evaluates five communication technologies (Bluetooth 3.0, Bluetooth 4.0, WiFi Direct, WiFi, and 3G).
- Shows up to 80% and 90% savings in time and energy, respectively, compared to offloading to the cloud.

MDC Experimental Platform:

- Develops an experimental platform for evaluating MDC-based solutions.
- Measures energy consumption on mobile devices while performing tasks using different communication technologies.
- Creates an Android-based mobile application for customizable and generic offloading.

Social-Based Offloadee Selection Algorithms:

- Addresses the challenge of selecting offloadee devices in MDCs.
- Proposes social-based algorithms utilizing contact history, friendship relationships, and common interests.
- Evaluates algorithms using real data sets from a conference setting.

Offloadee Selection Algorithm:

Algorithm 1 forward(u , T_t , subtasks, Type)

Require: subtasks > 0

Require: Neighbours $\neq 0$

1: function SELECTBESTCANDIDATE(Type, Neighbours) ->

 select the best Type-based neighbour candidate

2: end function

```
3: for subtasks  $\geq 0$  do
4:    $v \leftarrow \text{SELECTBESTCANDIDATE}(\text{Type}, \text{Neighbours})$ 
5:    $\text{Send}(v, Tt/\text{subtasks})$ 
6:    $\text{Neighbours} \leftarrow \text{Neighbours} - \{vk\}$ 
7:    $\text{subtasks} = \text{subtasks} - 1$ 
8: end for
```

Adaptive Computation Offloading from Mobile Devices into the Cloud[6]

The paper presents Mobile Augmentation Cloud Services (MACS), a middleware crafted to tackle the constraints posed by limited processing power and battery life in mobile phones, which often hinder the execution of computationally intensive applications. MACS facilitates the adaptive extension of Android application execution from a mobile client into the cloud, providing a remedy for resource limitations. This middleware manages adaptive application partitioning, resource monitoring, and computation offloading, enabling elastic mobile applications to effortlessly leverage remote computing resources. The evaluation results showcase noteworthy energy savings and performance enhancements, particularly for applications involving intricate algorithms and substantial computations.

Proposed method:-

- MACS middleware enables the execution of elastic mobile applications, dynamically adjusting the partitioning between the device and the cloud during runtime.
- MACS integrates with the established Android application model.
- Applications are structured using Android services pattern: core application (GUI, sensors) and offloadable services.
- Metadata for each service includes type, memory cost, code size, and dependency information.

The cost function is represented :

$$\text{Min} (C_{\text{transfer}} * W_{\text{tr}} + C_{\text{memory}} * W_{\text{mem}} + C_{\text{cpu}} * W_{\text{cpu}})$$

Where

C_{transfer} - Transfer cost

C_{memory} – Memory cost of mobile device

C_{cpu} – CPU cost on mobile device

W – cost of each weight

Process Offloading from Android Device to Cloud Using JADE[7]

The research paper explores issues related to power consumption, restricted processing capabilities, and storage limitations in mobile devices. Emphasis is placed on the offloading of data, applications, processes, and services to the cloud to leverage its processing power. However, the dependence on cloud providers for security introduces apprehensions. The suggested solution involves a mobile agent-based framework utilizing JADE (Java Agent DEvelopment Framework) to facilitate the secure transmission of data between remote nodes. Mobile agents autonomously migrate bundled state and code to a cloud environment without user intervention, bolstering security through intelligent behaviors designed to detect tampering.

Proposed method:-

- Utilize JADE (Java Agent Development Framework) for a mobile agent-based solution.
- Leverage mobile agents to facilitate the autonomous migration of bundled code and state between mobile devices and the cloud.
- Enable agents to autonomously detect tampering by malicious hosts during data transmission and execution.
- Bundle data, including code, instructions, and protocols, into mobile agents.
- Utilize JADE technology to encapsulate and transmit the bundled data between remote nodes.
- Allow mobile agents to migrate between cloud hosts autonomously.
- Implement a mechanism to save the current state of data, re-bundle, and migrate to alternate remote containers upon detection of security threats.

Literature Review Table

S. No	Paper Title [Cite]	Journal/Conference (Year)	Tools/Techniques/Dataset	Results	Limitations
1	A Survey of Computation Offloading for Mobile Systems	10 April 2012 © Springer Science+Business Media, LLC 2012	Java and XML, Virtualization	Different types of algorithms used to partition and offload programs.	Less recent data
2	Offloading Android Applications to the Cloud without Customizing Android	Eighth IEEE PerCom Workshop on Pervasive Wireless Networking 2012, Lugano (23 March 2012)	Android's AIDL (Android Interface Definition Language)	Reduction in time and energy to compute data	Dynamic Offloading Decision should be preferred.
3	An Android-based Application for Computation Offloading in Mobile Cloud Computing	Proceedings of ACM/CSI/IEEE CS Research & Industry Symposium on IoT Cloud For Societal Applications (IoTCloud'21)	Android-based application ClientFramework, OffloadingServer, computational algorithm	Proposed offloading mechanism will improve task execution time by a considerable amount.	More focus on android than offloading

4	A transparent code offloading technique for Android devices	2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)	Xposed Framework, virtual machine, Android 6.0.1	Gains up to 70% in terms of performance	Lack of decision algorithm
5	Towards Computational Offloading in Mobile Device Clouds	2013 IEEE International Conference on Cloud Computing Technology and Science	Android application , Java Sockets , traffic shaper	50% gain in time and 26% gain in energy	Lot of physical infrastructure is used
6	Adaptive Computation Offloading from Mobile Devices into the Cloud	2012 10th IEEE International Symposium on Parallel and Distributed Processing with Applications	Android application , MACS middleware	95% energy savings and significant performance gains	Overdependent on cloud sever , static decision making
7	Process Offloading from Android Device to Cloud Using JADE.	2015 International Conference on Circuit, Power and Computing Technologies [ICCPCT]	JADE , Mobile Agent-Based Offloading:	Successful implementation of mobile agent-based offloading using JADE	Security Enhancement , optimization

2.2 KEY GAPS IN THE LITERATURE

While summarizing the research papers, several key gaps and potential areas for improvement were identified across the studies:

Security Concerns:

- Most papers acknowledged security challenges in mobile offloading.
- There's a dependency on cloud providers for security, and potential threats from malicious hosts or code tampering were highlighted.
- Proposed solutions involve intelligent agent behaviours, but the effectiveness of these measures might need further validation.

Hardware Dependency:

- Some frameworks, like MAUI, were noted for being dependent on specific hardware architectures.
- Issues related to different CPU instruction architectures between mobile devices and servers were recognized.

Limited Real-World Application Scenarios:

Many studies offer comprehensive theoretical frameworks or simulations but lack extensive validation through real-world application deployments. This gap makes it difficult to gauge the practical effectiveness and challenges of these solutions in diverse, real-world environments.

Support for Diverse Technologies and Platforms:

Many studies focus on specific technologies or platforms (e.g., Android, JADE), potentially limiting their applicability. The integration of offloading techniques with a broader range of technologies, such as Bluetooth , Wi-Fi direct and emerging communication technologies like 5G, needs more attention.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS

3.1.1 Functional requirements

Task Offloading:

- Service Receivers should be able to submit computation-intensive tasks for offloading.
- Service Providers should be able to offer their resources for task execution.

Communication Interface:

- Establish a reliable communication interface between the offloadingMaster and OffloadingSlave.
- Ensure secure data transmission and reception.

Task Allocation:

- The system will allocate offloading tasks to available slave devices.
- Factors like battery percentage , CPU frequency, and time spent during task allocation will be taken care of.

Results Transmission:

- Successful completion of tasks by slave device should trigger the transmission of results back to the master device.
- Results will be efficiently communicated back.

3.1.2 Non - Functional requirements

Performance:

- The system should exhibit low latency and high responsiveness in processing offloading requests and task allocations.
- Offloading tasks should show improved performance compared to the local execution of tasks.

Scalability:

- The system must be scalable to handle concurrent and multiple offloading requests.
- Ensure that the system's performance does not significantly go decreasing with an increase in user base.

Security:

- Ensure secure communication channels between the offloadingMaster and OffloadingSlave.

Reliability:

- The system should be reliable, properly handling downtime.
- Properly implement error-handling mechanisms to address potential issues during task offloading.

Usability:

- The user interfaces UI of offloadingMaster and OffloadingSlave should be intuitive and user-friendly.
- Ensure easy navigation of the user .
- Should be easy to understand interface.

3.1.3 HARDWARE REQUIREMENTS

The hardware requirements are as follows:

- **Laptop:** A laptop with enough battery power and RAM is needed to run android studio.
- **Android Phones :** Mobile devices which act as offloadingMaster and OffloadingSlave having enough processing power to handle computation-intensive tasks efficiently.
- **Connectivity :** Reliable Wi-Fi connectivity , Bluetooth other suitable wireless communication technology.
- **GPS Module:** The device should have a GPS module to provide location data (latitude and longitude).
- **Battery Sensor:** The device should have a battery sensor capable of reporting battery level and other battery status information.
- **Android Version:** The device should be running at least Android 6.0 (Marshmallow) or higher to ensure compatibility with the APIs used in the application
- Mobile devices should have developer options and USB debugging to be able to run as offloadingMaster or offloadingServer.
- Compatible with Java runtime environments and networking protocols to support the proposed computation offloading mechanism.

3.1.4 SOFTWARE REQUIREMENTS

The software requirements needed for the completion of the project are as follows:

- **Java Runtime Environment (JRE):** JRE installed for executing the Java-based android application.
- **Operating System:** Android OS for running the android application.
- **Android Studio:** Android Studio serves as the official integrated development environment for Google's Android operating system. We can also use Netbeans or eclipse instead of Android studio.
- **JDK(Java Development Kit):** The Java Development Kit (JDK) represents a distribution of Java Technology provided by Oracle Corporation. It implements both the Java Language Specification (JLS) and the Java Virtual Machine Specification.

Libraries and imports:

- android.bluetooth.BluetoothDevice
- android.bluetooth.BluetoothSocket
- android.content.BroadcastReceiver
- android.location.Location
- android.location.LocationListener
- android.os.BatteryManager
- android.provider.Settings
- android.widget.Button;
- android.widget.ListView;
- android.widget.TextView;
- android.widget.Toast

Required Permissions:

- ACCESS FINE LOCATION
- ACCESS COARSE LOCATION
- INTERNET
- BLUETOOTH
- BLUETOOTH ADMIN

3.2 PROJECT DESIGN AND ARCHITECTURE

Project design

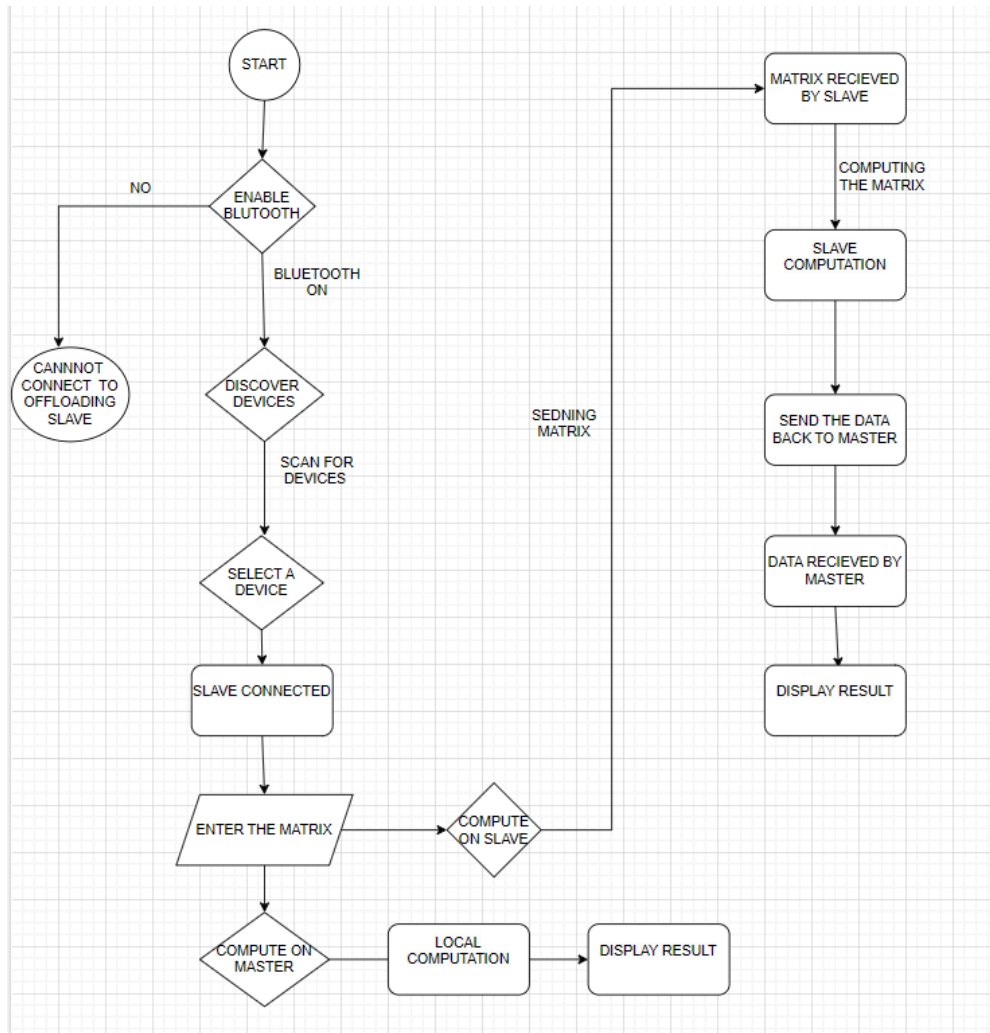


Fig1

3.2.1 METHODOLOGY

Implementation of Offloading Master

The offloading master application acts as the central hub for user interaction and distributed computation in this project. It provides an interface for users to enter matrices and displays the final results .

Here's a breakdown of its basic functionalities:

User Interface (UI) Management: This section handles user interactions like receiving matrix inputs, triggering Bluetooth operations, and displaying results.

Bluetooth Management: This component takes care of Bluetooth functionalities like enabling, connecting to paired devices, and data transfer. It establishes and maintains the connection with the slave device, potentially fetching battery level and location information upon user request.

Matrix Transfer: When offloading computation, the master prepares the matrices for transmission by converting them into a format suitable for Bluetooth communication (e.g., JSON). It then sends this formatted data to the slave device for processing.

Local Matrix Multiplication: The master device can also perform the matrix multiplication itself, providing an alternative to offloading the task. This might be useful if offloading proves inefficient or undesirable due to specific circumstances.

Power Consumption Analysis: The code likely monitors and measures power usage during both local computation and Bluetooth communication. This allows for comparing the efficiency of offloading calculations to the slave device versus performing them locally on the master device.

User Result Display: Finally, the master receives the computed results (C) either from the slave device or from the local computation module and presents them to the user through the UI.

Methods in offloading master:-

initializeByAttributes():

This function connects UI elements (buttons, text views, etc.) in the layout file to their corresponding member variables in the MainActivity class. This creates a link between the visual elements and the code that controls them.

onCreate():

This is a standard function called when the activity is first created. It performs various initializations like:

- Registering a broadcast receiver to listen for battery level changes.
- Requesting location permissions.
- Setting up Bluetooth functionalities:
 - Enabling Bluetooth (if not already enabled).
 - Displaying a list of paired devices for the user to select.
- Defining methods within this function for handling button clicks related to:
 - Bluetooth activation
 - Displaying paired devices
 - Selecting a device from the list
 - Transferring matrices for offloading computation
 - Performing computation on the master device (local computation)
 - Requesting information from the slave device (battery level, location)

calcPower():

This function retrieves the current battery charge counter and returns the value. This value can be used to calculate power consumption during different operations.

Bluetooth Functions:

These functions handle various Bluetooth functionalities:

- **enableBluetooth():** Turns on Bluetooth on the master device.
- **displayPairedDevices():** Retrieves a list of paired Bluetooth devices and displays them for user selection.
- **connectToSlaveDevice():** Establishes a Bluetooth connection with the selected slave device.

matTransfer():

This function handles transferring matrices to the slave device for offloading computation. It involves:

- Retrieving matrix values from user input fields.
- Converting the matrices to a format suitable for transmission (e.g., JSON).
- Sending the formatted data over Bluetooth to the slave device.
- Updating the power consumption data for distributed computation using `calcPower()`.

masterComp():

This function performs matrix multiplication on the master device (local computation). It involve:

- Retrieving matrix values from user input fields.
- Performing the multiplication algorithm using the retrieved matrices.
- Updating the power consumption data for local computation using `calcPower()`.

reqInfo():

This function sends a request to the slave device to get its battery level and location information. It involve:

- Creating a JSON message containing a request type (e.g., "getBatteryLevel").
- Sending the request message over Bluetooth to the slave device.

convertToString() :

This function convert an integer array (representing a matrix) to a string for better display in the UI. It's helpful for presenting the matrices in a user-friendly format.

handler:

This function processes messages received from the Bluetooth connection. It handle different message types based on predefined formats:

- Messages indicating successful/failed pairing with a slave device.
- Messages containing battery level and location information received from the slave device.
- Messages with computation results sent back by the slave device.
- The handler would then update the UI or perform actions based on the received message type.

Methods in slave:-

initializeByAttributes()

This Initializes UI elements by finding and assigning them to variables. It finds and assigns UI elements like buttons (bluetoothActivate, searchBtn), list view (listDevices), and text views (batteryDisp, compExecTime, conStat, msgRec).

onCreate():

This standard function performs initializations on the slave device when the application starts. It involve:

- Sets the layout for the activity using setContentView.
- Initializes UI elements and registers a battery information receiver.
- Requests location updates and handles location changes.
- Requests Bluetooth activation and sets up listeners for Bluetooth-related buttons.
- Starts a Bluetooth server (ServerClass) when the "Listen" button is clicked.
- Calls listDevMethod to handle item clicks in the device list.

onReceive():

This function handles incoming messages received over the Bluetooth connection from the master device. It involve:

- Identifying the message type based on predefined formats.
- Processing different message types:
 - Messages containing matrices sent for offloading computation.
 - Messages requesting battery level or location information from the master device.
- Extracting the data from the received message (e.g., matrix values from JSON format).

MatrixMultiplication:

This function performs the matrix multiplication on the slave device. It involve:

- Parses input strings (numOne, numTwo) representing matrices and performs matrix multiplication.
- Calculates power consumption based on the initial and final power values.
- Displays the power consumption in the UI .
- Returns the result of matrix multiplication as a 2D array.

Handler:

- Handles messages received from Bluetooth communication.
- Processes different message types (SEARCHING, PAIRING, PAIRING_SUCCESS, PAIRING_FAIL, MESSAGE_DETECTED) using a switch-case statement.
- Parses JSON messages received and takes appropriate actions based on the message content.
- Initiates matrix multiplication, updates UI elements, and sends responses over Bluetooth.

ServerClass extends Thread

- Represents a Bluetooth server thread.
- Creates a Bluetooth server socket (servSock) and listens for incoming connections.
- Handles incoming connections and sets up communication threads (SendReceive) for data exchange

calcPower()

- Calculates power consumption based on battery charge counter.
- Retrieves the battery charge counter using BatteryManager.
- Returns the calculated power consumption value.

listDevMethod()

- Handles item clicks in the Bluetooth device list.
- Sets an item click listener for the device list (listDevices).
- Initiates a Bluetooth client connection (ClientClass) when an item is clicked and updates UI status accordingly.

Interaction between offloading master and Offloading slave

The interaction between the offloading slave and master applications in this project follows a client-server model using Bluetooth communication:

Master (Client):

- 1. Initiates connection:** Establishes a Bluetooth connection with the selected slave device.
- 2. Sends matrices :**
 - a.** Prepares matrices for transmission by converting them to a suitable format (e.g., JSON).
 - b.** Sends the formatted data over Bluetooth to the slave device.
- 3. Requests information:**
 - a.** Sends a request message over Bluetooth asking for the slave's battery level or location information (if desired).
- 4. Receives results:**
 - a.** Waits for a response from the slave device.
 - b.** Receives the computed result sent back by the slave device in a transferable format (e.g., JSON).
- 5. Receives information:**

- a. Receives a response message containing the requested battery level and location information from the slave device .
- 6. Displays results:**
- a. Converts the received result back to a human-readable format .
 - b. Displays the results and the received information (battery level, location) to the user.

Slave (Server):

- 1. Waits for connection:** Listens for incoming Bluetooth connections from the master device.
- 2. Receives matrices:**
 - a. Upon receiving a connection and data from the master, extracts the matrices from the formatted message (e.g., JSON).
- 3. Performs computation:**
 - a. Executes the matrix multiplication algorithm using the received matrices.
- 4. Responds to information request:**
 - a. If the master requests information (battery level, location), retrieves the data and sends it back in a formatted message.
- 5. Sends results:**
 - a. Converts the calculated result to a transferable format (e.g., JSON).
 - b. Sends the formatted result back to the master device over Bluetooth.

Communication Protocol:

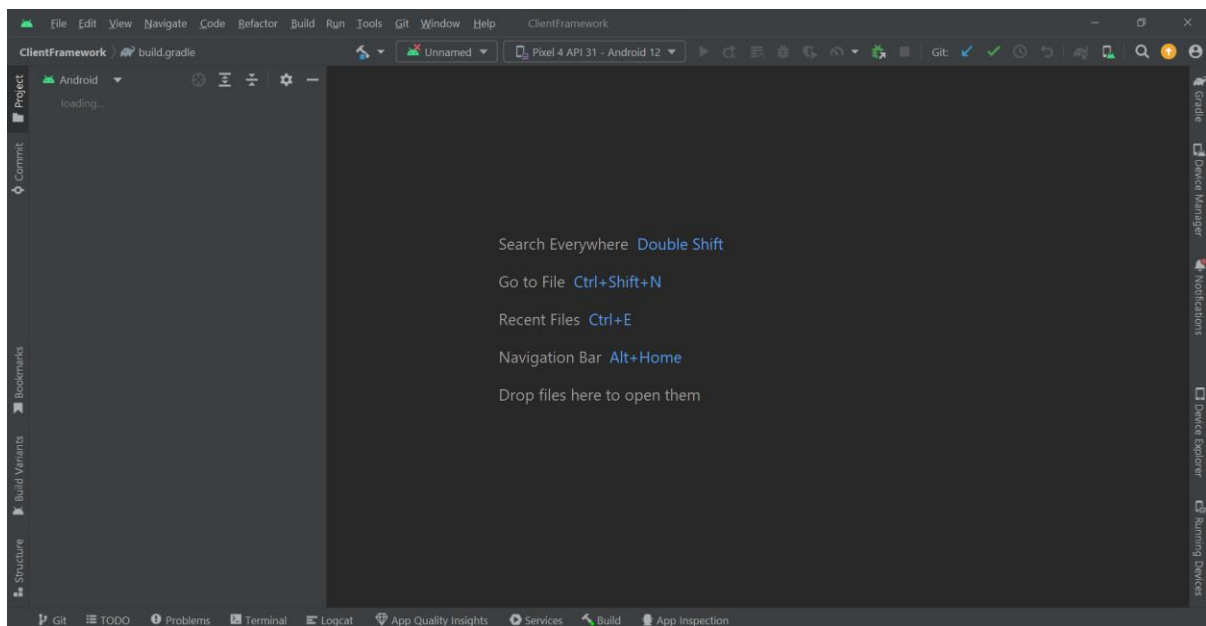
1. It uses Bluetooth for efficient communication.
2. Data transfer between devices relies on a predefined message format (e.g., JSON) for clarity and ease of parsing.

3.3 IMPLEMENTATION

The following are steps to implement the project:

3.3.1 Downloading essential software

The initial step to develop the project was downloading essential tools that would be further required in the making of the project. So we installed the Android studio and netbeans software which are necessary for building and run android application .



3.3.2 Downloading JRE

The next important step was to download JRE(Java Runtime Enviroment). JRE has many tools and libraries and also JVM (Java Virtual machine).

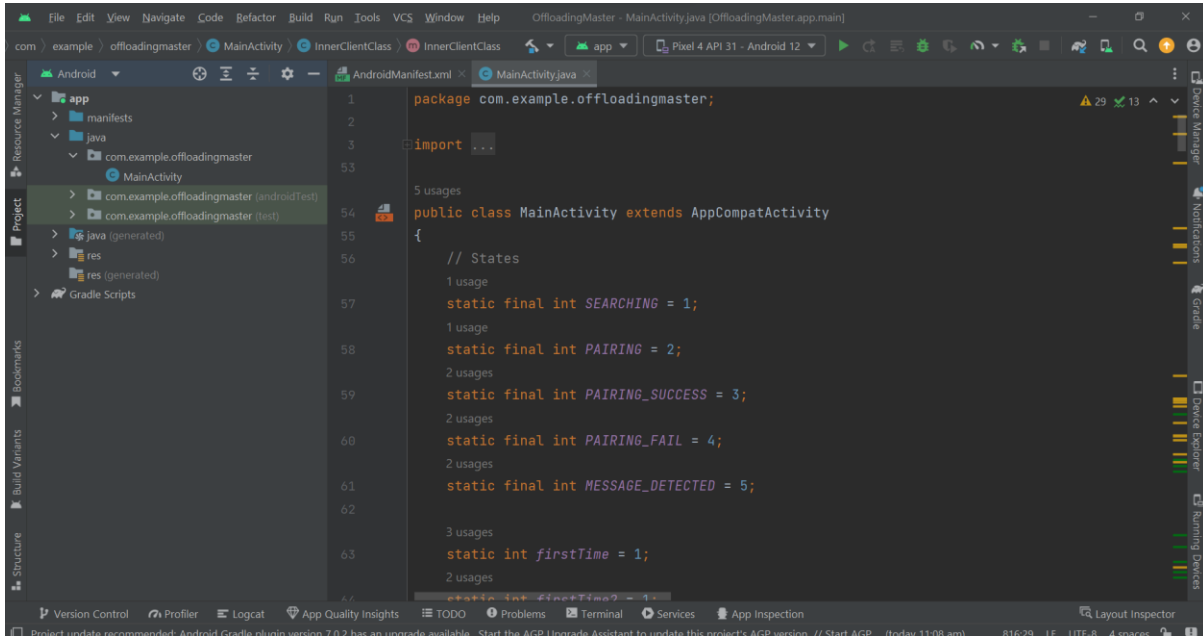
The screenshot shows the Oracle Java website's download page. The browser address bar displays 'https://www.java.com/en/download/manual.jsp'. The navigation bar includes 'Download', 'Developer Resources', and 'Help'. A search bar is located on the right. The main content area features a 'Help Resources' sidebar with links to 'Troubleshoot Java' and 'Remove older versions'. The central heading is 'Java Downloads for All Operating Systems', with a sub-heading 'Recommended Version 8 Update 391' and a release date of 'October 17, 2023'. A prominent yellow box contains 'Important Oracle Java License Information', stating that the Oracle Java License changed for releases starting April 16, 2019. It explains that the Oracle Technology Network License Agreement for Oracle Java SE is substantially different from prior licenses, allowing personal and development use at no cost, but other uses may no longer be available. It also notes that commercial license and support are available with a low-cost Java SE Subscription. Below this box, a disclaimer states that by downloading Java, users acknowledge they have read and accepted the terms of the Oracle Technology Network License Agreement for Oracle Java SE.

3.3.3 Master Application Development:

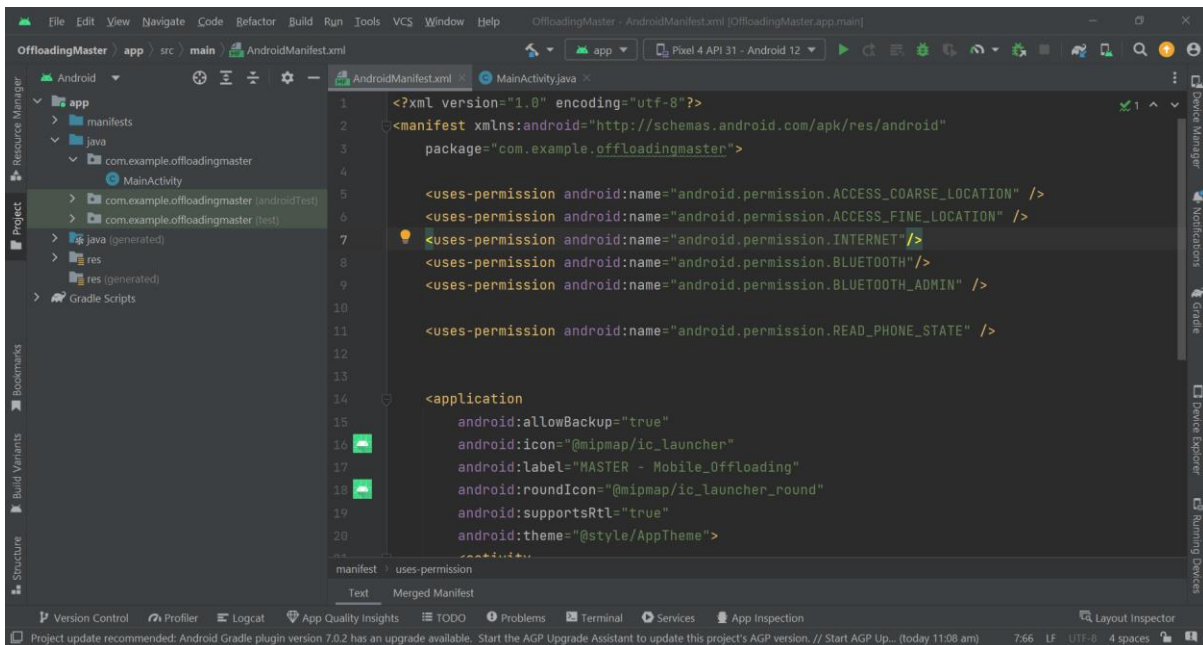
Now start working on offloading master application. Download all the required dependencies.

- **User Interface:** Create a user interface with elements like buttons, text fields, and a results display area using the android studio.
- **Bluetooth Management:** Implement functionalities to enable/disable Bluetooth, scan for paired devices, connect to the selected slave device, and manage data transfer over Bluetooth.
- **Matrix Handling:** Develop classes or data structures to represent matrices, including functions for input handling, conversion to transferable formats (JSON), and conversion back after receiving from the slave.
- **Local Computation:** Implement the matrix multiplication algorithm within the master application if local computation is desired.
- **Power Consumption Analysis :** Include functions to monitor power usage during local computation and Bluetooth communication for comparison purposes.
- **Communication Logic:** Implement functions to send matrices (A, B) to the slave for offloading, receive results (C), handle potential errors, and manage communication flow.

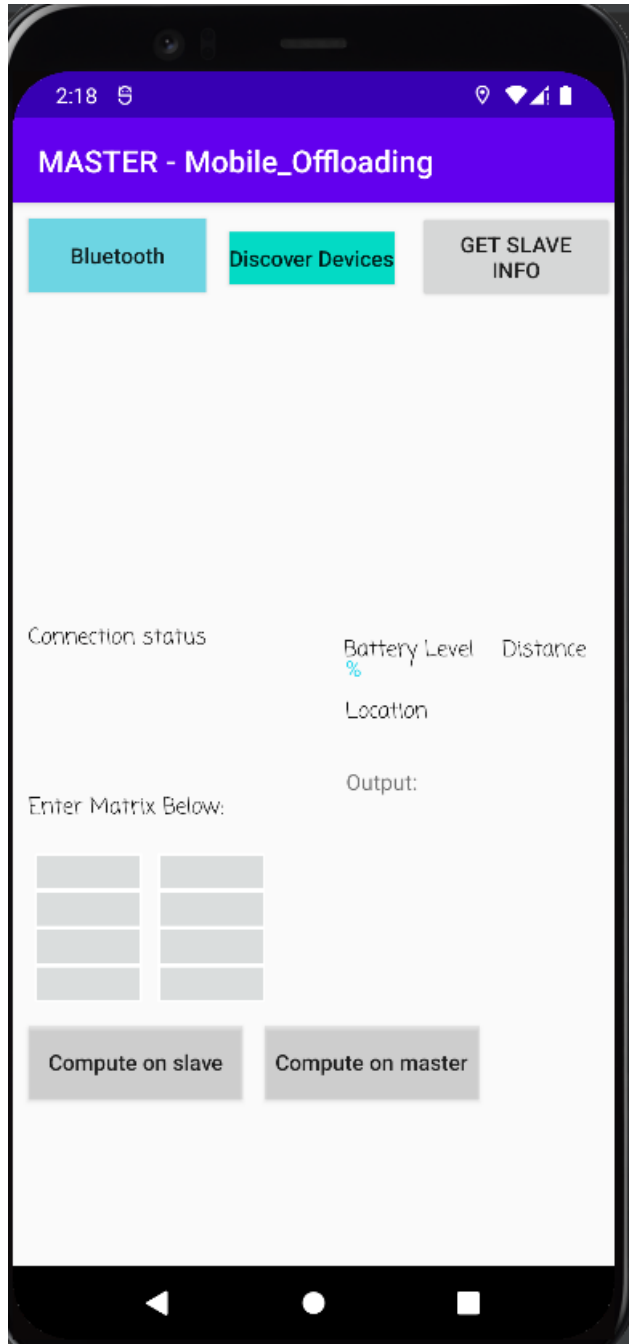
- **Information Request** : Develop functions for sending requests to the slave device to retrieve battery level and location information and process the received data.



```
1 package com.example.offloadingmaster;
2
3 import ...
4
5 5 usages
6
7 public class MainActivity extends AppCompatActivity
8 {
9     // States
10    1 usage
11    static final int SEARCHING = 1;
12    1 usage
13    static final int PAIRING = 2;
14    2 usages
15    static final int PAIRING_SUCCESS = 3;
16    2 usages
17    static final int PAIRING_FAIL = 4;
18    2 usages
19    static final int MESSAGE_DETECTED = 5;
20    3 usages
21    static int firstTime = 1;
22    2 usages
23
24 //
25
26 //
```



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.example.offloadingmaster">
4
5     <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
6     <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
7     <uses-permission android:name="android.permission.INTERNET" />
8     <uses-permission android:name="android.permission.BLUETOOTH" />
9     <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
10
11     <uses-permission android:name="android.permission.READ_PHONE_STATE" />
12
13
14 <application
15     android:allowBackup="true"
16     android:icon="@mipmap/ic_launcher"
17     android:label="MASTER - Mobile_Offloading"
18     android:roundIcon="@mipmap/ic_launcher_round"
19     android:supportRtl="true"
20     android:theme="@style/AppTheme">
```

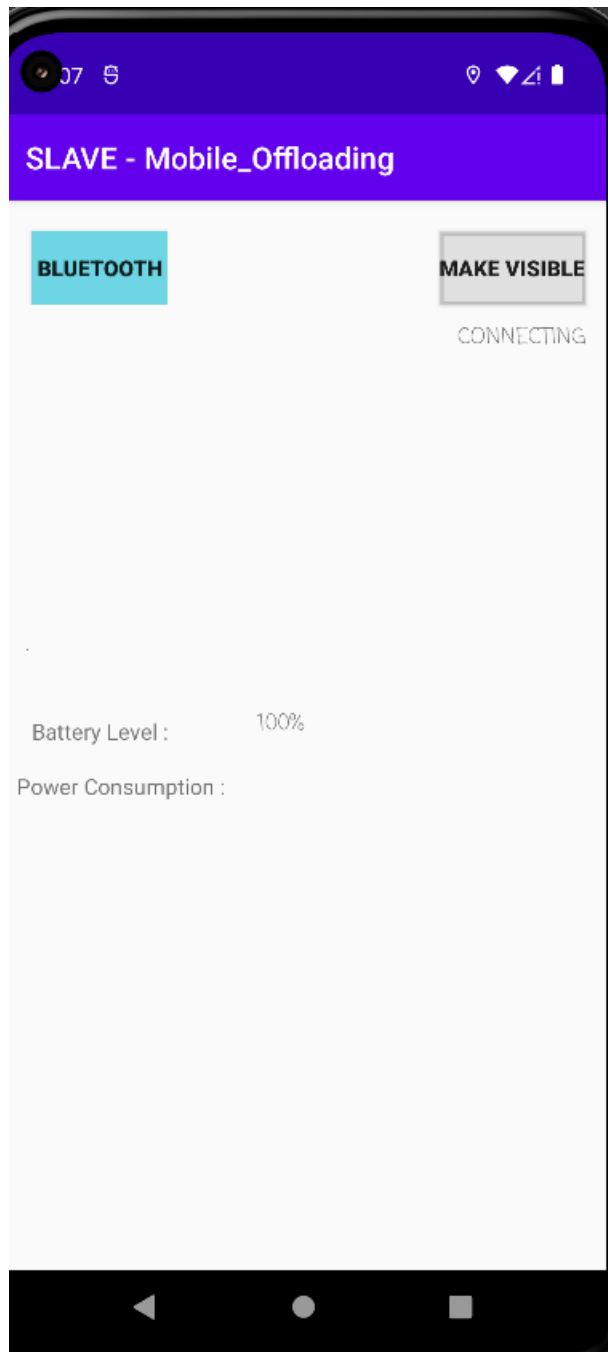


The screenshot shows the AndroidManifest.xml file for the 'OffloadingSlave' project. The file is located at 'app/src/main/AndroidManifest.xml'. The code defines the application's permissions and main activity.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.example.offloadingslave">
4
5     <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
6     <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
7     <uses-permission android:name="android.permission.INTERNET" />
8     <uses-permission android:name="android.permission.BLUETOOTH" />
9     <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
10
11     <application
12         android:allowBackup="true"
13         android:icon="@mipmap/ic_launcher"
14         android:label="SLAVE - Mobile Offloading"
15         android:roundIcon="@mipmap/ic_launcher_round"
16         android:supportRtl="true"
17         android:theme="@style/AppTheme">
18         <activity android:name=".MainActivity">
19             <intent-filter>
20                 <action android:name="android.intent.action.MAIN" />

```

The IDE interface includes a toolbar at the top with options like File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, and Help. The left sidebar shows the project structure with folders for 'app', 'manifests', 'java', 'res', and 'Gradle Scripts'. The bottom status bar displays 'Text Merged Manifest' and a notification: 'Project update recommended: Android Gradle plugin version 7.0.2 has an upgrade available. Start the AGP Upgrade Assistant to update this project's AGP version. // Start AGP Upg... (today 12:22 pm)'. The bottom right corner shows '11 | LF | 4 spaces'.



3.4.5 KEY CHALLENGES

Some key challenges that we faced while developing this project are:

- **Communication Protocols:** Implementing efficient and secure communication protocols between the mobile devices can be challenging.
- **Concurrency and Scalability:** Managing concurrent offloading requests and ensuring scalability to handle a large number of devices simultaneously can be complex.
- **Optimizing Offloading Decisions:** Developing algorithms to decide when and what tasks to offload to maximize the benefits without causing unnecessary delays or resource consumption.
- **Error Handling and Fault Tolerance:** Ensuring robust error handling and fault tolerance mechanisms to handle unexpected scenarios and ensure system stability.
- **Making of Application :** Making of entire application is itself a very difficult task . It was a big challenge to create this application.

CHAPTER 4: TESTING

4.1 TESTING STRATEGY

This testing strategy emphasizes a multi-layered approach to ensure the offloading project functions as intended. It involves testing individual components, communication protocols, and overall user experience.

Unit Testing:

- Focus on testing individual functionalities within each application (master and slave).
- Utilize unit testing frameworks (e.g., JUnit for Java) to create isolated test cases for:
 - Bluetooth functionalities (e.g., enabling, connecting, sending/receiving data)
 - Matrix operations (e.g., matrix creation, conversion to/from JSON)
 - Local computation logic.
 - Information retrieval .
 -

Communication Testing:

- Test the communication protocols between the master and slave applications.
- Set up controlled environments to simulate various scenarios:
 - Successful Bluetooth connection and data transfer
 - Connection failures
 - Invalid data formats (e.g., incorrect JSON messages)
 - Unexpected messages or disconnections

Performance Testing:

- Evaluate the performance of offloading computations compared to local execution.
- Measure factors like:
 - Time taken for computation on the slave device
 - Time taken for data transfer over Bluetooth
 - Battery consumption during local and offloaded computation.
- Analyze the results to understand the efficiency gains or trade-offs associated with offloading calculations.

CHAPTER 5: RESULTS AND EVALUATION

5.1 RESULTS

1. Performance Improvement:

- The experiments conducted demonstrated significant improvements in task execution time.
- Testing showed significant reduction of load on the master device..

2. Device Utilization:

- The utilization of offloading slave effectively demonstrated the concept of computational offloading, proving that devices with higher processing capacities can efficiently handle computationally intensive tasks.

3. Variable Workloads:

- The system's performance was tested with varying types of matrix, providing insights into how the offloading mechanism behaves under different workloads.

Power consumption in master device : 7.718mah

Power consumption in slave device : 1.415mah

There is a significant difference in the power consumption by master and slave device.

Slave Device

BLUETOOTH

MAKE VISIBLE

CONNECTED

Row1 of Matrix-1: 1 2 5 6
Row2 of Matrix-1: 3 4 7 8
Col1 of Matrix2: 3 4 7 8
Col2 of Matrix-2: 11 12 15 16

Battery Level : 72%

Power Consumption 1.415 mAh

Computation Time: 5083micro seconds
Computation Time: 9140micro seconds

Master Device

Bluetooth

GET SLAVE
INFO

Discover Devices

OnePlus 7T

Galaxy J7 Prime

boAt Rockerz

CONNECTED DEVICES:

Battery Level

OnePlus 7T - Slave

72 %

Enter Matrix Below:

Output:

1 2 3 4	1 2 3 4
5 6 7 8	5 6 7 8
9 10 11 12	9 10 11 12
13 14 15 16	13 14 15 16

90 100 110 120
202 228 254 280

Compute on slave

Compute on master

The resultant matrix from OnePlus 7T: 110 120 254 280

Computation Time by Master: 1383 micro seconds

Power Consumed without Distributed Computation: 7.718 mAH

Power Consumed with Distributed Computation: 9.241 mAH

CHAPTER 6: CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSION

This project has successfully explored the concept of offloading computation in a distributed application. The offloading master application acts as a user interface hub, facilitating matrix input, displaying results, and managing communication with the offloading slave device. By leveraging Bluetooth for efficient communication and JSON for data exchange, the project demonstrates the feasibility of offloading computationally intensive tasks.

Key Achievements:

Functional Offloading: The master application successfully sends matrices to the slave device for calculation. The slave device performs the matrix multiplication and sends the results back, showcasing the core functionality of offloading computation.

Local Computation: The master application can also perform the computation locally, providing a valuable alternative for scenarios where offloading might be less efficient.

Power Consumption Analysis : By monitoring power usage during local and offloaded computations, the project offers insights into the potential benefits of offloading in terms of power conservation, especially for resource-constrained devices.

Information Retrieval : The project demonstrates the possibility of requesting and displaying information like battery level or location from the slave device, potentially enriching the user experience.

6.2 FUTURE SCOPE

The future scope of the Mobile Cloud Computing (MCC) offloading mechanism project is promising, offering several avenues for further research, development, and application. Some key areas of future exploration and enhancement include:

Optimization and Algorithm Refinement:

Further optimization of the algorithm and exploration of alternative algorithms for resource allocation could enhance the efficiency of task offloading. Algorithmic improvements may lead to better decision-making processes regarding resource allocation in dynamic environments.

Dynamic Adaptability:

Investigate methods to make the offloading mechanism more adaptive to dynamic changes in network conditions, device capabilities, and user preferences. This adaptability could involve real-time adjustments to the offloading strategy based on varying parameters.

Energy-Efficiency Considerations:

Future research can delve deeper into the energy implications of the offloading mechanism. Developing strategies to minimize energy consumption during offloading processes, especially on resource-constrained devices, could contribute to more sustainable and eco-friendly mobile computing.

REFERENCES

1. A Survey of Computation Offloading for Mobile Systems 10 April 2012
© Springer Science+Business Media, LLC 2012.
2. Offloading Android Applications to the Cloud without Customizing Android Eighth IEEE PerCom Workshop on Pervasive Wireless Networking 2012, Lugano (23 March 2012)
3. An Android-based Application for Computation Offloading in Mobile Cloud Computing Proceedings of ACM/CSI/IEEECS Research & Industry Symposium on IoT Cloud For Societal Applications (IoTCloud'21)
4. A transparent code offloading technique for Android devices 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)
5. Towards Computational Offloading in Mobile Device Clouds 2013 IEEE International Conference on Cloud Computing Technology and Science
6. Adaptive Computation Offloading from Mobile Devices into the Cloud 2012 10th IEEE International Symposium on Parallel and Distributed Processing with Applications
7. R. Kemp, N. Palmer, and H. Bal, "Cuckoo: a Computation Offloading Framework for Smartphones," in Proceedings of the 2nd International ICST Conference on Mobile Computing (MobiCASE 2010), Santa Clara, CA, USA, October 2010.
8. H. Flores, P. Hui, S. Tarkoma, Y. Li, S. Srirama, R. Buyya, Mobile code offloading: From concept to practice and beyond, *IEEE Commun. Mag.* (2015).
9. B. Zhou, A.V. Dastjerdi, R.N. Calheiros, S.N. Srirama, R. Buyya, MCloud: A Context-Aware Offloading Framework for Heterogeneous Mobile Cloud, *IEEE Trans. Serv. Comput.* 10 (2017) 797–810.
10. A. Kumar, R. Yadav, G. Baranwal, NearBy-Offload: An Android based Application for Computation Offloading, in: 2021: pp. 357–362.

11. G. Orsini, D. Bade, W. Lamersdorf, Computing at the Mobile Edge: Designing Elastic Android Applications for Computation Offloading, in: Proc. - 2015 8th IFIP Wirel. Mob. Netw. Conf. WMNC 2015, 2016: pp. 112–119.
12. B. Zhou, S.N. Srirama, R. Buyya, An auction-based incentive mechanism for heterogeneous mobile clouds, *J. Syst. Softw.* 152 (2019) 151–164.
13. E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: Making smartphones last longer with code offload. In Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, MobiSys '10, pages 49–62, New York, NY, USA, 2010. ACM.
14. H. T. Dinh, C. Lee, D. Niyato, and P. Wang. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*, 13(18):1587–1611, 2013.
15. N. Fernando, S. W. Loke, and W. Rahayu. Mobile cloud computing: A survey. *Future generation computer systems*, 29(1):84–106, 2013.
16. J. Flinn. Cyber foraging: Bridging mobile and cloud computing. *Synthesis Lectures on Mobile and Pervasive Computing*, 7(2):1–103, 2012.
17. H. Flores, P. Hui, S. Tarkoma, Y. Li, S. Srirama, and R. Buyya. Mobile code offloading: from concept to practice and beyond. *IEEE Communications Magazine*, 53(3):80–88, March 2015.
18. M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen. Comet: Code offload by migrating execution transparently. In Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation, OSDI'12, pages 93–106, Berkeley, CA, USA, 2012. USENIX Association

ANDROID OFFLOADING USING CLOUD

ORIGINALITY REPORT

18%

SIMILARITY INDEX

16%

INTERNET SOURCES

9%

PUBLICATIONS

%

STUDENT PAPERS

PRIMARY SOURCES

1	ir.juit.ac.in:8080 Internet Source	5%
2	www.ir.juit.ac.in:8080 Internet Source	3%
3	www.researchgate.net Internet Source	2%
4	www.qscience.com Internet Source	1%
5	Romulo Reis de Oliveira, Nicolas Meira da Silva Schirmer, Mateus Machry, Tiago Coelho Ferreto. "A transparent code offloading technique for Android devices", 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), 2017 Publication	1%
6	onlinelibrary.wiley.com Internet Source	1%
7	www.freepatentsonline.com Internet Source	<1%

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT
PLAGIARISM VERIFICATION REPORT

Date:

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: _____ Department: _____ Enrolment No _____

Contact No. _____ E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found Similarity Index at(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
Report Generated on	<ul style="list-style-type: none"> • All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String 		Word Counts	
			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com