**Speaker 1:**  
Hello everyone.  
We are Riddhi Patil and Kashish Mamania from K. J. Somaiya School of Engineering.  
Today, we’ll present our project: “Agent-Based Multistage Application Partitioning in Mobile Cloud Computing.”

**Speaker 2:**  
Let’s start with the introduction.  
As mobile apps like facial recognition and gaming get more complex, they need more processing power and battery.  
Mobile Cloud Computing, or MCC, helps by sending heavy tasks to cloud servers, which improves performance and saves battery.  
But the main challenge is deciding which parts of an app should run on the phone and which should be offloaded to the cloud.  
Our framework uses smart agents and a multistage graph approach to make these decisions dynamically for the best results.

**Speaker 1:**  
Now, let’s talk about the motivation behind our work.  
Mobile devices have limited power and battery. Running complex tasks locally drains the battery and slows down the device.  
Offloading to the cloud helps, but it can cause delays due to network issues.  
So, we need a system that can decide in real time whether a task should run on the device, at the edge, or in the cloud.  
Our goal is to design such a system to reduce both energy use and delays.

**Speaker 2:**  
Moving on to the scope.  
Our Agent-Based Multistage Partitioning, or ABMP, framework is useful for any situation where mobile devices run heavy apps and need quick decisions about offloading.  
This includes areas like augmented reality, smart health monitoring, autonomous vehicles, mobile gaming, and industrial IoT.  
It works across three layers: mobile, edge, and cloud, adapting in real time to changing conditions like bandwidth and CPU load.  
However, it assumes stable network infrastructure and is currently designed for single users. Handling many users and predicting trends with machine learning are future goals.

**Speaker 1:**  
Let’s look at our objectives.  
Our main aim is to build an intelligent, agent-based system that improves how mobile apps are split and offloaded.  
We want to minimize energy use and execution time, use multistage graph partitioning for better control, and have a three-tier system: mobile, edge, and cloud, each with its own agent.  
We also want to make real-time, context-aware decisions and prove our model works better than existing methods.

**Speaker 2:**  
Now, related work.  
Earlier solutions often used static methods or simple machine learning, but they didn’t adapt well to changing environments.  
Some used DAGs, or Directed Acyclic Graphs, to model app workflows, but these had high delays and poor coordination.  
Our work builds on these ideas but adds more dynamic and responsive decision-making.

**Speaker 1:**  
Here are our key contributions:  
First, we use a multistage graph partitioning strategy.  
Second, we have a three-tier architecture with agents at each level.  
Third, we use a real-time offloading decision algorithm.  
And finally, we show experimental results that prove our method works better.

**Speaker 2:**  
Let’s discuss the system architecture.  
Our system has three layers: mobile devices, edge servers, and central cloud servers.  
Each layer has a smart agent that monitors resources and communicates with the others.  
When a user starts a task, the mobile agent checks with the others to decide the best place to run it, aiming to save energy and time.

**Speaker 1:**  
Now, the mobile agent architecture.  
The mobile agent has four modules:  
First, the Data Collection Module gathers info like memory, battery, and bandwidth.  
Second, the Graph Module turns app tasks into a multistage graph.  
Third, the Partitioning Module classifies tasks as offloadable, must-offload, or unoffloadable.  
And fourth, the Offloading Module decides where to run each task, considering system loads and waiting times.

**Speaker 2:**  
For application workflow modeling, each app is represented as a Directed Acyclic Graph, or DAG.  
Nodes are tasks, and edges show dependencies.  
Tasks are classified as unoffloadable, offloadable, or must-offload.  
This helps the system make smart, parallel, and context-aware decisions.

**Speaker 1:**  
Next is cost modeling.  
We focus on two main costs: execution time and energy use.  
The system calculates these costs for running tasks locally, at the edge, or in the cloud, considering CPU speed, bandwidth, data size, and power usage.  
The goal is to minimize both energy and time while keeping service quality high.

**Speaker 2:**  
Now, problem formulation.  
The main problem is an optimization:  
We want to minimize energy use for each user, while making sure the total execution time stays under a set deadline.  
This means finding the best place to run each task in the workflow.

**Speaker 1:**  
Let’s look at the ABMP scheme overview.  
The ABMP scheme works in two phases:  
First, it partitions the app into a multistage graph.  
Then, at each stage, it sorts tasks by offloading feasibility and uses real-time data to decide where to run them.  
This makes task management responsive and efficient.

**Speaker 2:**  
For multistage graph partitioning, unlike older methods that made one big decision for the whole app, our approach makes decisions at each stage.  
This gives better control and accuracy, ensuring tasks are distributed efficiently based on current conditions.

**Speaker 1:**  
Now, the offloading algorithm.  
The algorithm checks each group of tasks for resource availability, bandwidth, and urgency.  
It tries to offload to the edge first for lower delay.  
If resources are low, it can compress, retry, or send tasks to the cloud.  
This flexible approach ensures the best performance in changing conditions.

**Speaker 2:**  
Let’s talk about the experimental setup.  
We tested our framework on three real-world apps: Object and Pose Recognition, Indoor Localization, and Optical Character Recognition.  
Simulations were run using CloudSim and SOOT, with different CPU and RAM setups, and compared to previous studies.

**Speaker 1:**  
Now, the results.  
Our ABMP algorithm consistently had lower execution times than other methods, especially when bandwidth was low.  
It also used less energy, thanks to smarter partitioning and load distribution.  
This proves the value of using agents and multistage processing.

**Speaker 2:**  
For future work, we plan to expand the system for multi-user environments and ad hoc cloud networks.  
This will involve more agents and smarter algorithms to handle shared resources and user competition.

**Speaker 1:**  
To conclude, our ABMP framework uses smart agents and multistage graph analysis to make dynamic, context-aware decisions for offloading mobile app tasks.  
It outperforms older methods by making detailed, real-time decisions, reducing both energy use and execution time.

Thank you for your attention. We’re happy to take any questions.