1: Good Evening, Dr Siti Nurlaili. My name is Muhammad Afiq bin Zakaria. Today, I will be presenting my Final Year Project titled 'Easy Parcel: IoT-Based Parcel Storage System for University Villages'.

2: Here is the agenda for today's presentation. I will begin with the Introduction, followed by the Literature Review, Methodology, Results & Discussion, and finally, the Conclusion and Future Works.

4: Let's start with the background. Malaysia is experiencing a significant boom in e-commerce, with income reaching nearly 920 billion Ringgit in the first nine months of 2024 alone.

University students play a major role in this; they are tech-savvy and rely heavily on online platforms for convenience and discounts.

To handle parcels that are overflowing everyday, most universities, especially UTP, implement the centralized ParcelHub system. This system provides key benefits, such as ensuring a secure location to prevent parcel theft and allowing for the organized handling of high delivery volumes in a single, managed facility.

5: While the ParcelHub aims to simplify parcel collection, it still has notable flaws that impact student convenience.

The location is far from most residential villages, making it difficult for students without transport.

It suffers from long queues during peak hours and limited parking.

Furthermore, students are burdened with additional service fees just to collect their parcels when they already have to pay for shipping fees in e-commerce platforms.

The current system simply isn't convenient enough.

6: To address this, my project has four main objectives:

To design an IoT-based smart locker for secure storage.

To build a mobile app for both students and couriers

To implement a secure random password (OTP) and barcode system for identification.

To evaluate the system's usability through user testing

7: The scope of this project involves three main technical pillars:

Hardware: Developing a prototype using microcontrollers and sensors, simulated first via Tinkercad

Software: Developing a Flutter mobile app for cross-platform access. Coding was excuted in VSCode with libraries.

Backend: Implementing Supabase for authentication and real-time database management, to handle OTP and barcode records.

9: "My theoretical framework is built on four concepts:

IoT: To connect the physical locker to the internet.

Cloud Computing: Using Supabase to manage scalability without complex local servers.

Client-Server Architecture: Where the mobile app communicates via HTTP requests with the ESP32 and database.

Authentication: Using OTPs as a time-sensitive security mechanism.

10: In my literature review, I analyzed several key domains. Studies confirm that IoT is the cornerstone of modern automation.

While some smart lockers use biometrics, research shows that barcode-based lockers are a lower-cost alternative that effectively reduces parcel loss.

Furthermore, utilizing cloud databases is essential for real-time synchronization in these IoT systems."

11: The main gap in current solutions is cost and centralization.

Existing smart lockers often use expensive hardware like facial recognition , and campus hubs are too centralized.

My solution addresses this by being Low-Cost, Decentralized (placed at residential blocks) , and Open Source (using Flutter and Supabase).

13: Moving on to methodology, the development was divided into five phases: Hardware Design & Simulation, Hardware Assembly & Testing, Mobile App Development, Database Integration and System Integration.

14: I started with Virtual Prototyping. Since Tinkercad doesn't support ESP32, I used an Arduino proxy to simulate and validate the circuit safety before purchasing any physical components. This helped verify the interactions between the keypad and servo.

This is the schematic diagram for connections of components in Tinkercard.

15: The physical prototype utilizes an ESP32 as the main controller.

It controls a Servo Motor for the locking mechanism , a Keypad for OTP entry , an LCD for status display , and a Buzzer for audio feedback.

16: "During assembly, the ESP32 was programmed in C++ using the Arduino IDE.

I implemented a Web Server on the chip to process app requests and enabled Over-the-Air (OTA) updates to allow for wireless code modifications.

17: For the software, I used Flutter in VS Code.

The app features a Unified Entry Point handling both login and registration.

It implements Role-Based Logic to direct users to either the Student or Courier dashboard.

It uses TabBarController to separate active tasks from completed history.

It also manages the IoT communication to send OTPs to the locker.

The system handles Parcel Processing by auto-generating unique OTPs and barcodes.

For efficiency, I designed a Dual-Input System allowing couriers to use manual entry or QR scanning.

Finally, the app includes Verification Logic that updates the parcel status to 'Collected' once the student scans their package.

18: Finally, for the backend, I integrated Supabase. It hosts relational tables for Users, Parcels, and Lockers and enables real-time synchronization, ensuring that when a parcel status changes, the app updates instantly

20: Here is the final hardware prototype. You can see the integration of the LCD, Keypad, LEDs, and the Servo motor mechanism housed within the locker structure.

21: The logic flow is straightforward. The system starts in an initial state displaying the IP address.

When an OTP is received, it waits for user input. If the entered OTP matches the stored code, the door opens.If the OTP entered is wrong the screen will display "Invalid Input" and user must key in the OTP again.

After collection, the user presses 'A' to lock it again."

22: On the mobile app side, we have a clean Login and Registration interface. It captures essential user information and includes a toggle for password visibility.

The 'Student' dropdown ensures users are categorized correctly upon signup.

23: This is the Courier's view. Couriers have two ways to input data: manual entry or by scanning the student's QR code to reduce errors.

Once submitted, the system automatically generates a unique OTP and barcode.

The History Tab keeps a log of all delivered parcels with real-time status updates.

24: And this is the Student's view. The dashboard shows parcels 'Ready for Pickup' with the assigned locker number.

The student clicks 'Get OTP' to retrieve their code.

After unlocking the locker and taking the parcel, they use the app to scan the parcel's barcode, which updates the status to 'Collected' and moves it to the History tab.

26: In conclusion, Easy Parcel successfully combined ESP32, Supabase, and Flutter to create a secure, automated system. The integration was successful, performance was reliable , and user testing indicated a positive experience with reduced manual handling.

I have met all objectives and contributed to sustainable campus living align with SDG 11: Sustainable Cities and Communities.

27: For future improvements, I recommend:

Strengthening network connectivity using GSM or Ethernet.

Adding administrative features for usage analytics and maintenance alerts.

Expanding the hardware to a multi-locker design for higher volume.

Testing with a wider demographic for commercial scalability.

31: The Locking Mechanism

Current: Servo Motor.

The Upgrade: Electronic Solenoid Latch (12V or 24V).

For the prototype, the servo demonstrates the logic of unlocking. However, in a commercial setting, servo motors are easily forced open and consume power to hold a position. I would replace this with a 12V Solenoid Lock. These are the standard for Amazon Lockers.

They use a spring-loaded latch that snaps shut automatically (fail-secure) and only requires a short pulse of electricity to open, which is more energy-efficient and physically secure.

32: The Microcontroller (The Brain)

Current: ESP32 Development Board.

The Upgrade: Custom PCB (Printed Circuit Board) incorporating an ESP32-WROOM module.

The development board is great for testing, but pin headers can vibrate loose. For mass production, I would design a custom PCB.

This allows me to solder the ESP32 module directly to the board (SMD), integrate the relays for the locks, and add voltage protection circuits. This reduces the size and eliminates wiring failures.

33: User Input (Keypad)

Current: Membrane Keypad.

The Upgrade: IP65 Rated Stainless Steel Matrix Keypad.

Membrane keypads wear out quickly and are easy to peel off. I would use a metal, vandal-resistant keypad (like those on ATM machines). These are water-resistant (IP65 rated) and can withstand heavy daily usage and rough handling by students.