

Predictive RFID Tour Monitoring Dashboard Integrating Multimodal Machine Learning and Contextual Data

Heng Jin Koey¹, Siti Zuraidah Ibrahim^{2*}, Ismahayati Adam³,

Megat Syahirul Amin Megat Ali⁴, Rashidah Che Yob⁵, Sugchai Tantivivat⁶, Sazwan Syafiq Mazlan⁷

Faculty of Intelligent Computing (FKC), Universiti Malaysia Perlis (UniMAP), Perlis, Malaysia^{1, 2}

Faculty of Electronic Engineering and Technology (FKTEN), Universiti Malaysia Perlis (UniMAP), Perlis, Malaysia^{3, 5}

Microwave Research Institute, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia⁴

Faculty of Industrial Education and Technology, Rajamangala University of Technology Srivijaya (RUTS), Songkhla, Thailand⁶

Faculty of Engineering, Universiti Pertahanan Nasional Malaysia, Kuala Lumpur, Malaysia⁷

Abstract—Traditional tourism monitoring systems rely on manual tracking and overlook dynamic factors influencing travel duration. To address this limitation, this research presents an intelligent web dashboard integrating RFID infrastructure and multimodal machine learning to assess the risk of delay in tourist activities. The project encompasses the design and physical prototyping of a Radio Frequency Identification (RFID) based checkpoint system. This prototype demonstrated functional detection of members upon entry and exit, providing an automated identification layer. To predict delays, the system processes two diverse datasets. A biometric gait dataset facilitates human speed prediction utilizing age, gender, body mass, body size, and shoe size. Simultaneously, a vehicle dataset provides environmental context through travel distance, weather ratings, traffic conditions, and vehicle classification. The study evaluates Ridge Regression, Random Forest, and Gradient Boosting Regressor algorithms for human speed prediction, while Random Forest, Gradient Boosting Regressor, and Long Short-Term Memory (LSTM) networks are utilized for vehicle speed prediction. Evaluation revealed that the Gradient Boosting Regressor achieved the lowest error metrics across both datasets. This superior model was exported and implemented into a Python Flask backend to generate live transit time estimations. The dashboard provides live visibility into member locations and implements automated missing member notifications. This allows tour organizers to respond promptly to movement irregularities without manual tracking. Overall, the results demonstrated the feasibility of the proposed automated tour group monitoring system. However, the current prototype was validated through controlled functional testing, and further multi-user and field-based evaluation is required to establish its scalability and generalizability for larger smart tourism environments.

Keywords—RFID; predictive analytics; multimodal machine learning; contextual data; smart tourism

I. INTRODUCTION

Tour group management often requires continuous coordination of participants during movement between destinations, especially when groups are required to gather at specific checkpoints before continuing to the next activity. Traditional approaches using manual roll calls, name lists, and verbal confirmations need direct human intervention and may

result in inaccurate or delayed attendance information. Such methods are time-consuming and susceptible to human error, particularly when attendance must be repeatedly verified throughout a journey [8]. These limitations become more significant as the number of participants and checkpoints increases, since it becomes difficult to accurately verify the presence of every individual when the group moves between different locations. The absence of an automated tracking mechanism may result in members being unintentionally left behind, leading to delays, confusion, dissatisfaction, and potential safety issues [11].

Digital communication tools and mobile messaging applications can provide an alternative method for sharing attendance updates; however, these approaches still depend heavily on manual input and coordination. Group leaders may need to request individual responses from participants to confirm their arrival or departure, while participants may forget to respond, provide inaccurate information, or experience connectivity problems. Such methods also provide limited automated verification and structured historical records for confirming whether a participant has passed through a specific checkpoint. Consequently, when a member arrives late or fails to reach a designated location, group leaders may have difficulty determining whether the member is still travelling, delayed, or missing. This creates a conflict between maintaining the planned travel schedule and ensuring that all participants remain accounted for throughout the journey.

Radio Frequency Identification (RFID) provides an automated identification mechanism by assigning a unique identifier to each participant. RFID-based systems can perform automatic data capture and reduce the need for manual attendance verification. Instead of calling names or checking participant lists individually, members can present their RFID tags at designated checkpoints, allowing their attendance and checkpoint information to be recorded electronically. RFID can therefore provide structured identification records that can subsequently be transmitted to a centralized monitoring platform. Previous work has demonstrated the use of RFID with embedded hardware and cloud-based data logging to automatically record participant attendance, while RFID-based

*Corresponding author.

tracking systems can also be extended to provide centralized monitoring of member movements.

However, RFID identification alone does not provide sufficient support for tour management. A conventional RFID checkpoint system records whether a participant has been detected at a particular location but does not directly determine whether the participant is likely to arrive at the next checkpoint on time. Furthermore, when several checkpoints are operating simultaneously, the collected information needs to be synchronized and presented in a form that can be easily interpreted by tour leaders. Data synchronization and communication latency are therefore important considerations in real-time tracking systems, particularly when multiple devices are required to transmit information to a centralized platform [11]. A monitoring system must therefore not only collect RFID data but also provide timely and reliable updates to support decision-making during a journey.

Collected checkpoint records generally represent historical or status information and may require additional analysis to determine potential delays. A participant's latest recorded checkpoint alone does not indicate how long the participant will need to reach the next destination. Machine learning can complement RFID-based monitoring by learning relationships between participant characteristics, transportation conditions, and travel speed. The predicted travel performance can subsequently be used to estimate travel time and compare the expected arrival time with the checkpoint deadline. This provides predictive information that can assist group leaders in identifying participants who may be at risk of arriving late rather than relying solely on a missing-member status after the deadline has passed.

Therefore, this study proposes an RFID-Based Tour Group Tracking and Predictive Monitoring System that integrates RFID checkpoint identification, a cloud-based database, real-time web monitoring, automated missing-member notification and machine learning-based travel prediction. The RFID checkpoints record member identification at designated locations and transmit the records to a centralized database, while the dashboard synchronizes the information and presents the current member status to administrators and tour leaders in real time. The proposed architecture therefore combines physical identification, cloud data management, real-time communication, and predictive analysis within a single monitoring system.

II. LITERATURE REVIEW

Radio Frequency Identification (RFID) uses radio waves to identify and track tagged objects without physical contact or line-of-sight, a capability that traces back to World War II friend-or-foe radar identification [1]. A typical RFID system consists of a transponder (tag), a scanning antenna, and an emitter–receiver that together form the reader, or interrogator [2]. The reader broadcasts a radio-frequency signal to power and activate the tag, then demodulates the returned signal into digital data for downstream processing [3]. Passive high-frequency (HF) tags are preferred for short-range, contact-triggered check-in/check-out applications because they require no onboard battery, resist interference in crowded environments, and remain cost-effective for multi-checkpoint

deployment, unlike ultra-high-frequency (UHF) or microwave tags whose longer read range would allow a member to pass a checkpoint undetected.

Research found that commercial group-management platforms illustrate current approaches to traveler tracking. VNFCPass assigns each traveler an NFC bracelet or badge, letting operators import participant lists, verify presence with a single tap, and manage multiple groups in parallel [4]. TourTrax combines GPS positioning with checkpoint scanning and two-way communication to provide real-time alerts and task-completion reporting for patrol-style tour operations [5]. While effective for group visibility, neither platform incorporates predictive analytics to flag members at risk of missing a scheduled deadline, a gap this study addresses. Several studies apply RFID and related IoT sensing to attendance and tracking problems. Farag and Abouelela [6] implemented a low-cost passive-RFID attendance system that logs check-in events with minimal hardware overhead. Sunder and Am [7] developed an RFID-based asset-management platform for tracking equipment across a facility. Gopinath et al. [8] proposed a real-time group-coordination system that alerts operators when a member deviates from the group. Belka et al. [9] designed a Bluetooth Low Energy (BLE) indoor-positioning system with overlap-resistant beacon placement for tourism navigation, reporting improved localisation accuracy over single-beacon layouts. Hutabarat [10] combined RFID wristbands for indoor detection with smartphone GPS for outdoor tracking and IP cameras for visual confirmation across a large recreational area, while Yadav [11] logged ESP32/RC522 RFID scans directly to Google Sheets for lightweight cloud attendance recording. Camera-based alternatives that use the MATLAB face-recognition system of Pamith et al. [12], the Flutter/Wi-Fi proximity system of Mahadik et al. [13], and the Flask–Supabase face-recognition system of Ibezim [14] achieve tag-free identification but remain sensitive to lighting, network reliability, and privacy concerns. Collectively, these works confirm RFID as a low-latency, low-power solution for controlled indoor and semi-outdoor checkpoint tracking, though few integrate a predictive layer for anticipating schedule risk.

Beyond detection, several works apply ML/DL to strengthen the predictive value and reliability of sensor data. Abbass et al. [15] used a Long Short-Term Memory (LSTM) network to predict power-system voltage stability from historical grid data, reporting high accuracy but also flagging overfitting risk on limited datasets, a consideration directly relevant to the temporal regression models developed in this study. Wang [16] proposed a CNN-based adaptive-modulation and error-correction scheme for chip-less RFID tags, showing that convolutional pattern recognition can raise signal-to-noise ratio and reduce bit-error rate compared with PSO- and 1D-CNN-based decoding, demonstrating that deep learning can improve the reliability of RFID-derived data even at the physical-signal level. In a related embedded-IoT context, Tomari et al. [17] trained YOLOv8/v9/v10 detectors to identify recyclable items on a CPU-only automated recycling machine, selecting YOLOv8-m as the best accuracy–efficiency trade-off for real-time inference on constrained hardware; the same accuracy-versus-resource trade-off suggests this project's

choice of lightweight regression models (Ridge Regression, Random Forest, Gradient Boosting) alongside LSTM for a resource-constrained Flask backend.

The reviewed literature confirms RFID as a mature, low-power solution for indoor/controlled-environment tracking [1]–[3]; the asset-management paper [4], [7] and RFID/IoT attendance systems [6]–[14] can reliably log checkpoint events. Separately, ML/DL techniques have improved prediction accuracy and signal reliability in adjacent IoT domains [15]–[17]. However, none of the reviewed group-tracking or RFID-attendance systems combine real-time checkpoint logging with a machine-learning layer that predicts individual travel risk, such as the likelihood of missing a schedule deadline, based on biometric, distance, and contextual features. This gap motivates the proposed RFID-based tour-tracking dashboard, which integrates checkpoint-level RFID logging with a two-stage ML pipeline for walking/vehicle speed and deadline-risk prediction.

III. METHODOLOGY

A. Project Design Flow

Each tour member is assigned a unique passive RFID tag. At each checkpoint, an RC522 RFID reader connected to an ESP32 detects the tag UID when the member taps the card to the reader. The ESP32 validates the received UID, filters repeated scans, associates the scan with the corresponding checkpoint, and records the timestamp. A valid scan is then transmitted through the ESP32's Wi-Fi connection to the Supabase backend in JSON format. The received RFID data is processed through the Supabase REST API and stored in the PostgreSQL database.

The React-based web dashboard retrieves and synchronizes these records using Supabase. Supabase Realtime enables new RFID events and status changes to be reflected on the dashboard without requiring manual page refreshes. The dashboard displays each member's status and checkpoint information, allowing administrators and group leaders to monitor the tour in real time. When a member fails to reach the designated checkpoint before the configured deadline, the system updates the member's status to "Missing" and generates an alert.

The system includes predictive analysis to estimate travel performance and potential delay. Machine learning models are trained using participant and transportation-related data to predict walking speed and vehicle speed. When a user requests a risk assessment, the React dashboard sends the required member and route information to the Flask backend through a RESTful API. The backend processes the input using the trained model and estimates the travel speed and corresponding travel time. The predicted travel time is compared with the predefined checkpoint deadline to determine the member's potential delay risk. The prediction results are then returned to the dashboard and presented through the Analytics Page.

B. System Architecture

The proposed system is an integrated hardware-software platform consisting of four main layers: embedded hardware, cloud database, frontend application and machine learning

backend, as shown in Fig. 1. The overall architecture enables RFID data collected at physical checkpoints to be transmitted to the cloud database, synchronized with the web dashboard in real time, and used for travel-time and delay-risk prediction.

The embedded hardware layer collects RFID checkpoint events, while the cloud database layer stores and manages system data. The frontend application provides interfaces for member management, real-time monitoring, route configuration, and analytics. The machine learning backend processes prediction requests from the frontend and provides travel-time and delay-risk results. These layers work together to provide an automated system for monitoring tour group movements and identifying potential delays.

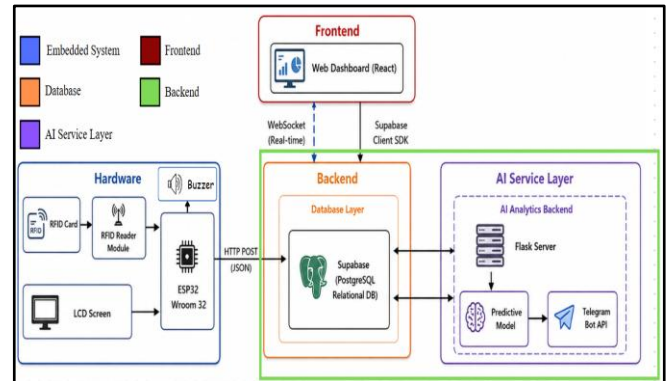


Fig. 1. Block diagram of the integrated RFID-based tracking and monitoring system for tour group members.

C. RFID Checkpoint Hardware Implementation

The RFID checkpoint was designed as a portable identification station that can be deployed at different locations during a group journey. The ESP32-WROOM-32 was selected as the central microcontroller because it provides integrated Wi-Fi connectivity and supports the communication interfaces required to connect the RFID reader and display. The RC522 reader was selected as the HF RFID reader, while passive RFID cards were used as member identification tags. The use of passive tags eliminates the requirement for individual tag batteries and is therefore suitable for repeated use by tour participants.

The RC522 communicates with the ESP32 through the SPI interface. The RFID reader is connected to the ESP32 through the corresponding power, reset, clock, master-in/slave-out, master-out/slave-in, and chip-select lines. An I²C LCD is connected separately to provide visual feedback, while the buzzer is connected to a GPIO output for audible confirmation. The hardware arrangement allows the ESP32 to simultaneously communicate with the RFID reader and peripheral feedback devices.

Fig. 2 shows the flowchart of mode operations at the checkpoint. Two operating modes are implemented at the RFID checkpoint: the registration mode and tracking mode. During registration, an administrator scans an RFID card and associates its UID with the selected participant profile. During tracking, the system verifies whether the scanned UID has been registered. For a valid UID, the system records the participant's identity, checkpoint, and timestamp and provides successful-

scan feedback through the LCD and buzzer. The RFID scan record is subsequently transmitted to the cloud database for further processing.

This approach allows the physical checkpoint to function independently as a data acquisition device while the database and web application handle storage, visualization, and higher-level analysis.

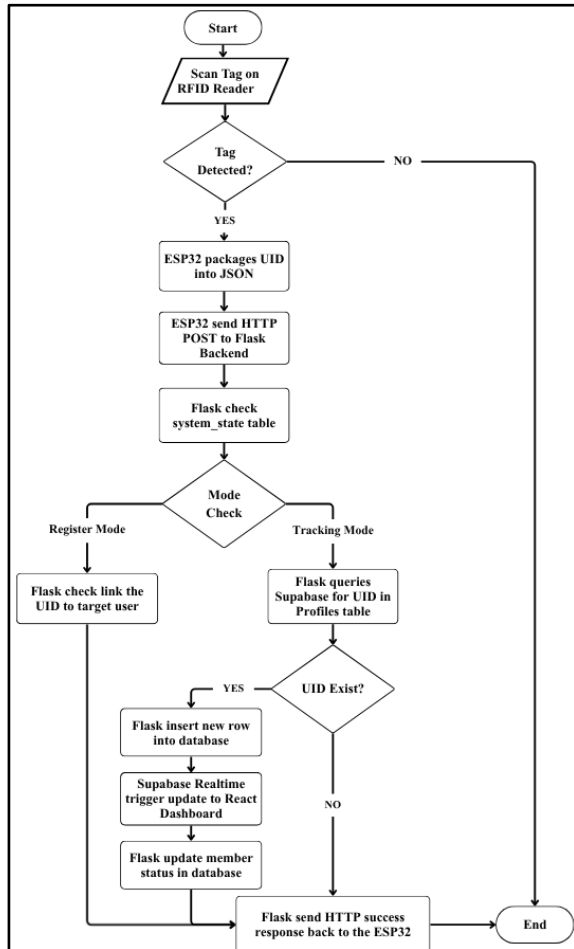


Fig. 2. Flowchart of track and register mode at checkpoint.

D. Cloud Database and Real-Time Monitoring

The Supabase PostgreSQL database serves as the central storage layer for the system. The database contains structured information covering participant profiles, RFID identifiers, travel groups, checkpoints, travel sessions, and system states. RFID scan records contain the information required to determine a participant's most recent confirmed checkpoint and movement history.

After an RFID scan is successfully processed by the ESP32, the scan record is inserted into the database through an HTTP request. The database update then becomes the source event for the real-time monitoring system. Instead of requiring the dashboard to repeatedly reload the database, Supabase Realtime is used to detect changes to subscribed tables and propagate them to connected clients.

The React dashboard maintains subscriptions to relevant database changes. When a new RFID record is inserted, or an existing participant status is updated, the corresponding dashboard components refresh their displayed information. This allows the current checkpoint, member status, and group attendance information to be presented without requiring manual page refreshes.

The dashboard also provides administrative functions for creating and modifying members, groups, checkpoints, and routes. Administrators can configure checkpoint locations, assign members to groups, and create a designated gathering or exit checkpoint. Group leaders can use the monitoring interface to observe participant status and identify members who have not reached the required checkpoint.

E. Real-Time Communication and Missing-Member Detection

Two communication mechanisms were considered for synchronizing checkpoint changes with the dashboard which are HTTP polling and WebSocket communication. In the polling approach, the client periodically sends HTTP requests to check whether new data is available. The system used a three-second polling interval during performance evaluation. Although polling is straightforward to implement, updates can remain undetected until the next polling cycle, and shorter polling intervals increase the number of requests and network traffic.

The proposed system therefore uses Supabase Realtime WebSocket communication for the deployed monitoring function. Supabase Realtime uses PostgreSQL logical replication to detect subscribed INSERT and UPDATE operations. When a relevant database record changes, the update is pushed to connected clients through a persistent WebSocket connection. This event-driven approach is suitable for RFID checkpoint monitoring because scan events occur irregularly and may require immediate notification.

A missing-member detection mechanism is implemented using the checkpoint deadline defined for a travel route. When the system determines that a participant has not reached the designated checkpoint within the required time, the participant status can be updated to "Missing". The status is displayed on the dashboard, and a notification can be generated for the administrator. The notification provides information such as the participant's name, group and contact information to support follow-up action.

F. Route Management and Travel-Time Prediction

The system includes a route-management module that allows administrators or group leaders to define a sequence of checkpoints before a journey begins. Each route is divided into segments, with each segment containing a starting checkpoint, destination checkpoint, and transportation mode. The transportation modes supported include walking, vehicle, and bus.

After a route has been created, the user can select a participant and initiate a risk prediction. The system sends the participant's profile information, route information, checkpoint sequence, and transportation modes to the Flask prediction

backend. The backend calculates the expected travel duration for each route segment according to its assigned transportation mode. The predicted segment durations are then combined to obtain the estimated total journey time.

For a designated exit or gathering checkpoint, the estimated arrival time is compared with the configured deadline. The difference between the predicted arrival time and the required arrival time is used to determine whether the participant is likely to be late. The resulting prediction is returned to the React dashboard, where the estimated journey duration, remaining time, risk percentage, route status, and segment-level prediction are displayed.

This prediction process allows the system to provide information before and during a journey rather than relying solely on historical RFID records. When a new RFID scan is received during the journey, the participant's latest confirmed checkpoint can be used to update the prediction according to the remaining route. The following equations define the estimated travel time and the corresponding deadline-utilization ratio.

$$T_{estimated} = \frac{D}{V_{predicted}} \quad (1)$$

where:

- D = remaining travel distance
- $V_{predicted}$ = predicted walking or vehicle speed
- $T_{estimated}$ = estimated time required to reach the destination/checkpoint

and

$$R = \frac{T_{needed}}{T_{remaining}} \times 100 \% \quad (2)$$

where:

$T_{remaining}$ = time remaining until the relevant deadline

R = risk-utilization ratio, expressed as a percentage

G. Machine Learning Model Development

Fig. 3 shows the pipeline of the machine learning process. The machine learning component was developed to estimate participant travel speed and support delay-risk assessment. The development process consisted of data collection, preprocessing, dataset splitting, model training, hyperparameter tuning, performance evaluation, model selection and deployment. The full pipeline follows the process of preparing the datasets, training multiple candidate models, evaluating their predictive performance, and deploying the best-performing model into the integrated tracking system.

1) *Walking speed prediction:* The walking-speed dataset contains 1,274 participant records of participant characteristics including age, gender, body mass, body size, shoe size, and measured gait speed. The dataset was cleaned before model development by removing duplicated records, standardizing attribute names, encoding categorical gender information, and handling missing values. Missing values were processed using

a two-stage imputation approach based on participant grouping.

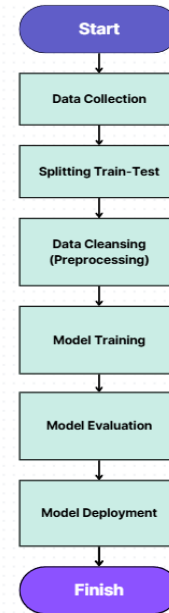


Fig. 3. Pipeline of machine learning process.

The processed dataset was divided into training and testing subsets using a random 80:20 train-test split, resulting in 1,019 records for training and 255 records for testing. Feature scaling was applied to the numerical inputs where required, with the scaler fitted using the training data to avoid information leakage into the testing set.

Three regression algorithms were evaluated using Ridge Regression, Random Forest Regression, and Gradient Boosting Regression. These models were selected to compare linear regression with ensemble tree-based approaches capable of representing nonlinear relationships between participant characteristics and walking speed.

2) *Vehicle speed prediction:* A separate vehicle travel dataset containing 146 trip records was used for vehicle-speed prediction. Unlike the human gait dataset, the vehicle dataset consists of only 146 trip records. A random 80:20 train-test split was initially performed, resulting in 116 records for training and 30 records for testing. Due to the relatively small number of training samples, a data augmentation technique was subsequently applied to expand the training set and provide additional samples for model development. The testing set was isolated before augmentation and remained completely untouched throughout the training process. The input features include travel distance, traffic condition, weather condition, vehicle type, day, and travel hour. These variables were selected to represent the environmental and temporal factors that may influence vehicle travel speed.

The dataset was preprocessed before training and divided into training and testing subsets. Three models were evaluated for vehicle-speed prediction using Random Forest Regression, Gradient Boosting Regression, and Long Short-Term Memory

(LSTM). Random Forest and Gradient Boosting were used as ensemble regression approaches, while LSTM was included to evaluate whether a recurrent neural network could capture patterns in the available travel data.

The models were evaluated using regression performance metrics, particularly MAE and the model achieved the lowest prediction error and was subsequently selected as the deployed vehicle-speed model. The selected model was serialized and integrated into the Flask backend together with the walking-speed model.

3) *Evaluation metrics*: In machine and deep learning, selecting appropriate evaluation metrics is important to measure the predictive accuracy of a trained model and to make fair comparisons across different algorithms. For regression tasks where the output is a continuous numerical value, three widely adopted metrics are Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2). Below are the equations used [18]:

a) *Mean absolute error*: MAE calculates the average errors between predicted and actual values without considering their direction in the dataset [18].

$$MAE = \frac{1}{n} \sum_{i=1}^N |y_i - \hat{y}_i| \quad (3)$$

b) *Root mean squared error*: RMSE is the square root of MSE (Mean Squared Error); it measures on how far the variation in the dependent variable is explained by the independent variables. Lower RMSE indicates less error.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^N (y_i - \hat{y}_i)^2} \quad (4)$$

c) *R-squared*: R-squared is also known as the coefficient of determination which measures how well a model fits the data and the power to predict future outcomes. The ranges are capping from 0 to 1, where 1 indicates a perfect fit of the model to the data.

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} \quad (5)$$

4) *Prediction deployment*: After evaluation, the selected models were exported using Joblib and stored in the Flask backend, avoiding the need for retraining during each journey. When a prediction is requested, the React dashboard sends the participant and route data to the Flask API. The backend selects the appropriate model based on transportation mode, processes the data and predicts travel speed. The predicted speed and route distance are then used to estimate travel time, calculate total journey duration, and determine delay risk against the checkpoint deadline.

This architecture separates data acquisition, storage, user interaction and prediction, allowing the RFID hardware to remain lightweight while machine learning processing is handled by the backend. It also allows models to be updated independently without modifying the physical RFID system.

IV. RESULT AND DISCUSSION

A. Result from RFID Reader

Fig. 4 shows the circuit setup of the proposed work. When an RFID tag is detected by the reader, the buzzer beside the ESP32 microcontroller will sounds to notify the user that the scan is detected, along with visual feedback on the LCD screen displaying “Scanned” to confirm the scanning action was successful; the status is updated to the current checkpoint.

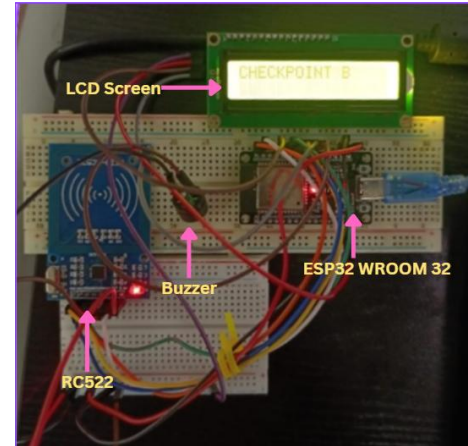


Fig. 4. Circuit setup.

Table I shows the scan detection test of the proposed work. From the results, the RFID reader can read the tag at a maximum of 4 cm from the reader. When the card tag is held further away from the reader, the card is not detected; no scan is triggered until the card enters the range of the reader scanning field. This is because the operating frequency of the RFID MFRC522 reader operates at high frequency of 13.56 MHz, which relies on inductive coupling for communication between the reader and the passive RFID tag. The magnetic field strength decreases rapidly as the distance increases, resulting in a relatively short operating range. Because passive RFID tags have no internal power source, they rely entirely on the reader's magnetic field to induce enough voltage in the tag antenna to power the chip.

TABLE I. SCAN DETECTION TEST

Test Case	Input	Expected Result	Pass/Fail
Card within range	RFID card held 1cm from reader	UID detected	Pass
Card at max range	RFID card held 2 cm from reader	UID detected	Pass
Card at max range	RFID card held 3 cm from reader	UID detected	Pass
Card at max range	RFID card held 4 cm from reader	UID detected	Pass
Card at max range	RFID card held 5 cm from reader	UID detected	Fail

As the magnetic field weakens rapidly with distance, the tag cannot retrieve sufficient energy to operate or send back its UID. This principle follows the equation of the inverse-cube law, where:

$$H \propto \frac{1}{d^3} \quad (4)$$

H= magnetic field strength

d= distance from the reader antenna

The greater the distance from the antenna, the weaker the magnetic field strength.

1) *Tracking mode*: Fig. 5 shows the serial monitor output when a member's RFID card was scanned at Checkpoint A. The ESP32 successfully initialized the RC522 RFID reader and established a Wi-Fi connection with the Supabase backend before entering tracking mode. When the RFID card was detected, the RC522 captured the card UID, and the ESP32 verified it against the registered member records in Supabase. The scanned UID was successfully matched to the registered member, CCY, and the system recorded the member as checked in at Checkpoint A. The ESP32 then updated the corresponding movement record in the Supabase database with the checkpoint and timestamp information.

```
Connected! IP: 10.163.125.109
=====
RUNNING AS CHECKPOINT: B
=====
Scan card...

Card UID: 177C2E03
Checking System State...
TRACKING MODE
Found: CCY | Status: D
Card Processed: CCY → Checked-in at Checkpoint B
Database trigger updated locations & tracking logs smoothly.
```

Fig. 5. Arduino serial monitor output during tracking scan.

The database update was propagated through Supabase Realtime WebSocket, allowing the React dashboard to update the member's status immediately. As shown in Fig. 6, the member's previous checkpoint status was changed from Checkpoint D (Queensbay Mall) to Checkpoint A (Universiti Malaysia Perlis), confirming successful RFID detection, database update, and real-time dashboard synchronization.

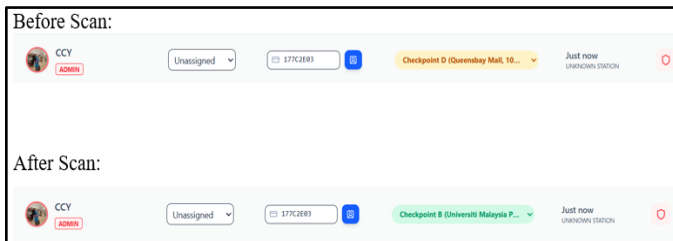


Fig. 6. Member status update to Checkpoint B in web dashboard.

2) *Register mode*: As shown in Fig. 7, an administrator can register or update a member's RFID card by entering the UID manually or selecting the Scan button. The system switches to REGISTER MODE, where the ESP32 captures the RFID UID and links it to the selected member's profile in the Supabase Profiles table. After successful registration, the system returns to TRACKING MODE, while the dashboard changes from "Waiting for Scan" to "Card Scanned", confirming successful registration.

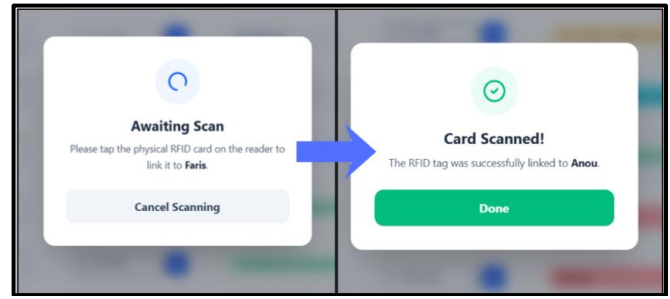


Fig. 7. After the card is scanned in the web dashboard.

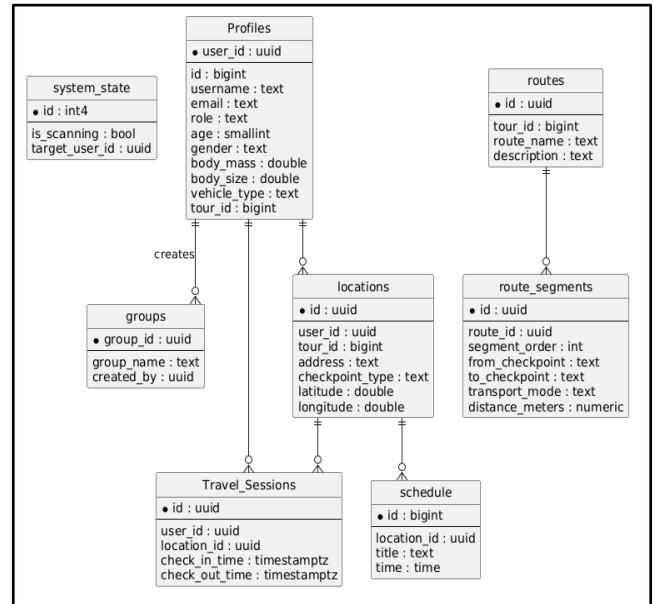


Fig. 8. Entity relationship diagram of RFID-based travel group tracking and monitoring system.

Fig. 8 shows the Entity Relationship Diagram (ERD) of the RFID-based Travel Group Tracking and Monitoring System. The Profiles table acts as the central entity, storing member information such as biometric data, RFID tag number and vehicle type. It is linked to the Groups, Travel_Session and Locations tables for member management, travel status, and location tracking. The Locations table records checkpoint scans, coordinates, timestamps and calculated distances to the exit checkpoint, supporting the Analytics dashboard and risk prediction. The Checkpoints table stores the location and exit status of each checkpoint. The Routes table defines each route, while Route_Segments stores its sequence, checkpoints, transportation mode, and travel distance. The Schedule table is linked to locations and stores planned activities and expected arrival times, which support system monitoring and risk prediction.

B. Supabase Table Relationship Design

C. Web Dashboard Test Result

Tables II, III, IV, and V show the member, group, and route analytics management page tests, respectively.

TABLE II. MEMBER MANAGEMENT PAGE TEST

Feature Tested	Expected Behaviour	Observed Behaviour	Result
RFID tag registration via scan button	New card UID links to selected member profile	Card UID written to database table and user interface shows output.	Pass
Manual RFID UID entry	Admin can type UID directly without scanning	UID saved to database on submit	Pass
Status edits by admin	Admin changes member status	Status updated in Supabase and UI refreshed	Pass
Status edits attempt by member role	Edit controls should be disabled for non-admin users	Edit fields not interactive for member role	Pass
Role toggle (Member to Admin)	Clicking action icon switches user role	Role updated and shield icon highlighted in red	Pass
Real-time update from external change	Table reflects changes made by another admin session	New status appeared without manual refresh	Pass

TABLE III. GROUP MANAGEMENT PAGE TEST

Feature Tested	Expected Behaviour	Observed Behaviour	Result
Create new group	New team card appears after naming	Team created and visible immediately	Pass
Assigning member to group	Selected member added to team list	Members appeared under correct team with inherited transport mode	Pass
Removed from group	Member unassigned and returned to pool	Member removed from team list instantly	Pass
Change group transport mode	All members in group update to new mode	Transport mode updated in bulk across all members	Pass
Rename group	Group name updates across all associated records	Name changed and reflected in member records	Pass

TABLE IV. ROUTE MANAGEMENT PAGE TEST

Feature Tested	Expected Behaviour	Observed Behaviour	Result
Create new route	Route saved with defined checkpoint sequence	Route appeared in route list with correct segments	Pass
Delete route	Route removed from list permanently	Route no longer appeared after deletion	Pass
Run risk prediction for member	Predicted travel time and risk percentage displayed	Segment breakdown and total journey time shown correctly	Pass
Segment transport mode variation	Each segment uses its assigned transport mode for prediction	Different predicted times shown per transport mode per segment	Pass

Across all four dashboard pages, Member Management, Group Management, Route Management, and Analytics, all test cases passed; Objective 2, which is functional webpages is achieved. Role-based access control, member and transport-mode updates, route-specific risk predictions, and transport-mode-based analytics were successfully verified. The Analytics

page also correctly identified delayed members and triggered missing-member notifications, fulfilling the automated alert requirement.

TABLE V. ANALYTICS PAGES TEST

Feature Tested	Expected Behaviour	Observed Behaviour	Result
Display walking mode predictions	Members using walking mode show predicted time and risk	Risk percentages and time needed displayed per member	Pass
Display vehicle mode predictions	Members using vehicle mode show predicted speed and risk	Risk percentages and predicted speed displayed per member	Pass
Display bus mode predictions	Members using bus mode show predicted time and risk	Risk percentages displayed correctly for bus group	Pass
Toggle between transport types	Updates displayed member list and predictions	Member list and predictions changed per selected mode	Pass
Delay risk alert trigger	Member exceeding deadline threshold flagged as delay risk	Red highlight and extra time needed shown for at-risk member	Pass
Real-time prediction update	Predictions update when a new RFID scan occurs	Risk values refreshed after simulated new scan event	Pass

D. Result from Machine Learning

Fig. 9 shows risk prediction feature estimates whether a selected member can reach the exit checkpoint before the deadline. The user selects the route segments, transportation mode, and member before running the prediction. The system then uses trained machine learning models to estimate travel time for each segment based on the member's profile and selected transport mode. The results show the total travel time, remaining time, delay risk percentage, and status (On Track or Delay Risk). A segment breakdown also shows the predicted travel time and distance for each segment, helping administrators identify potential delays and adjust the route or transportation mode before the trip.

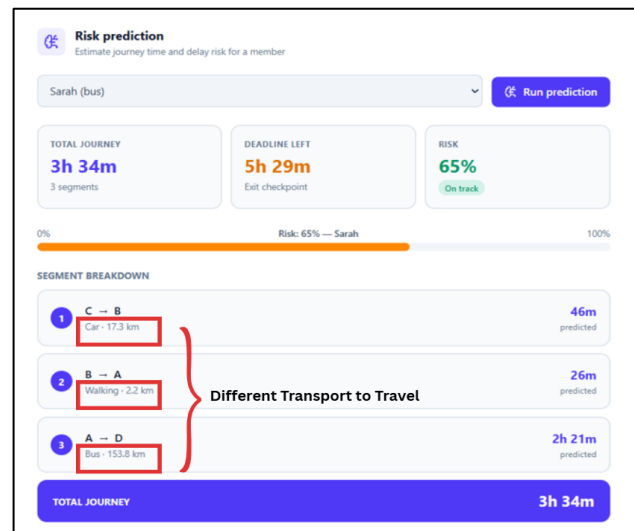


Fig. 9. Route time prediction.

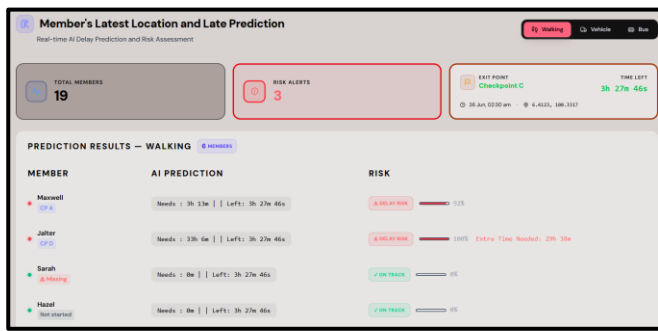


Fig. 10. Analytics page for transport using walking type member.

The Analytics Page integrates the trained machine learning models to provide real-time travel predictions. The system retrieves member biometric data, current location and remaining time from Supabase and sends them to the Flask backend for prediction. The dashboard displays each member's location status, predicted travel time, remaining time, and delay risk percentage using the formula:

$$\text{Risk} = (\text{Predicted Time Needed} \div \text{Time Remaining}) \times 100$$

Members predicted to exceed the deadline are marked as "DELAY RISK" and display the estimated extra time required. For example, in Fig. 10, three members are predicted to be late. The system updates predictions when new RFID scans or checkpoint deadline changes occur, allowing administrators to monitor members and act against potential delays.

TABLE VI. COMPARISON OF MODEL PERFORMANCE ON THE HUMAN DATASET

Metric	Random Forest	Gradient Boosting	Ridge Regression
R ²	0.9307	0.9313	0.0417
Mean Absolute Error (m/s)	0.0235 approx. (0.08 km/h)	0.0234 approx. (0.08 km/h)	0.0908 approx. (0.33 km/h)
Root Mean Square Error	0.0303	0.0302	0.1127
Training Time (s)	37.44	15.05	0.2003

As depicted in Table VI, Random Forest and Gradient Boosting achieved strong performance, with R² above 0.93, while Ridge Regression performed poorly with an R² of 0.0417 due to its inability to capture the non-linear relationship in gait speed.

Gradient Boosting achieved the lowest MAE of 0.0234 m/s and RMSE of 0.0302 m/s, closely matching Random Forest. It was also faster to train (15.05s vs. 37.44s), making it the preferred model. The low prediction error corresponds to approximately ± 1.3 seconds of travel-time uncertainty per 100 m, indicating sufficient accuracy for reliable delay-risk assessment.

From the evaluation results, Gradient Boosting Regressor achieved the best result among these 3 models; hence, it is selected and downloaded to use as the model for the walking speed predictions and further implemented in the Flask backend later. The selected model and its scaler were then

exported as a .pkl file and integrated into the Flask backend to make real-time walking speed prediction within the prediction system.

TABLE VII. COMPARISON OF MODEL PERFORMANCE ON THE VEHICLE DATASET

Metric	Random Forest	Gradient Boosting	LSTM
R ²	0.5177	0.6222	0.0296
Mean Absolute Error (km/h)	5.0695	5.0152	6.9337
Root Mean Square Error (km/s)	6.9913	6.1875	9.9186
Total Train Time (s)	26.25	23.42	8.95

As shown in Table VII, Gradient Boosting achieved the best performance, with the highest R² of 0.6222, lowest MAE of 5.0152 km/h, and lowest RMSE of 6.1875 km/h. Random Forest performed slightly worse, while LSTM had the lowest accuracy with an R² of 0.0296. Although LSTM had the shortest training time, Gradient Boosting was selected as the final model due to its better prediction accuracy and fit to the study situation.

The Gradient Boosting MAE represents approximately 32.09% relative error, which is higher than the walking-speed model due to factors such as traffic, road conditions, and driver behaviour.

Based on the evaluation of Random Forest, Gradient Boosting, and LSTM, the Gradient Boosting model was selected due to its lowest MAE and RMSE. The final model and scaler were saved as .pkl files and integrated into the Flask backend for speed estimation.

V. CONCLUSION

All three project objectives were successfully achieved. The RFID-based prototype successfully detected and registered tour group members at designated checkpoints, fulfilling the first objective. The second objective was achieved through the development of a real-time web dashboard that monitors member locations and provides automated missing-member alerts. The third objective was fulfilled through the training and evaluation of machine learning models for walking and vehicle speed prediction, with Gradient Boosting selected for its superior performance and integrated into the Flask backend. Overall, the integration of RFID, IoT devices, cloud databases and machine learning presented the feasibility of an automated tour group monitoring system. However, limitations such as the short reading range of HF RFID, Internet dependency, and limited dataset size remain challenges for large-scale deployment.

Future work will focus on upgrading the HF RFID module to a UHF long-range reader to achieve automatic scanning without requiring users to present their cards close to the reader. The system can also be improved by implementing stronger security and a multi-trip management structure. Finally, collecting larger and more diverse datasets across different locations, transportation modes and traffic conditions will improve the accuracy and generalizability of the machine learning models.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to Universiti Malaysia Perlis (UniMAP) for providing the necessary facilities and funding to support this work.

REFERENCES

- [1] "What is RFID technology – a definition and how it works," IoT Journey, Orange, 2025.
- [2] R. Solutions, "The Basics of an RFID System," Atlas, 2025.
- [3] "RFID readers collect data from RFID tags and labels," RFID4U, 2020.
- [4] "NFC App to Manage Groups and Attendance," Global Tag, 2025.
- [5] "Real-Time Monitoring," Digital Security Concepts, TourTrax, 2023.
- [6] M. Farag and A. Abouelela, "Low-Cost Active Monitoring of Attendance using Passive RFID Technology," J. Ilmiah Teknik Elektro Komputer dan Informatika, vol. 8, no. 4, 2022, doi: 10.26555/jiteki.v8i4.25168.
- [7] S. Sunder and A. Am, "Software development for asset management using RFID," 2025, doi: 10.37421/2168-9695.2025.14.309.
- [8] M. Gopinath, S. Bodke, R. Shelake, and A. Shedge, "A Real Time Tracking System using Group Coordination," Zenodo, 2025, doi: 10.5281/zenodo.18114747.
- [9] R. Belka, R. S. Deniziak, G. Łukawski, and P. Pięta, "BLE-Based Indoor Tracking System with Overlapping-Resistant IoT Solution for Tourism Applications," Sensors, vol. 21, no. 2, p. 329, 2021, doi: 10.3390/s21020329.
- [10] D. P. Hutabarat and R. Saleh, "Human tracking system in a certain area based on IoT application," Int. J. Sci. Technol. Res., vol. 9, no. 4, 2020.
- [11] A. Yadav, "RFID-Based Attendance Management System Using ESP32 and Google Sheets," ResearchGate, 2025, doi: 10.53032/tvcr/2025.v7n2.51.
- [12] Pamith Madusanka Kumara Mehrdad Tahmasebi, and Devika Sethu, "Face Recognition based Attendance Management System," Int. J. Eng. Res. Technol., vol. 9, no. 5, 2020, doi: 10.17577/IJERTV9IS050861.
- [13] S. S. Mahadik, R. Patil, and N. Patil, "Next-Gen Attendance Tracking: Leveraging Flutter and WiFi Connectivity for Educational Efficiency," ResearchGate, 2024.
- [14] F. Ibezim, "Real-Time Web-Based Student Attendance System Utilizing Face Recognition and Supabase," ResearchGate, 2025, doi: 10.13140/RG.2.2.15090.00966.
- [15] M. J. Abbass, R. Lis, and W. Rebizant, "A Predictive Model Using Long Short-Time Memory (LSTM) Technique for Power System Voltage Stability," Applied Sciences, vol. 14, no. 16, p. 7279, 2024, doi: 10.3390/app14167279.
- [16] L. Wang, "RFID Integration with Internet of Things: Data Processing Algorithm Based on Convolutional Neural Network," Int. J. Adv. Comput. Sci. Appl., vol. 16, no. 6, pp. 812–820, 2025.
- [17] R. Tomari, A. A. Kadir, W. N. W. Zakaria, D. Das, and M. B. Azni, "Deep Learning Model for Enhancing Automated Recycling Machine with Incentive Mechanisms," Int. J. Adv. Comput. Sci. Appl., vol. 15, no. 8, pp. 813–820, 2024.
- [18] W.-M. Thor, "Regression Evaluation Metrics," APXML, 2023. [Online]. Available: <https://apxml.com/courses/getting-started-with-scikit-learn/chapter-2-supervised-learning-regression/regression-evaluation-metrics>.