Design Document for Notification Tracking Android Application

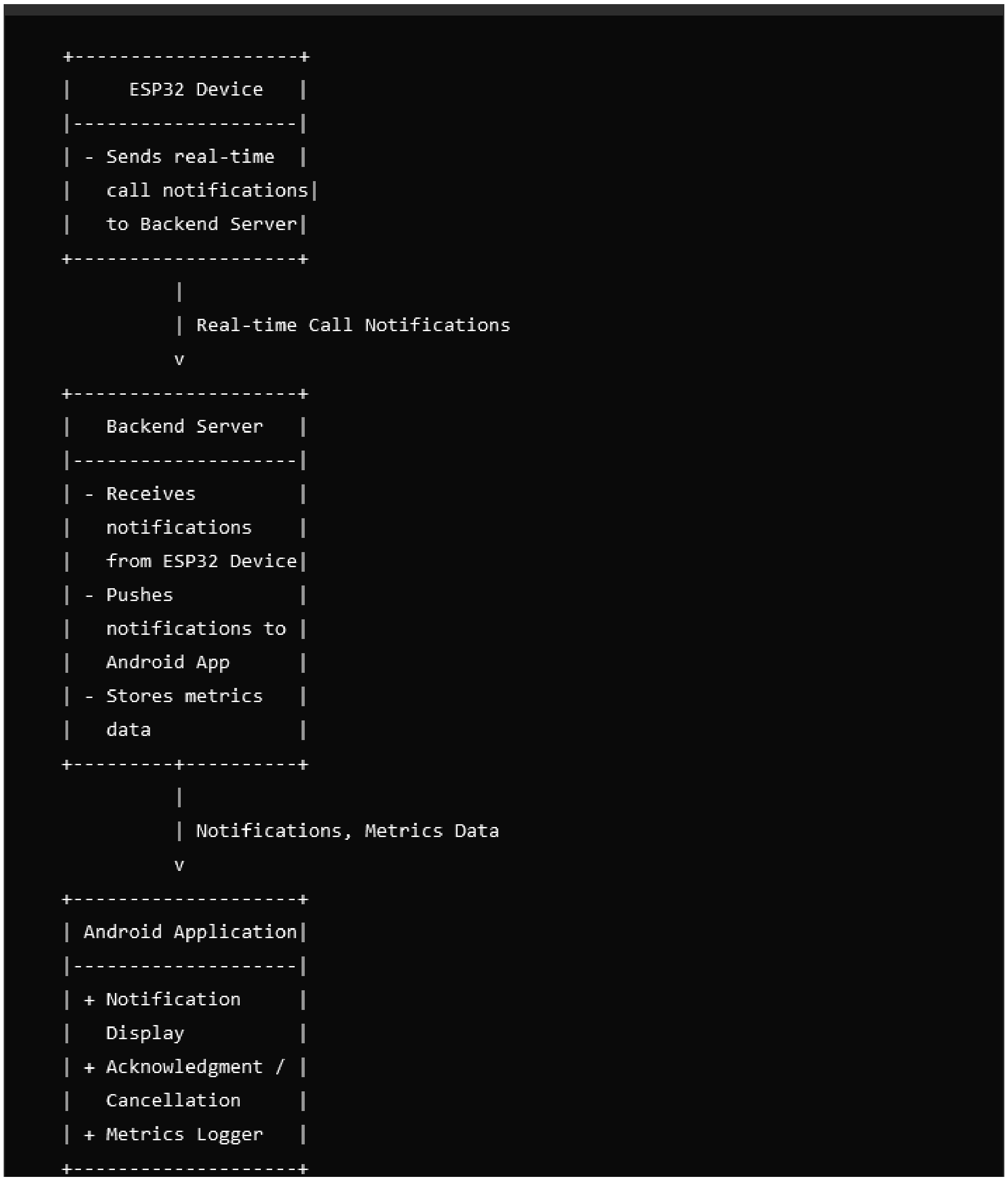
1. Introduction

This design document outlines the architecture, components, and interfaces of the Android application that provides notifications for a list of tables where calls have been made. The design ensures that notifications are cancelable, tracks the time between notification appearance and acknowledgment, and collects metrics on user interactions.

2. Architecture Overview

2.1 System Architecture

The application will follow a client-server architecture where the Android app (client) communicates with a backend server to receive real-time notifications and upload metrics data. The backend server will receive updates from the ESP32 devices and send notifications to the Android app.



```
+--------------------+
|     ESP32 Device   |
|--------------------|
| - Sends real-time  |
|   call notifications|
|   to Backend Server|
+--------------------+
          |
          | Real-time Call Notifications
          v
+--------------------+
|   Backend Server   |
|--------------------|
| - Receives         |
|   notifications    |
|   from ESP32 Device|
| - Pushes           |
|   notifications to |
|   Android App      |
| - Stores metrics   |
|   data             |
+---------+----------+
          |
          | Notifications, Metrics Data
          v
+--------------------+
| Android Application|
|--------------------|
| + Notification     |
|   Display          |
| + Acknowledgment / |
|   Cancellation     |
| + Metrics Logger   |
+--------------------+
```

ESP32 Device:

- **Function**: Sends real-time call notifications to the Backend Server.
- **Interaction**: Communicates with the Backend Server to report calls made by tables.

Backend Server:

- **Function**: Receives notifications from ESP32 devices, processes them, and pushes notifications to the Android Application. It also stores metrics data received from the Android app.
- **Interactions**:
  - **With ESP32 Device**: Receives real-time notifications.

- **With Android Application**: Sends notifications to the Android app and receives metrics data from it.

Android Application:

1. **Function**: Displays notifications to the users, allows them to acknowledge or cancel notifications, and collects interaction metrics.
2. **Components**:
    a. **Notification Display**: Shows notifications received from the Backend Server.
    b. **Acknowledgment / Cancellation**: Allows users to acknowledge or cancel notifications.
    c. **Metrics Logger**: Logs user interactions and sends this data back to the Backend Server.
3. **Interactions**:
    a. **With Backend Server**: Receives notifications and sends interaction metrics data.

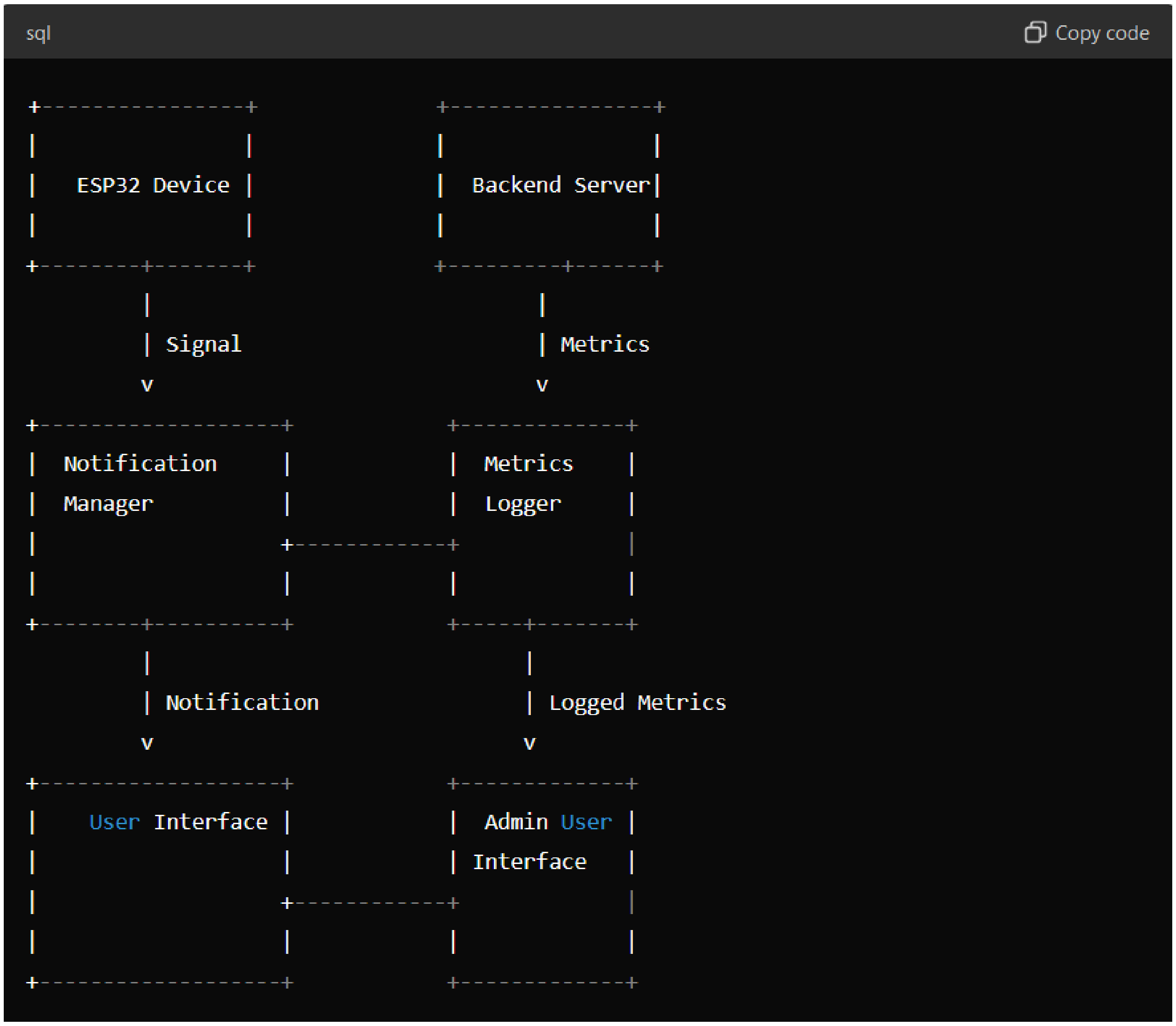
2.2 User Interaction Flow

1. **Main Screen**:
    - The user opens the app and sees the main screen with a list of active notifications.
    - If the user is an admin, they can tap the `FloatingActionButton` to access metrics.
2. **Notification Popup**:
    - When a call is made to a table, the notification popup appears.
    - The user reads the table number and time of the call from the `TextView`.
    - The user can either acknowledge or cancel the notification by tapping the respective button.
    - The app logs the user's action and the time taken to perform the action, then dismisses the popup.

Data Flow Diagram

Below is a Data Flow Diagram (DFD) for the Notification Tracking Android Application, illustrating the flow of data through the system components. The DFD is broken down into two levels: Level 0 (context diagram) and Level 1 (detailed process flows).

Level 0: Context Diagram



```
+----------------+              +----------------+
|                |              |                |
|   ESP32 Device |              |  Backend Server|
|                |              |                |
+--------+-------+              +---------+------+
         |                              |
         | Signal                       | Metrics
         v                              v
+-------------------+            +-------------+
| Notification      |            | Metrics     |
| Manager           |            | Logger      |
|                   +------------+             |
|                   |            |             |
+--------+----------+            +-----+-------+
         |                             |
         | Notification                | Logged Metrics
         v                             v
+-------------------+            +-------------+
|    User Interface |            |  Admin User |
|                   |            | Interface   |
|                   +------------+             |
|                   |            |             |
+-------------------+            +-------------+
```

Level 1: Detailed Process Flow

Process 1: Notification Generation

1. Source: ESP32 Device
2. Destination: Notification Manager
3. Data: Call Signal

Process 2: Notification Display

1. Source: Notification Manager
2. Destination: User Interface
3. Data: Notification Details (table number, time of call)

Process 3: User Interaction

1. Source: User Interface

2. Destination: Notification Manager, Metrics Logger

3. Data: Acknowledgment or Cancel Action

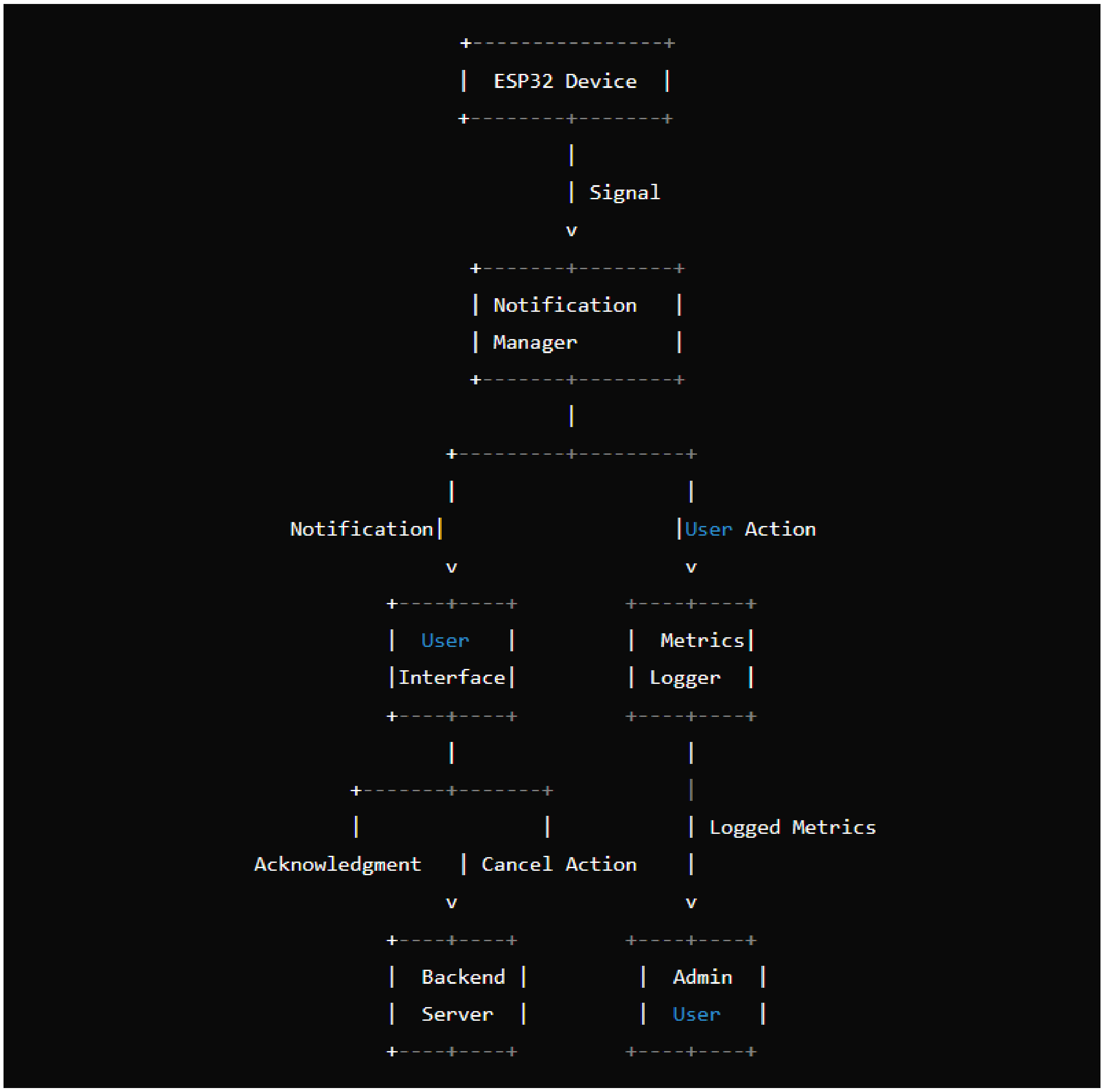
Process 4: Metrics Storage

1. Source: Metrics Logger

2. Destination: Backend Server

3. Data: User Interaction Metrics (notification displayed, acknowledged, canceled, time taken)

Below is a graphical representation of the Level 1 DFD:



```
                 +----------------+
                 |  ESP32 Device  |
                 +--------+-------+
                          |
                          | Signal
                          v
                  +-------+--------+
                  | Notification   |
                  | Manager        |
                  +-------+--------+
                          |
                +---------+---------+
                |                   |
    Notification|                   |User Action
                v                   v
           +----+----+         +----+----+
           |  User   |         |  Metrics|
           |Interface|         | Logger  |
           +----+----+         +----+----+
                |                   |
        +-------+-------+           |
        |               |           | Logged Metrics
Acknowledgment    | Cancel Action   |
                v                   v
           +----+----+         +----+----+
           |  Backend |         |  Admin  |
           |  Server  |         |  User   |
           +----+----+         +----+----+
```

Descriptions:

1. ESP32 Device: Sends a call signal when a table requests service.

2. Notification Manager: Receives the signal and triggers a notification.

3. User Interface: Displays notifications and allows users to acknowledge or cancel them.

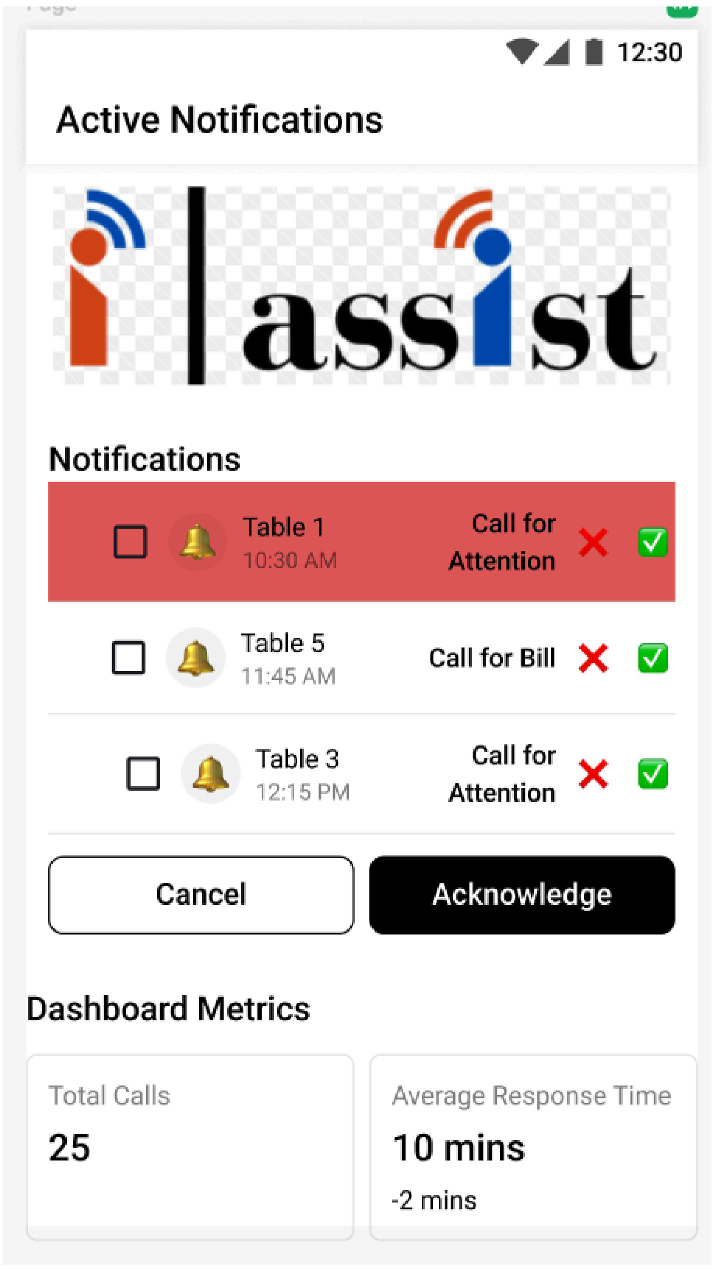
4. Metrics Logger: Logs user interactions, including the time taken to acknowledge or cancel notifications.

5. Backend Server: Stores logged metrics for access by admin users.

6. Admin User Interface: Allows admin users to view metrics and analyze performance.

3. Detailed Design

3.1 User Interface Design


12:30

Active Notifications

Notifications

☐	Table 1 10:30 AM	Call for Attention	✗	✓
☐	Table 5 11:45 AM	Call for Bill	✗	✓
☐	Table 3 12:15 PM	Call for Attention	✗	✓

Dashboard Metrics

Total Calls
25

Average Response Time
10 mins
-2 mins

3.1.1 Main Screen

Description: The main screen displays a list of active notifications and provides access for admins to view metrics.

Components:

1. **CardView**: To display active notifications.
2. **FloatingActionButton**: For admin access to metrics.

User Interaction Flow:

1. When the user opens the application, the main screen loads.
2. The `ListView` displays all active notifications in real-time.
3. Admins can tap the `FloatingActionButton` to access the metrics interface.

UI Elements:

- **ListView**:
  - ID: `listViewNotifications`
  - Description: Displays a list of notifications with details such as table number and time of the call.
  - Data Source: Binds to a data adapter that provides real-time notification data.
- **FloatingActionButton**:
  - ID: `fabAdminMetrics`
  - Description: Provides admin users access to the metrics screen.
  - OnClickListener: Navigates to the metrics interface when tapped.
-

3.1.2 Notification Popup

Description: A popup that appears when a new call notification is received, displaying the table number and time of the call, and providing options to acknowledge or cancel the notification.

Components:

1. **TextView**: Displays the table number and time of the call.
2. **Button**: For acknowledging the notification.
3. **Button**: For canceling the notification.

User Interaction Flow:

1. When a new call notification is received, the popup appears.
2. The `TextView` shows the table number and the time of the call.
3. The user can tap the "Acknowledge" button to acknowledge the notification or tap the "Cancel" button to dismiss it.
4. The action is logged and the popup is dismissed.

UI Elements:

- **TextView**:
  - ID: `textViewNotificationDetails`

    - Description: Displays the details of the call (table number and time of call).
- **Acknowledge Button**:
    - ID: `buttonAcknowledge`
    - Description: Acknowledges the notification.
    - OnClickListener: Logs the acknowledgment and dismisses the popup.
- **Cancel Button**:
    - ID: `buttonCancel`
    - Description: Cancels the notification.
    - OnClickListener: Logs the cancellation and dismisses the popup.

3.1.3 Metrics Screen (Admin Only)

- **Description**: Displays collected metrics data.
- **Components**:
    - RecyclerView: To list metrics.
    - TextView: Summarized metrics data.

3.2 Backend Server Design

Main Screen API

Endpoint: Get Active Notifications

Description: Fetches a list of active notifications to be displayed on the main screen.

URL: `/api/notifications/active`

Method: GET

Request Headers:

- `Authorization`: `Bearer <token>`

Response:

- **Status Code**: 200 OK
- **Body**:

```
{
  "notifications": [
		{
		"id": "1",
		"tableNumber": "12",
		"timeOfCall": "2024-07-01T12:34:56Z",
		"typeOfCall": "Attention"
		},
		{
		"id": "2",
		"tableNumber": "5",
		"timeOfCall": "2024-07-01T12:35:56Z",
            "typeOfCall": "Bill"
            }
  ]
}
```

Endpoint: Get Metrics

Description: Fetches metrics data for admin access.

URL: `/api/metrics`

Method: GET

Request Headers:

- `Authorization`: `Bearer <admin_token>`

Response:

- **Status Code**: 200 OK
- **Body**:

```
{
  "metrics": {
            "totalNotifications": 150,
            "acknowledgedNotifications": 120,
            "canceledNotifications": 30,
            "averageAcknowledgeTime": "45s",
            "averageCancelTime": "30s"
  }
}
```

Notification Popup API

Endpoint: Acknowledge Notification

Description: Acknowledges a notification.

URL: `/api/notifications/{id}/acknowledge`

Method: POST

Request Headers:

- `Authorization`: `Bearer <token>`

Request Parameters:

- **Path Parameters**:
  - `id`: The ID of the notification to acknowledge.

Request Body:

- **JSON**:

```
{
  "acknowledgeTime": "2024-07-01T12:36:56Z"
}
```

Response:

- **Status Code**: 200 OK
- **Body**:

```
{
  "message": "Notification acknowledged successfully."
}
```

Endpoint: Cancel Notification

Description: Cancels a notification.

URL: `/api/notifications/{id}/cancel`

Method: POST

Request Headers:

- `Authorization`: `Bearer <token>`

Request Parameters:

- **Path Parameters**:
  - `id`: The ID of the notification to cancel.

Request Body:

- **JSON**:

```
{
  "cancelTime": "2024-07-01T12:36:56Z"
}
```

Response:

- **Status Code**: 200 OK
- **Body**:

```
{
  "message": "Notification canceled successfully."
}
```

Error Handling

Error Response:

Status Code	**Body**
400 Bad Request	{ "error": "Invalid request parameters." }
401 Unauthorized	{ "error": "Unauthorized access." }
404 Not Found	{ "error": "Notification not found." }
500 Internal Server Error	{ "error": "An internal server error occurred. Please try again later." }

3.3 Database Design

This database design ensures efficient storage and retrieval of notifications, user actions, and aggregated metrics for the notification tracking application.

Tables

1. **Notifications**
2. **UserActions**
3. **Metrics**

Table 1: Notifications

Purpose: Stores information about each notification.

Column Name	**Data Type**	**Constraints**	**Description**
id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for the notification
table_number	INT	NOT NULL	Number of the table that made the call
time_of_call	DATETIME	NOT NULL	Timestamp when the call was made

type_of_call	VARCHAR(10)	NOT NULL	Type of call (e.g., "Attention", "Bill")
status	VARCHAR(10)	NOT NULL	Status of the notification ("Active", "Acknowledged", "Canceled")

Table 2: UserActions

Purpose: Logs user actions related to notifications (acknowledgment and cancellation).

Column Name	**Data Type**	**Constraints**	**Description**
id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for the user action
notification_id	INT	FOREIGN KEY (references Notifications.id)	Identifier of the related notification
action_type	VARCHAR(20)	NOT NULL	Type of action performed ("Acknowledge", "Cancel")
action_timestamp	DATETIME	NOT NULL	Timestamp when the action was performed

Table 3: Metrics

Purpose: Stores aggregated metrics data for admin access.

Column Name	**Data Type**	**Constraints**	**Description**
id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for the metrics entry
total_notifications	INT	NOT NULL	Total number of notifications
acknowledged_notifications	INT	NOT NULL	Total number of acknowledged notifications
canceled_notifications	INT	NOT NULL	Total number of canceled notifications

average_acknowledge_time	VARCHAR(20)	NOT NULL	Average time taken to acknowledge notifications (in seconds)
average_cancel_time	VARCHAR(20)	NOT NULL	Average time taken to cancel notifications (in seconds)

Relationships

- **Notifications** to **UserActions**: One-to-Many
  - One notification can have multiple user actions (e.g., acknowledged and then canceled).
- **UserActions** to **Metrics**: Aggregated
  - User actions are aggregated to produce metrics data.

3.4 Notification Handling

3.4.1 Receiving Notifications

- **Firebase Cloud Messaging (FCM)**: Used for real-time notification delivery.
- **Service**: NotificationService to handle FCM messages and display notifications.

3.4.2 Acknowledging/Canceling Notifications

- **IntentService**: Handle user actions (Acknowledge/Cancel).
- **Metrics Logging**: Record the time and action, and send to backend.

3.5 Metrics Collection

3.5.1 User Interaction Logging

- **User Actions**: Acknowledge, Cancel.
- **Data Collected**:
  - Notification ID.
  - Action performed.
  - Timestamp of action.
  - Response time.

3.5.2 Sending Metrics to Backend

- **API Call**: Metrics data sent to `/api/metrics`.
- **Data Format**: JSON.

3.6 Security

3.6.1 Data Encryption

- **In Transit**: Use HTTPS for secure communication between app and backend.
- **At Rest**: Encrypt sensitive data stored on the server.

3.6.2 Authentication

- **User Authentication**: Implement OAuth2 for user authentication.
- **Admin Access**: Role-based access control to restrict metrics access to admin users.

4. Detailed Usability Instructions

4.1 Notification Usability

Visibility

- **High-Priority Alerts**: Notifications should appear prominently on the screen, ensuring that users do not miss them.
- **Clear Display**: Use contrasting colors and large fonts to make notifications stand out.

Details

- **Table Number and Time**: Ensure that the table number and the time of the call are clearly visible in the notification popup.
- **Readable Text**: Use a legible font size and style for all text elements.

Actions

- **Accessible Buttons**: Place the "Acknowledge" and "Cancel" buttons prominently within the notification popup, ensuring they are easy to tap.
- **Intuitive Labels**: Clearly label the buttons with descriptive text to indicate their actions.

4.2 Metrics Collection Usability

- **Automatic Logging**: Automatically log acknowledgment and cancel actions with timestamps.
- **Secure Transmission**: Ensure metrics are securely transmitted to the backend server.
- **Admin Interface**: Provide a clean interface for admin users to view and analyze metrics.

5. Development and Deployment Plan

5.1 Development Tools and Technologies

- **Android Studio**: IDE for Android development.
- **Firebase Cloud Messaging**: For real-time notifications.
- **RESTful APIs**: For communication with the backend server.
- **SQLite**: Local storage for temporary data on the Android app.

- **Spring Boot**: For backend server development.
- **MySQL**: For storing notifications and metrics data.

5.2 Deployment Plan

1. **6.Development Phase**: Develop Android app, backend server, and integration with ESP32.
2. **Testing Phase**: Conduct unit testing, integration testing, and user acceptance testing.
3. **Deployment Phase**: Deploy the backend server to a cloud service and publish the Android app on Google Play Store.

6. Appendices

6.1 Appendix A: Glossary

- **Notification**: A message displayed to the user to alert them of a call.
- **Acknowledgment**: User's action of responding to a notification.
- **Cancelable**: Ability to dismiss or remove a notification.
- **Metrics**: Data collected on user interactions, such as clicks and acknowledgment times.

6.2 Appendix B: References

- Android Development Documentation: developer.android.com
- Firebase Cloud Messaging: firebase.google.com

6.3 Appendix C: SQL Examples

Action	**SQL**
Create Notifications Table:	CREATE TABLE Notifications (id INT PRIMARY KEY AUTO_INCREMENT, table_number INT NOT NULL, time_of_call DATETIME NOT NULL, type_of_call VARCHAR(10) NOT NULL, status VARCHAR(10) NOT NULL);
Create UserActions Table	CREATE TABLE UserActions (id INT PRIMARY KEY AUTO_INCREMENT, notification_id INT, action_type VARCHAR(20) NOT NULL, action_timestamp DATETIME NOT NULL, FOREIGN KEY (notification_id) REFERENCES Notifications(id));
Create Metrics Table:	CREATE TABLE Metrics (id INT PRIMARY KEY AUTO_INCREMENT, total_notifications INT NOT NULL,

	acknowledged_notifications INT NOT NULL, canceled_notifications INT NOT NULL, average_acknowledge_time VARCHAR(20) NOT NULL, average_cancel_time VARCHAR(20) NOT NULL);

Example Data
 	
Notifications Table:
	INSERT INTO Notifications (table_number, time_of_call, type_of_call, status)
	VALUES (12, '2024-07-01 12:34:56', 'Attention', 'Active'),
	(5, '2024-07-01 12:35:56', 'Bill', 'Active');

Insert UserActions Table:	INSERT INTO UserActions (notification_id, action_type, action_timestamp)
	VALUES (1, 'Acknowledge', '2024-07-01 12:36:56'),
	(2, 'Cancel', '2024-07-01 12:37:56');

Insert Metrics Table:	INSERT INTO Metrics (total_notifications, acknowledged_notifications, canceled_notifications, average_acknowledge_time, average_cancel_time)
	VALUES (150, 120, 30, '45s', '30s');

4. Appendix D: Figma Links

https://www.figma.com/proto/WprBjAMQH0QspSXOTtOjs2/Waiter-Notification?node-id=109-163&t=2w6BXoarHbHgitcK-0&scaling=min-zoom&content-scaling=fixed&page-id=0%3A1&starting-point-node-id=109%3A163