Problem statement : In today's fast-paced work environments, employee engagement and satisfaction are crucial for maintaining productivity and fostering a positive organizational culture. However, keeping employees motivated, informed, and supported can be challenging, especially in large or geographically dispersed organizations. To address these challenges and enhance employee attention and satisfaction, we propose the implementation of an innovative "Employee Attention System" with integrated QR code functionality.

The Employee Attention System aims to provide a comprehensive platform for employees to access information, communicate with colleagues and management, participate in events, provide feedback, and access support resources seamlessly. By leveraging QR code technology, we can enhance the accessibility and usability of the system, making it easier for employees to engage with its features and functionalities.

Software/IDE used: PyCharm IDE

Code:

1. **Main Page**

```
import subprocess
import sys

from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QPushButton, QLabel

class MyWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Employee Attendance Management System')
        self.setMinimumWidth(600)
        self.setMaximumWidth(600)
        self.initUI()

    def initUI(self):
        layout = QVBoxLayout()
        title_label = QLabel("Employee Attendance Management System")
        title_label.setAlignment(Qt.AlignCenter)
            title_label.setStyleSheet("font-size: 24px; font-weight: bold; color: #fff; padding:30px")
```

```
        layout.addWidget(title_label)
        button1 = QPushButton("Admin Access")
        button2 = QPushButton("Attendance")
        button1.setStyleSheet(
            "font-size: 13px; color: white; background-color: orangered; padding: 15px 10px; border: none; border-radius: 5px;")
        button2.setStyleSheet(
            "font-size: 13px; color: white; background-color: blue; padding: 15px 10px; border: none; border-radius: 5px;")
        layout.addWidget(button1)
        layout.addWidget(button2)
        self.setLayout(layout)
        self.center()
        button1.clicked.connect(self.button1_clicked)
        button2.clicked.connect(self.button2_clicked)

    def center(self):
        screen_geometry = QApplication.desktop().screenGeometry()
        center_point = screen_geometry.center()
        window_top_left = center_point - self.rect().center()
        self.move(window_top_left)

    def button1_clicked(self):
        subprocess.Popen(['python', 'adminAccess.py'])

    def button2_clicked(self):
        subprocess.Popen(['python', 'takingAttention.py'])

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = MyWindow()
    window.show()
    sys.exit(app.exec_())
```

2. **Admin Access Panel**

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QLabel, QLineEdit, QPushButton, QMessageBox, QDesktopWidget
import sqlite3
import subprocess

class LoginPage(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Login')
        self.setMinimumWidth(300)
```

```
        self.initUI()

    def initUI(self):
        layout = QVBoxLayout()

        self.username_label = QLabel('Username:')
        self.username_input = QLineEdit()
        layout.addWidget(self.username_label)
        layout.addWidget(self.username_input)

            self.username_input.setStyleSheet("font-size: 24px; font-weight: bold;
color: #fff; padding: 5px; text-align: center;");

        self.password_label = QLabel('Password:')
        self.password_input = QLineEdit()
        self.password_input.setEchoMode(QLineEdit.Password)
        layout.addWidget(self.password_label)
        layout.addWidget(self.password_input)

            self.password_input.setStyleSheet("font-size: 24px; font-weight: bold;
color: #fff; padding: 5px; text-align: center;");

        self.login_button = QPushButton('Login')
        self.login_button.clicked.connect(self.login)
        layout.addWidget(self.login_button)
        self.login_button.setStyleSheet(
             "font-size: 15px; color: white; background-color: orangered; padding:
15px 10px; border: none; border-radius: 5px;margin-top:20px")

        self.setLayout(layout)
        self.create_database()

    def create_database(self):
        conn = sqlite3.connect('database.db')
        cursor = conn.cursor()
        cursor.execute('''CREATE TABLE IF NOT EXISTS admin (
                        id INTEGER PRIMARY KEY,
                        username TEXT UNIQUE,
                        password TEXT,
                        email TEXT
                       )''')
        conn.commit()
        conn.close()
```

```
    def login(self):
        username = self.username_input.text()
        password = self.password_input.text()

        conn = sqlite3.connect('database.db')
        cursor = conn.cursor()
            cursor.execute("SELECT * FROM admin WHERE username=? AND
password=?", (username, password))
        user = cursor.fetchone()
        conn.close()

        if user:
            subprocess.Popen(['python', 'dashboard.py'])
            self.close()
        else:
                QMessageBox.warning(self, 'Login Failed', 'Invalid username or
password')

    def centerWindow(self):
        qr = self.frameGeometry()
        cp = QDesktopWidget().availableGeometry().center()
        qr.moveCenter(cp)
        self.move(qr.topLeft())

    def showEvent(self, event):
        self.centerWindow()
        super().showEvent(event)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    login_page = LoginPage()
    login_page.show()
    sys.exit(app.exec_())
```

3. Taking Attention Panel

```
import subprocess
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout,
QPushButton, QDesktopWidget

class Attention(QWidget):
    def __init__(self):
        super().__init__()
```

```
        self.setWindowTitle('Attention')
        self.setMinimumWidth(300)
        self.setMinimumHeight(300)
        self.initUI()

    def initUI(self):
        layout = QVBoxLayout()
        self.button = QPushButton('Take Attention')
        layout.addWidget(self.button)
        self.button.setStyleSheet(
                "font-size: 15px; color: white; background-color: orangered; padding:
15px 10px; border: none; border-radius: 5px;margin-top:20px")
        self.setLayout(layout)

        self.button.clicked.connect(self.openQRCodeScanner)

    def openQRCodeScanner(self):
        subprocess.Popen(['python', 'qrCodeScanner.py'])


    def centerWindow(self):
        qr = self.frameGeometry()
        cp = QDesktopWidget().availableGeometry().center()
        qr.moveCenter(cp)
        self.move(qr.topLeft())

    def showEvent(self, event):
        self.centerWindow()
        super().showEvent(event)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    Attention_page = Attention()
    Attention_page.show()
    sys.exit(app.exec_())
```

4. **QR Code detected**

```
import cv2
import sqlite3
import datetime

# Connect to the SQLite database
conn = sqlite3.connect('database.db')
```

```
cursor = conn.cursor()

# Create a table to store entry and exit times
cursor.execute('''CREATE TABLE IF NOT EXISTS employee_logs (
                id INTEGER PRIMARY KEY,
                unique_id TEXT,
                name TEXT,
                entry_time TIMESTAMP,
                exit_time TIMESTAMP
                )''')
conn.commit()

def decode_qr_code(image):
    # Load the image
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    qr_code_detector = cv2.QRCodeDetector()
    data, bbox, _ = qr_code_detector.detectAndDecode(gray)

    # If QR code detected, return the data
    if bbox is not None:
        print("QR Code detected!")
        return data
    else:
        print("No QR Code detected.")
        return None

def toggle_status(unique_id, status):
     cursor.execute("SELECT status FROM employees WHERE unique_id=?",
(unique_id,))
    cursor.fetchone()

     cursor.execute("UPDATE employees SET status=? WHERE unique_id=?",
(status, unique_id))
    conn.commit()

def record_entry_exit(unique_id):
        cursor.execute("SELECT * FROM employees WHERE unique_id=?",
(unique_id,))
    employee = cursor.fetchone()
    if employee:
        name = employee[1]
                    current_time = datetime.datetime.now().strftime('%Y-%m-%d
%H:%M:%S')
```

```
        cursor.execute("SELECT * FROM employee_logs WHERE unique_id=? AND exit_time IS NULL", (unique_id,))
        existing_entry = cursor.fetchone()
        if existing_entry:
            # Employee is exiting
            exit_time = current_time
                cursor.execute("UPDATE employee_logs SET exit_time=? WHERE unique_id=? AND exit_time IS NULL", (exit_time, unique_id))
            print("Exit recorded for employee:", name)
        else:
            # Employee is entering
            entry_time = current_time
                    cursor.execute("INSERT INTO employee_logs (unique_id, name, entry_time) VALUES (?, ?, ?)", (unique_id, name, entry_time))
            print("Entry recorded for employee:", name)
        conn.commit()
    else:
        print("Employee data not found for ID:", unique_id)

def main():
    # Capture video from default camera
    cap = cv2.VideoCapture(0)

    while True:
        ret, frame = cap.read()
        if not ret:
            print("Error: Unable to capture video.")
            break

        # Display the frame
        cv2.imshow("QR Code Scanner", frame)

        # Check for QR code
        qr_data = decode_qr_code(frame)
        if qr_data:
            print("QR Code Data:", qr_data)
            record_entry_exit(qr_data)
            break

        # Check for key press to exit
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # Release the capture
```

```
    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```

5. **Dashboard panel**

```
import subprocess
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QPushButton, QDesktopWidget
from PyQt5.QtCore import Qt

class DashboardApp(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.setWindowTitle('Dashboard')
        self.setMinimumWidth(1000)
        self.setMinimumHeight(300)

        # Create widgets
        title_label = QLabel("Dashboard")
        title_label.setAlignment(Qt.AlignCenter)
        title_label.setStyleSheet("font-size: 24px; font-weight: bold; color: #fff; padding:30px")

        button2 = QPushButton("Check Employees Details")

        button3 = QPushButton("Check Attention Detail")
        button4 = QPushButton("Add New Employee Detail")
        button2.setStyleSheet(
            "font-size: 13px; color: white; background-color: blue; padding: 15px 10px; border: none; border-radius: 5px;")
        button3.setStyleSheet(
            "font-size: 13px; color: white; background-color: blue; padding: 15px 10px; border: none; border-radius: 5px;")
        button4.setStyleSheet(
            "font-size: 13px; color: white; background-color: blue; padding: 15px 10px; border: none; border-radius: 5px;")
```

```
        layout = QVBoxLayout()
        layout.setAlignment(Qt.AlignCenter)
        layout.addWidget(title_label)

        button_layout = QHBoxLayout()
        button_layout.setAlignment(Qt.AlignCenter)
        button_layout.addWidget(button2)
        button_layout.addWidget(button3)
        button_layout.addWidget(button4)
        layout.addLayout(button_layout)
        self.setLayout(layout)

        cp = QDesktopWidget().availableGeometry().center()
        self.move(cp - self.rect().center())

        button1.clicked.connect(self.openNewAdminWindow)
        button2.clicked.connect(self.openEmployeeDetail)
        button3.clicked.connect(self.openEmployeeAttendance)
        button4.clicked.connect(self.openAddEmployee)

    def openNewAdminWindow(self):
        subprocess.Popen(['python', 'addingAdmin.py'])
    def openEmployeeAttendance(self):
        subprocess.Popen(['python', 'employeeAttendance.py'])
    def openEmployeeDetail(self):
        subprocess.Popen(['python', 'employeeDetail.py'])
    def openAddEmployee(self):
        subprocess.Popen(['python', 'addEmployee.py'])

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = DashboardApp()
    ex.show()
    sys.exit(app.exec_())
```

6. **Adding New Employee Panel**

```
import sys
import random
import string
import sqlite3
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout,
QLabel, QLineEdit, QPushButton, QMessageBox, QDesktopWidget
```

```
class AddEmployeeWidget(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Add Employee Detail')
        self.setMinimumWidth(400)
        self.setMinimumHeight(300)
        self.initUI()

    def initUI(self):
        layout = QVBoxLayout()

        self.name_label = QLabel('Name:')
        self.name_input = QLineEdit()
        layout.addWidget(self.name_label)
        layout.addWidget(self.name_input)
          self.name_input.setStyleSheet("font-size: 24px; font-weight: bold; color:
#fff; padding: 5px; text-align: center;");

        self.address_label = QLabel('Address:')
        self.address_input = QLineEdit()
        layout.addWidget(self.address_label)
        layout.addWidget(self.address_input)
         self.address_input.setStyleSheet("font-size: 24px; font-weight: bold; color:
#fff; padding: 5px; text-align: center;");


        self.mobile_label = QLabel('Mobile Number:')
        self.mobile_input = QLineEdit()
        layout.addWidget(self.mobile_label)
        layout.addWidget(self.mobile_input)
          self.mobile_input.setStyleSheet("font-size: 24px; font-weight: bold; color:
#fff; padding: 5px; text-align: center;");

        self.generateUniqueID()

        self.submit_button = QPushButton('Submit')
        self.submit_button.clicked.connect(self.submitDetails)
        layout.addWidget(self.submit_button)
        self.submit_button.setStyleSheet(
              "font-size: 13px; color: white; background-color: blue; padding: 15px
10px; border: none; border-radius: 5px;")

        self.setLayout(layout)
```

```
        self.centerOnScreen()

    def generateUniqueID(self):
            unique_id = ''.join(random.choices(string.ascii_uppercase + string.digits,
k=6))
        self.unique_id = unique_id

    def submitDetails(self):
        name = self.name_input.text()
        address = self.address_input.text()
        mobile = self.mobile_input.text()

        if not name or not address or not mobile:
                QMessageBox.warning(self, 'Missing Information', 'All fields are
required.')
            return

        self.addToDatabase(name, address, mobile)

                msg = f'Name: {name}\nAddress: {address}\nMobile Number:
{mobile}\nUnique ID: {self.unique_id}'
        QMessageBox.information(self, 'Employee Details', msg)

    def addToDatabase(self, name, address, mobile):
        conn = sqlite3.connect('database.db')
        cursor = conn.cursor()

        cursor.execute('''CREATE TABLE IF NOT EXISTS employees (
                        id INTEGER PRIMARY KEY,
                        name TEXT,
                        address TEXT,
                        mobile TEXT,
                        status INTEGER,
                        unique_id TEXT
                      )''')

        # Insert data into table
          cursor.execute("INSERT INTO employees (name, address, mobile, status,
unique_id) VALUES (?, ?, ?, ?, ?)",
                    (name, address, mobile, 1, self.unique_id))
        conn.commit()

        # Close database connection
        conn.close()
```

```
    def centerOnScreen(self):
        qr = self.frameGeometry()
        cp = QDesktopWidget().availableGeometry().center()
        qr.moveCenter(cp)
        self.move(qr.topLeft())

if __name__ == '__main__':
    app = QApplication(sys.argv)
    add_employee_widget = AddEmployeeWidget()
    add_employee_widget.show()
    sys.exit(app.exec_())
```

7. Check Employee Attention Panel

```
import sqlite3
import sys

from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QTableWidget, QTableWidgetItem, QPushButton

class EmployeeAttention(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Employee Attention')
        self.setMinimumWidth(600)
        self.setMinimumHeight(400)
        self.initUI()

    def initUI(self):
        layout = QVBoxLayout()

        self.table = QTableWidget()
        self.table.setColumnCount(4)
        self.table.setHorizontalHeaderLabels(['Unique ID', '  Name  ', 'Entry Time', 'Exit Time'])
        self.table.setStyleSheet(
            """
            QTableWidget {
                background-color: #333333;
                border: 2px solid #666666;
                border-radius: 5px;
                padding:10px;
                color: #ffffff;
```

```
            }
            QTableWidget::item {
                padding: 10px;
                border-bottom: 1px solid #666666;
            }

            QTableWidget::item:selected {
                background-color: #555555;
                color: #ffffff;
            }

            QHeaderView::section {
                margin: 5px;
                background-color: #121212;
                border: none;
                padding: 5px 15px;
                border-radius:5px;
                font-weight: bold;
                color: #fff;
            }
            """
        )

        self.table.verticalHeader().setVisible(False)

        layout.addWidget(self.table)

        self.loadEmployeeDetails()

        self.refresh_button = QPushButton('Refresh')
        self.refresh_button.clicked.connect(self.loadEmployeeDetails)

        self.refresh_button.setStyleSheet(
            "font-size: 13px; color: white; background-color: blue; padding: 15px
10px; border: none; border-radius: 5px;")

        layout.addWidget(self.refresh_button)

        self.setLayout(layout)

    def loadEmployeeDetails(self):
        try:
            conn = sqlite3.connect('database.db')
            cursor = conn.cursor()
```

```
            cursor.execute("SELECT unique_id, name, entry_time, exit_time FROM employee_logs")
        employees = cursor.fetchall()

        self.table.setRowCount(0)

        for row_num, employee in enumerate(employees):
            self.table.insertRow(row_num)
            for col_num, data in enumerate(employee):
                item = QTableWidgetItem(str(data))
                item.setTextAlignment(0x84)
                self.table.setItem(row_num, col_num, item)

        conn.close()
    except Exception as e:
        print(f"Error: {e}")


if __name__ == '__main__':
    app = QApplication(sys.argv)
    employee_attention_widget = EmployeeAttention()
    employee_attention_widget.show()
    sys.exit(app.exec_())
```

8. Employee Detail Panel

```
import sqlite3
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QTableWidget, QTableWidgetItem, QPushButton

class EmployeeDetailsWidget(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Employee Details')
        self.setMinimumWidth(600)
        self.setMinimumHeight(400)
        self.initUI()

    def initUI(self):
        layout = QVBoxLayout()

        # Table to display employee details
```

```
        self.table = QTableWidget()
        self.table.setColumnCount(5)
         self.table.setHorizontalHeaderLabels(['ID', 'Name', 'Address', 'Mobile No',
'Unique ID','Unique ID'])
        self.table.setStyleSheet(
            """
            QTableWidget {
                background-color: #333333;
                border: 2px solid #666666;
                border-radius: 5px;
                padding:10px;
                color: #ffffff;
            }
            QTableWidget::item {
                padding: 10px;
                border-bottom: 1px solid #666666;
            }

            QTableWidget::item:selected {
                background-color: #555555;
                color: #ffffff;
            }

            QHeaderView::section {
                margin: 5px;
                background-color: #121212;
                border: none;
                padding: 5px 15px;
                border-radius:5px;
                font-weight: bold;
                color: #fff;
            }
            """
        )

        self.table.verticalHeader().setVisible(False)

        layout.addWidget(self.table)

        self.loadEmployeeDetails()

        # Refresh button
        self.refresh_button = QPushButton('Refresh')
        self.refresh_button.clicked.connect(self.loadEmployeeDetails)
```

```
        self.refresh_button.setStyleSheet(
            "font-size: 13px; color: white; background-color: blue; padding: 15px
10px; border: none; border-radius: 5px;")

        layout.addWidget(self.refresh_button)

        self.setLayout(layout)

    def loadEmployeeDetails(self):
        try:
            conn = sqlite3.connect('database.db')
            cursor = conn.cursor()

            cursor.execute("SELECT  id, name, address, mobile, unique_id FROM
employees")
            employees = cursor.fetchall()

            self.table.setRowCount(0)

            for row_num, employee in enumerate(employees):
                self.table.insertRow(row_num)
                for col_num, data in enumerate(employee):
                    item = QTableWidgetItem(str(data))
                    item.setTextAlignment(0x84)  # Align center
                    self.table.setItem(row_num, col_num, item)


            conn.close()
        except Exception as e:
            print(f"Error: {e}")

if __name__ == '__main__':
    app = QApplication(sys.argv)
    employee_details_widget = EmployeeDetailsWidget()
    employee_details_widget.show()
    sys.exit(app.exec_())
```

Output:

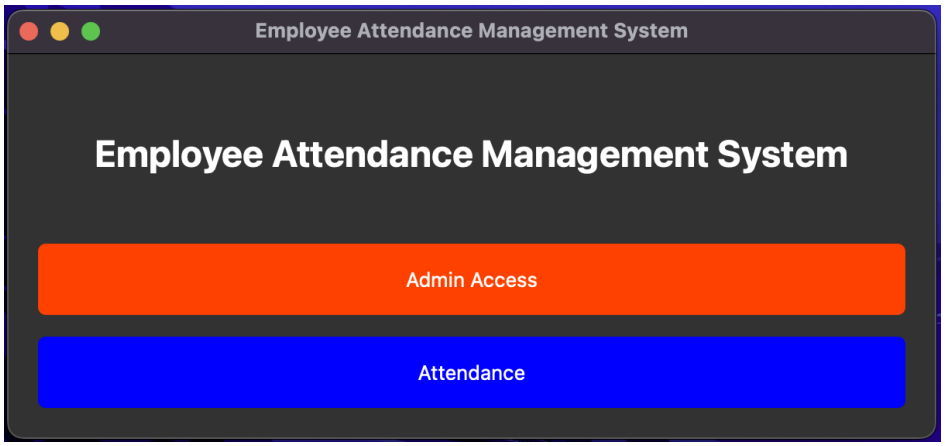

Figure 1 : Employee Attendance System Control Panel

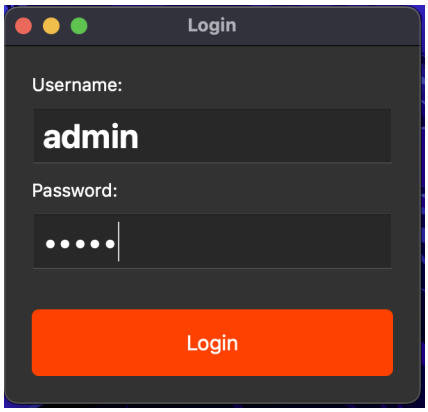

Figure 2 :Admin Control Panel

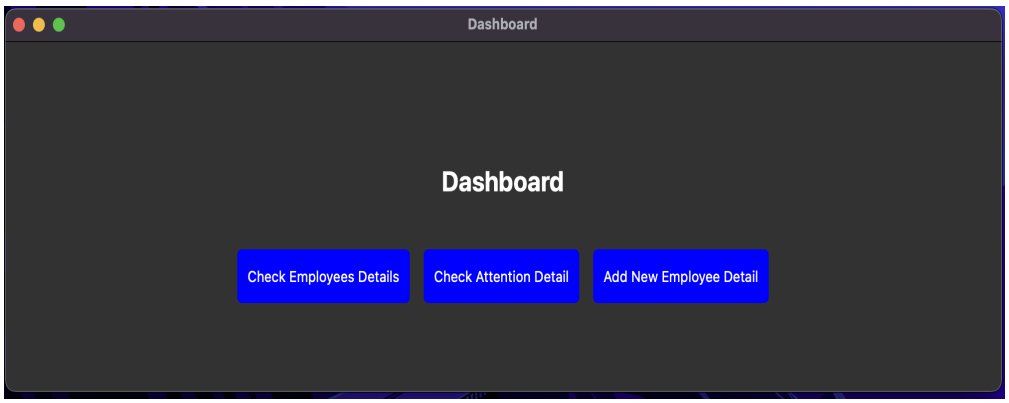

Figure 3 : Admin Dashboard Panel

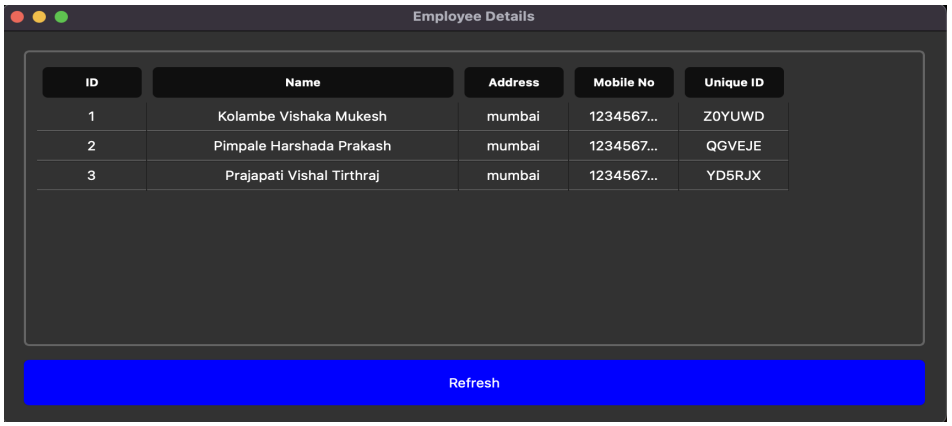

Figure 4 : Employee Details Panel

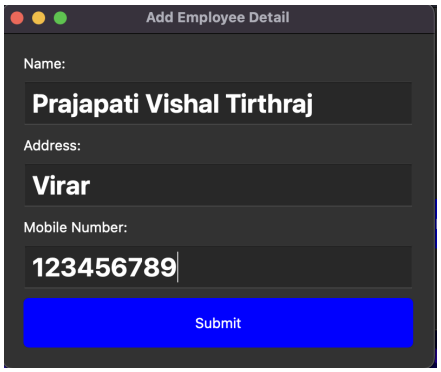

Figure 5 : New Employee Adding Panel

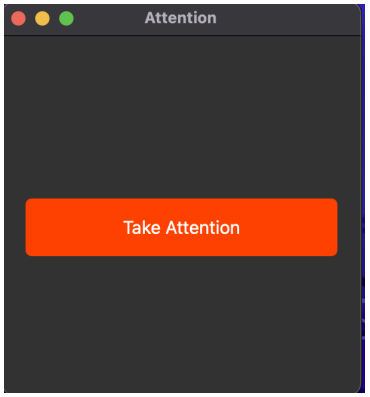

Figure 7 : To Start Taking Attendance Panel

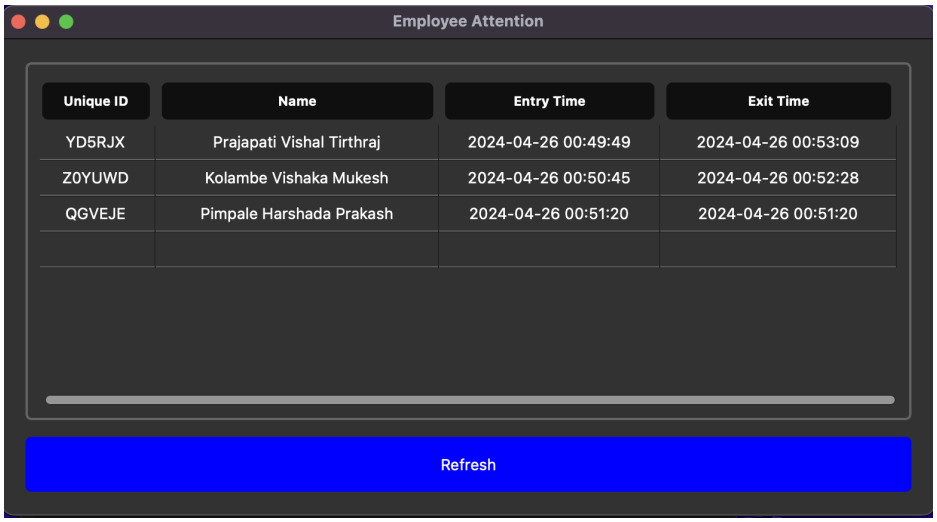

Figure 6 : Employee Attendance Detail Panel

Figure 8 : Opening Camera to Take Attendance Panel

Conclusion: In conclusion, the implementation of an Employee Attention System with QR code integration represents a significant opportunity for our organization to prioritize employee engagement, satisfaction, and productivity. By centralizing communication, feedback, training, and support resources within a user-friendly platform, we can create a more cohesive and supportive work environment where employees feel valued and empowered.