```
#AUTOMATED SYSTEM UPDATE TOOL POC

import schedule
import time
import threading
from datetime import datetime

# In-memory storage for update records
update_records = {}
update_counter = 1  # To generate unique update IDs

# CRUD Operations
def create_update_record(description, scheduled_time):
    global update_counter
    update_id = update_counter
    update_records[update_id] = {
        'description': description,
        'scheduled_time': scheduled_time,
        'status': 'Scheduled'
    }
    print(f"[CREATE] Update Record {update_id} created.")
    update_counter += 1
    return update_id

def read_update_record(update_id):
```

```
    record = update_records.get(update_id)
    if record:
        print(f"[READ] Update Record {update_id}: {record}")
        return record
    else:
        print(f"[READ] Update Record {update_id} not found.")
        return None

def update_update_record(update_id, description=None, scheduled_time=None):
    record = update_records.get(update_id)
    if record:
        if description:
            record['description'] = description
        if scheduled_time:
            record['scheduled_time'] = scheduled_time
        print(f"[UPDATE] Update Record {update_id} updated.")
        return True
    else:
        print(f"[UPDATE] Update Record {update_id} not found.")
        return False

def delete_update_record(update_id):
    if update_id in update_records:
        del update_records[update_id]
        print(f"[DELETE] Update Record {update_id} deleted.")
```

```
        return True
    else:
        print(f"[DELETE] Update Record {update_id} not found.")
        return False

# Scheduling System Update
def schedule_system_update(update_id):
    record = update_records.get(update_id)
    if not record:
        print(f"[SCHEDULE] Update Record {update_id} does not exist.")
        return

    scheduled_time = record['scheduled_time']
    try:
        run_time = datetime.strptime(scheduled_time, "%Y-%m-%d %H:%M:%S")
        delay = (run_time - datetime.now()).total_seconds()
        if delay < 0:
            print(f"[SCHEDULE] Scheduled time for Update {update_id} is in the past.")
            return
    except ValueError:
        print(f"[SCHEDULE] Invalid datetime format for Update {update_id}. Use YYYY-MM-DD
HH:MM:SS")
        return

    print(f"[SCHEDULE] Update {update_id} scheduled to run at {scheduled_time}.")
```

```
    def job():
        perform_update(update_id)

    schedule_time = run_time.strftime("%Y-%m-%d %H:%M:%S")
    schedule.every().day.at(run_time.strftime("%H:%M:%S")).do(job)

def perform_update(update_id):
    record = update_records.get(update_id)
    if not record:
        print(f"[PERFORM] Update Record {update_id} does not exist.")
        return
    print(f"[PERFORM] Performing Update {update_id}: {record['description']}")
    # Simulate update process
    time.sleep(2)  # Simulate time taken to perform update
    record['status'] = 'Completed'
    print(f"[PERFORM] Update {update_id} completed.")

# Tracking Update Status
def track_update_status(update_id):
    record = update_records.get(update_id)
    if record:
        print(f"[TRACK] Update {update_id} Status: {record['status']}")
        return record['status']
    else:
        print(f"[TRACK] Update Record {update_id} not found.")
```

```
        return None

# Background thread to run the scheduler
def run_scheduler():
    while True:
        schedule.run_pending()
        time.sleep(1)

# Example Usage
if __name__ == "__main__":
    # Start the scheduler in a separate thread
    scheduler_thread = threading.Thread(target=run_scheduler, daemon=True)
    scheduler_thread.start()

    # Create some update records
    update1 = create_update_record("Security patch for module X", "2024-09-28 14:30:00")
    update2 = create_update_record("Feature update for application Y", "2024-09-28 15:00:00")

    # Read update records
    read_update_record(update1)
    read_update_record(update2)

    # Update an update record
    update_update_record(update1, description="Critical security patch for module X")
```

```
# Schedule updates
schedule_system_update(update1)
schedule_system_update(update2)

# Track update status
time.sleep(1)  # Wait a moment for scheduling to take effect
track_update_status(update1)
track_update_status(update2)

# Keep the main thread alive to allow scheduled jobs to run
try:
    while True:
        time.sleep(10)
except KeyboardInterrupt:
    print("Scheduler stopped.")
```