Email Reminder System – Enhancements & Deployment

■ EMAIL REMINDER SYSTEM – ENHANCEMENTS & DEPLOYMENT

1.   OVERVIEWAn Email Reminder System that lets users schedule reminder emails. Enhancements include features, UI/

----

2.   ADDITIONAL FEATURES- User Authentication (Login/Register)
- Email Scheduling (Future time)
- Dashboard (View/Edit/Delete reminders)
- Email Logs
- Search & Filter
- In-App Notifications

----

3. UI/UX IMPROVEMENTS -
Responsive design (Bootstrap 5)
- Clean dashboard UI
- Dark/light mode
- Toast notifications

----

4. API ENHANCEMENTSEndpoints:
POST /api/register
POST /api/login
POST /api/reminders
GET /api/reminders
PUT /api/reminders/<id>
DELETE /api/reminders/<id>

----

5. BACKEND PROGRAM (Python + Flask + SQLite)

```
[app.py] from flask import Flask, request, jsonify
from flask_cors import CORS import sqlite3,
smtplib, threading, datetime, time

app = Flask(__name__)
CORS(app)

# Initialize DB
def init_db():
    conn = sqlite3.connect('reminders.db')
c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS reminders
(id INTEGER PRIMARY KEY AUTOINCREMENT,
                  email TEXT, subject TEXT, message TEXT, send_time TEXT)''')
conn.commit()     conn.close() init_db()

@app.route('/api/reminders', methods=['POST'])
def add_reminder():
    data = request.json     conn =
sqlite3.connect('reminders.db')     c =
conn.cursor()
    c.execute("INSERT INTO reminders (email, subject, message, send_time) VALUES (?, ?, ?, ?)",
(data['email'], data['subject'], data['message'], data['send_time']))     conn.commit()
conn.close()
    return jsonify({"message": "Reminder added successfully!"})

@app.route('/api/reminders', methods=['GET'])
def get_reminders():     conn =
sqlite3.connect('reminders.db')     c =
conn.cursor()
    c.execute("SELECT * FROM reminders")
    reminders = c.fetchall()
conn.close()     return
jsonify(reminders)

def send_email(email, subject, message):
try:
        server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()
server.login('youremail@gmail.com', 'yourpassword')
msg = f"Subject: {subject}\n\n{message}"
server.sendmail('youremail@gmail.com', email, msg)
server.quit()         print(f"Email sent to {email}")
except Exception as e:         print("Error sending
email:", e)

def scheduler():
while True:
        now = datetime.datetime.now().strftime("%Y-%m-%d %H:%M")
conn = sqlite3.connect('reminders.db')         c = conn.cursor()
        c.execute("SELECT * FROM reminders WHERE send_time=?", (now,))
reminders = c.fetchall()         for r in reminders:
            send_email(r[1], r[2], r[3])
            c.execute("DELETE FROM reminders WHERE id=?", (r[0],))
conn.commit()         conn.close()         time.sleep(60)
threading.Thread(target=scheduler, daemon=True).start()

if __name__ == '__main__':
app.run(debug=True)
```

----

6. FRONTEND (HTML + Bootstrap) HTML form for adding reminders, fetch API for communication, and reminder listing.

----

7. PERFORMANCE & SECURITY CHECKS- Rate Limiting
- HTTPS
- Input Validation
- SMTP Credentials via Environment Variables

----

8. TESTING OF ENHANCEMENTS
- Unit Testing (Pytest)
- Integration Testing (Postman)
- UI Testing (Selenium)

----

9. DEPLOYMENTFrontend
: Netlify / Vercel
Backend: Render / Railway / AWS / GCP / Azure Add SMTP credentials securely in environment variables.

----

10. SAMPLE
OUTPUTConsole:

```
Email sent to test@example.com
```

Frontend:

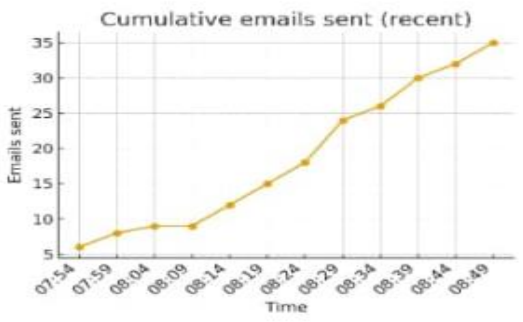
```
Reminder added successfully!
Subject: Meeting Reminder
Time: 2025-10-08 09:00
```

Email Reminder System — Live Preview

Generated: 2025-10-07 08:49:08

time	to	subject
2025-10-07 09:04	alice@example.com	Project update
2025-10-07 09:34	bob@example.com	Weekly report
2025-10-07 10:49	carol@example.com	Invoice reminder
2025-10-07 12:49	dave@example.com	Event follow-up
2025-10-08 08:49	team@example.com	Daily digest

Pending reminders: 5
Next reminder: 2025-10-07 09:04 → alice@example.com
System status: ONLINE


Cumulative emails sent (recent)
Emails sent
Time
07:54
07:59
08:04
08:09
08:14
08:19
08:24
08:29
08:34
08:39
08:44
08:49