Industry Standard Directive: SMS Scheduling System Development

Objective:
Develop a secure and efficient SMS scheduling system for Samparka with a clear distinction between customer and admin functionalities. The system must adhere to best practices in software development, ensuring maintainability, scalability, and security.

Project Guidelines

General Standards

1. Code Quality:
   - Follow industry-standard coding practices, including proper commenting, modular structure, and adherence to the SOLID principles.
   - Ensure code is readable, maintainable, and optimized for performance.

2. Security:
   - Use encrypted storage for sensitive data such as passwords.
   - Implement input validation to prevent SQL injection and other vulnerabilities.

3. Collaboration Protocol:
   - No repository sharing outside authorized personnel.
   - Maintain version control using Git (e.g., GitHub/Bitbucket/Private Repo).
   - Document all changes with clear commit messages.

4. Documentation:
   - Provide clear documentation for both customer and admin functionalities.
   - Include a README file outlining system requirements, setup instructions, and deployment steps.

5. Testing:
   - Write test cases for critical functions and perform rigorous testing to ensure system stability and reliability.

System Requirements

Customer Portal

- Login Functionality:
  - Customers log in using credentials issued by the admin.
  - Implement a robust authentication mechanism.
  - On failed login, display an appropriate message: “Credentials do not match.”

- Navbar:
  - Include Samparka’s logo and a link to the customer’s profile.

- Message Scheduling Form:
  - Fields:
    - Message Box: Restrict input to 30 characters max.
    - Date and Time Selector: Allow scheduling messages at a specific time.
  - On submission:
    - Display a confirmation message (e.g., "Message submitted successfully").
    - Log the message in the Message History.

- Message History:
  - Display all scheduled messages along with their statuses: Submitted, Processing, Confirmed, Sent.
  - Each message must include the timestamp and scheduled time.

- SMS Charges:
  - Display the current SMS charge (default: NPR 1.7).
  - Dynamically update the displayed charge based on admin-side changes.

Admin Portal
- Customer Management:
  - Provide a dashboard to view all customers along with their email and encrypted passwords.

- Message Management:
  - Receive an email alert for every new message scheduled by a customer.
  - View, edit, and update message statuses: Submitted, Processing, Confirmed, Sent.
  - Include sorting and filtering options for better usability.

- SMS Charge Management:
  - Display the current SMS charge and allow the admin to update the rate.
  - Changes should be immediately reflected on the customer portal.

Technical Requirements

Frontend:
- Use a modern framework such as React.js or Vue.js for better user experience.
- Ensure a responsive design compatible with desktops and mobile devices.

Backend:
- Develop the backend using Node.js, Django, or similar frameworks.
- Use RESTful APIs for communication between the frontend and backend.

Database:
- Use a secure and scalable database like PostgreSQL or MongoDB.
- Encrypt sensitive data such as passwords.

Email Integration:
- Implement email notifications using a reliable service (e.g., SendGrid, SMTP).

Deployment:
- Host the application on a scalable platform such as AWS, Azure, or Heroku.

Development Tools

1. Frontend Framework: React.js or Vue.js
2. Backend Framework: Node.js or Django
3. Database: PostgreSQL or MongoDB
4. Email Integration: SendGrid or SMTP
5. Version Control: Git (GitHub/Bitbucket)
6. Deployment: AWS, Azure, or Heroku

Project Deadline: 2 Weeks