1. Modular and Generic Web Crawler Design:

Key Components:

- **Core Crawler Engine:** Core module in OOP responsible for fetching web pages, extracting data, and storing it.
- **Site-specific Modules:** Modules containing site-specific parsing logic. These modules can be easily added or updated without modifying the core crawler engine.
- **Database Interface:** Handles storing crawled data efficiently.
- **Scheduler:** Controls the frequency of crawling for each site.

2. Infrastructure Design:

To optimize costs and monitoring capabilities, we'll leverage cloud services, specifically Google Cloud Platform (GCP).

Infrastructure Components:

- **Compute Engine:** Utilize GCE instances for running the crawler. Instead of deploying one instance per site, we'll design the crawler to run multiple tasks concurrently on each instance.
- **Kubernetes Engine (GKE):** Containerize the crawler components and deploy them as microservices on GKE clusters. Kubernetes will manage scaling and orchestration, improving resource utilization.
- **Cloud Scheduler:** Configure Cloud Scheduler to trigger crawling tasks based on the predefined schedule.
- **Cloud Logging and Monitoring:** Utilize Cloud Logging for centralized log management and Cloud Monitoring for monitoring instance health and performance metrics.

3. Implementation Guidelines:

- **Containerization:** Containerize each component of the crawler using Docker for portability and consistency.
- **Configuration Management:** Use configuration files or environment variables to store site-specific configurations, making it easier to add or update sites without code changes.

- **Version Control:** Implement version control for the crawler codebase to track changes and collaborate effectively.
- **Error Handling and Retry Mechanism:** Implement robust error handling and retry mechanisms to handle transient failures and ensure data integrity.
- **Logging and Alerting:** Integrate logging libraries to capture relevant information and set up alerts for critical events or errors.
- **Security Measures:** Implement security best practices, such as access controls, encryption, and regular vulnerability assessments, to protect the crawler and crawled data.

4. Maintenance and Monitoring:

- **Automated Testing:** Implement automated tests to verify the functionality of the crawler and detect regressions early.
- **Continuous Integration/Continuous Deployment (CI/CD):** Set up CI/CD pipelines to automate the deployment process and streamline code updates.
- **Health Checks:** Implement health checks for each component to detect failures and automatically restart or replace instances if necessary.
- **Dashboard and Alerts:** Create dashboards in Cloud Monitoring to visualize instance metrics and set up alerts for abnormal behavior or performance degradation.

Conclusion:

By following these design principles and implementation guidelines, we can build a scalable, cost-effective, and maintainable web crawling infrastructure capable of handling the specified requirements. This solution will reduce manual effort, improve reliability, and provide better visibility into the crawling process.