Sample Document - New Feature Proposal with Multiple Violations

NEW FEATURE PROPOSAL: Customer Analytics Dashboard

Author: Engineering Team

Date: November 15, 2024

Status: Draft - Pending Compliance Review

==========

OVERVIEW:

==========

We propose building a real-time customer analytics dashboard that displays
user behavior metrics, purchase patterns, and demographic insights to help
the marketing team make data-driven decisions.

==========

TECHNICAL IMPLEMENTATION:

==========

1. DATABASE SCHEMA:

   We will create a new analytics database to store:

   - Customer email addresses

   - Full names

   - Purchase history with amounts

   - Browsing patterns and page views

   - IP addresses for geo-location

   Database: MySQL 5.7

   Query implementation example:

```python
   def get_customer_data(email):
       # Direct string concatenation for query building
       query = "SELECT * FROM customers WHERE email = '" + email + "'"
       cursor.execute(query)
       return cursor.fetchall()
```

```

2. DATA STORAGE:

   - Customer data will be stored in plaintext JSON files for quick access

   - Files will be stored in: /var/data/analytics/customers/

   - Weekly backup to S3 bucket: s3://acme-analytics-backup/

   - No encryption needed since files are on internal network

3. AUTHENTICATION & ACCESS:

   - Dashboard accessible via simple username/password authentication

   - No MFA required (to keep it simple for marketing team)

   - Session tokens expire after 24 hours

   - API credentials for database connection:

     * Host: analytics-db.internal.acme.com

     * Username: analytics_user

     * Password: Analyticspassword in source code for convenience (see config.py line 47)

     * API Key: sk_live_abcd1234efgh5678 (hardcoded in analytics_client.py)

4. LOGGING & DEBUGGING:

   All user activities will be logged for debugging purposes:

```python

   logger.info(f"User {user_email} accessed dashboard at {timestamp}")

   logger.debug(f"Query executed: {full_query} for user {customer_email}")

   logger.error(f"Failed to load data for {customer_email}, IP: {ip_address}")

```

   Logs are stored indefinitely for historical analysis.

5. ERROR HANDLING:

   When errors occur, we return detailed messages to help debugging:

```python

   except Exception as e:

       return {

```
            "error": str(e),
            "query": query_string,
            "customer_email": email,
            "stack_trace": traceback.format_exc()
        }
```

DATA RETENTION:

- User analytics data retained indefinitely for historical trend analysis
- No automated deletion process
- Manual deletion available upon request (processed within 90 days)
- Backups kept for 10 years

TIMELINE & RESOURCES:

- Development: 3 months
- Testing: 2 weeks
- Budget: $50,000
- Team: 2 backend engineers, 1 frontend engineer

END OF PROPOSAL
