1. Basic Data Validation Module

Functionality: This module ensures that all incoming data, such as user inputs for emails, phone numbers, and dates, adheres to specified formats and standards before any processing or storage occurs. It also sanitizes inputs to prevent SQL injection attacks.

How It Works:

- **Validation:** The module uses regex patterns to match input data against predefined formats for emails (e.g., **user@example.com**), phone numbers (allowing characters like **+**, **-**, **(**, **)**, and spaces), and dates (e.g., **YYYY-MM-DD**).
- **Sanitization:** It removes or escapes characters that could be used in SQL injection attacks, ensuring inputs are safe before being used in database queries.

Technologies and Methodologies:

- Utilize regex for pattern matching.
- Implement common sanitization libraries or functions to escape hazardous characters.
- Could be developed as middleware in web frameworks to automatically process data from HTTP requests.

2. Data Compression Module

Functionality: This module compresses string data using Huffman coding, a common data compression algorithm that reduces the size of text data by using variable-length codes for characters, with shorter codes for more frequent characters.

How It Works:

- **Compression:** Analyze the frequency of characters in the input text and build a Huffman tree based on these frequencies. Each character is then replaced by its corresponding Huffman code in the compressed data.
- **Decompression:** Use the Huffman tree to decode the compressed data back into its original form.
- The module reads the text to be compressed from a file, performs compression or decompression, and then outputs the result.

Technologies and Methodologies:

- Implementation of Huffman coding algorithms.
- File I/O operations for reading input text and writing compressed or decompressed text.

3. Data Encryption Module

Functionality: Implements a simple configurable substitution cipher for strings, allowing for the encryption and decryption of text based on a user-defined configuration. This module can be used to secure sensitive information.

How It Works:

- **Encryption:** Replace each character in the input string with another character according to a substitution table provided in the configuration.
- **Decryption:** Reverse the process by replacing each character in the encrypted string back to its original character using the same substitution table.

- The substitution table and other configurations can be defined in an external file, making the cipher easily adjustable.

Technologies and Methodologies:

- Use of maps or dictionaries to store substitution pairs.
- Configuration management to load and parse the substitution table from an external file.

4. Configuration Management Module

Functionality: Loads application settings from an external JSON or XML file. This module allows the application to be flexible and configurable, supporting various data types and special characters in the configuration files.

How It Works:

- **Loading Configuration:** Parse the JSON or XML configuration file to read settings, which could include database connection details, application preferences, or module-specific settings like the substitution table for the Data Encryption Module.
- **Handling Different Data Types:** The module interprets and correctly handles different data types found in the configuration file, such as numbers, strings, lists, and boolean values.
- **Special Characters:** Ensure that the parser can handle special characters in strings, and possibly encode or escape them if necessary for security or functionality reasons.

Technologies and Methodologies:

- JSON or XML parsing libraries to read and interpret the configuration files.
- Data type handling mechanisms to ensure that settings are correctly utilized within the application.