Assignment - 01

Design and Development of a Modular, Extensible Archive File Management System

1. Research and Analysis

- **Objective:** Understand the structure and nuances of tar and zip archive formats.
- **Activities:**
  - Study the specifications of tar and zip formats.
  - Investigate existing libraries and tools like libtar, zlib, and libzip that might be utilized to handle these formats efficiently.

2. System Design

- **Modular Architecture:**
  - Design the core application logic to be independent of the archive format handlers.
  - Plan for dynamic loading of format-specific modules, allowing for future expansion.
- **Plugin System:**
  - Define a common interface for archive handlers that includes methods for listing contents, extracting files, and adding files to the archive.
  - Implement a system to dynamically load these plugins at runtime based on the file extension or user selection.

3. Implementation Details

- **Core Application:**
  - Develop a command-line interface (CLI) that accepts file paths and plugin selection as inputs.
  - Implement a dynamic loading system to instantiate the correct handler based on the file extension or user input.
- **Archive Handler Interface:**
  - Specify the functions and properties that each archive handler must implement, ensuring consistency and ease of integration.
- **Tar and Zip Handler Plugins:**
  - Develop these as shared libraries (.dll, .so, .dylib) depending on the platform.
  - Ensure that these handlers can list the contents of their respective archive formats effectively.

4. Development Process

- **Design Phase:**
  - Finalize the plugin interface and core application architecture.
  - Design the CLI, focusing on user experience and ease of use.
- **Implementation Phase:**
  - Code the core application framework and the handler interface.
  - Develop the tar and zip handlers according to the specified interface.

- **Testing Phase:**
  - Write unit tests for each component to ensure reliability and correct functionality.
  - Conduct integration tests to verify that the application works as intended when all components are integrated.

5. Error Handling and Robustness

- **Input Validation:**
  - Implement thorough input validation to mitigate risks such as security vulnerabilities.
- **Error Handling:**
  - Ensure comprehensive error handling within both the core application and the plugins to maintain stability.
- **Resource Management:**
  - Implement efficient resource management to handle memory and file operations cleanly and effectively.

6. Performance and Security

- **Efficiency and Scalability:**
  - Optimize the process of reading and parsing archives, ensuring the application can handle large archives smoothly.
- **Security:**
  - Prioritize the secure parsing of archives, safeguarding against common exploits like buffer overflows.
- **Dependency Management:**
  - Manage dependencies carefully, especially when incorporating third-party libraries, to maintain security and stability.

7. User Documentation and Support

- **Documentation:**
  - Provide comprehensive documentation for both users and developers, detailing the application usage and plugin development process.
- **Help System:**
  - Integrate a help system within the application, offering guidance on usage and available commands.

8. Deliverables

- Finalize and deliver the source code for the core application and handlers, complete with unit and integration test suites.
- Include user and developer documentation to assist with usage and future development.
- Provide a compiled version of the application, ready for immediate use.

This structured approach ensures the development of a robust, efficient, and extensible archive file handler that meets the project's requirements while laying a solid foundation for future enhancements and format support.