No. Of Functional Features Included In The Solution

- User Registration and Authentication

- User accounts and login.
- Website Design and Editing

- Graphic design and customization.
- Website creation tools.
- Content Management

- Adding, editing, and organizing content.
- Website Preview and Testing

- Real-time website preview.
- Cross-device compatibility.
- Publishing and Hosting

- Website hosting and domain management.
- User Interaction

- Contact forms and comments.
- Analytics and Reporting

- Website traffic and user behavior tracking.
- Security and Privacy

- Data encryption and user data protection.
- Third-Party Integrations

- Integration with external services.
- Feedback and Support

- User feedback collection and support.

Code-Layout, Readability, and Reusability:

Code Layout: A well-structured code layout promotes consistency and makes the codebase more readable. It involves using indentation, consistent spacing, and following coding style conventions (e.g., PEP 8 for Python).

Readability: Readable code is easy to understand and maintain. It should use descriptive variable and function names and have clear and concise comments explaining complex sections or algorithms.

Reusability: Reusable code components are designed in a modular fashion, making it easy to use them in different parts of the application. Libraries, modules, and functions should be encapsulated in a way that allows them to be used elsewhere, reducing code duplication and improving maintainability.

Utilization of Algorithms, Dynamic Programming, Optimal Memory Utilization:

Algorithms: The choice of algorithms depends on the specific tasks the software must perform. It includes sorting algorithms, searching algorithms, data compression algorithms, and more. Selecting the right algorithm is critical for optimizing the performance of the software.

Dynamic Programming: Dynamic programming is a technique for solving problems by breaking them down into smaller subproblems and reusing the solutions to those subproblems. It's particularly useful for optimizing solutions in scenarios where overlapping subproblems exist.

Optimal Memory Utilization: Efficient memory usage is essential for performance optimization and preventing memory leaks. It involves minimizing memory consumption and deallocating memory when it's no longer needed. Proper memory management is especially important in lower-level languages like C and C++.

Debugging & Traceability:

Debugging: Debugging is the process of identifying and fixing defects, errors, or unexpected behaviors in the code. Debugging tools, breakpoints, and logging are essential for tracking down issues and ensuring the code works as intended.

Traceability: Traceability helps in understanding the history of code changes. It often involves version control systems like Git, which allow you to track changes to the codebase over time. Traceability is crucial for identifying the source of issues, tracking enhancements, and managing collaborative development.

Exception Handling:

Exception Handling: Exception handling is a way to deal with runtime errors and exceptional situations in a graceful manner. This involves using try-catch blocks (in languages like Java and C#) or error handling mechanisms (in languages like Python and JavaScript) to handle errors and prevent application crashes.

Logging: Exception handling often includes logging the details of errors or exceptions, which helps in diagnosing issues, especially in production environments.

User-Friendly Error Messages: When an error occurs, presenting clear and user-friendly error messages is essential. It assists users in understanding what went wrong and how to proceed.