**Software Requirements Specification (SRS) for Web Application Project:**

1. **Functional Requirements:**
   * The web application shall allow users to view information about house management services.
   * Users shall be able to navigate through different sections of the website, including "About Us," "Services," "Careers," and "Comment."
   * The application shall provide a platform for users to submit comments and feedback.
   * An administrative interface shall be available for managing user comments, including editing and deleting comments as necessary.
   * Users shall be able to download forms and upload resumes for career opportunities.
2. **Non-Functional Requirements:**
   * **Performance:** The application shall load within a reasonable time frame, with minimal latency.
   * **Reliability:** The application shall be robust and resilient, with error handling mechanisms in place to handle unexpected scenarios gracefully.
   * **Security:** User data shall be securely stored and transmitted using encryption protocols. Access to administrative features shall be restricted to authorized personnel only.
   * **Usability:** The user interface shall be intuitive and user-friendly, with clear navigation and responsive design to support various devices and screen sizes.
   * **Scalability:** The application architecture shall be scalable to accommodate potential growth in user traffic and data volume.
   * **Compatibility:** The application shall be compatible with modern web browsers and adhere to web standards and accessibility guidelines.
3. **Features, Interfaces, and Constraints:**
   * **Features:**
     + Comprehensive house management information, including services offered and pricing.
     + Ability to submit comments and feedback.
     + Administrative interface for managing user comments.
     + Career opportunities section with the option to download forms and upload resumes.
   * **Interfaces:**
     + User interface for accessing and navigating the website.
     + Administrative interface for managing comments.
     + Integration with external libraries or frameworks for frontend and backend development.
   * **Constraints:**
     + Compatibility with existing hardware and software infrastructure.
     + Adherence to budget and time constraints for development and deployment.
     + Compliance with relevant regulations and standards, such as data protection regulations and web accessibility guidelines.
4. **External Interfaces and Dependencies:**
   * The web application may utilize external APIs or services for features such as user authentication (e.g., OAuth) and file storage (e.g., cloud storage providers).
   * Dependencies may include third-party libraries, frameworks, or tools used for frontend and backend development, such as Bootstrap for frontend styling and ASP.NET Core for backend development.

This Software Requirements Specification outlines the functional and non-functional requirements, features, interfaces, and dependencies of the web application project. It serves as a guide for development, ensuring that the application meets the needs and expectations of its users while adhering to quality standards and best practices.

**Software Design Description (SDD) for Web Application Project:**

1. **Overall Structure and Components:**
   * The web application follows a Model-View-Controller (MVC) architecture to separate concerns and facilitate modularity and scalability.
   * Components include the Model layer, responsible for data management and business logic; the View layer, responsible for rendering user interfaces; and the Controller layer, responsible for handling user inputs and orchestrating interactions between the Model and View.
2. **Detailed Design Specifications:**
   * **Model Layer:**
     + The Model layer consists of entities and data access components.
     + Data access components interact with the database using Entity Framework Core, providing CRUD (Create, Read, Update, Delete) operations for entities.
   * **View Layer:**
     + The View layer comprises Razor views and HTML templates for rendering dynamic web pages.
     + Views are structured using the Razor syntax to embed C# code and generate dynamic content based on data from the Model layer.
     + HTML templates are styled using Bootstrap for responsive design and enhanced user experience.
   * **Controller Layer:**
     + Controllers handle HTTP requests from clients and invoke appropriate actions to process the requests.
     + Each controller corresponds to a specific area of functionality, such as managing comments or handling user authentication.
     + Controllers utilize dependency injection to access services and repositories needed to fulfill requests.
3. **Interactions:**
   * **User Interaction:**
     + Users interact with the application through web browsers, accessing different pages and functionalities via hyperlinks and form submissions.
     + User inputs, such as submitting comments or uploading resumes, trigger corresponding controller actions to process and respond to the requests.
   * **Component Interaction:**
     + Components within the application interact via well-defined interfaces and protocols.
     + Controllers invoke methods on service classes to perform business logic operations and retrieve data from repositories.
     + Views consume data provided by controllers and render dynamic content using HTML templates.
4. **Diagrams:**
   * **Class Diagram:** Provides an overview of the classes and their relationships within the application, including entities, controllers, services, and repositories.
   * **Sequence Diagram:** Illustrates the sequence of interactions between components for specific use cases, such as submitting a comment or authenticating a user.

**Software Maintenance Documentation for Web Application Project:**

1. **Maintenance and Update Process:**
   * The maintenance and update process for the web application follows structured procedures to ensure smooth operation and continuous improvement.
   * Version control procedures are implemented using a Git repository, allowing for efficient collaboration, version tracking, and rollback capabilities.
   * Release management involves planning and scheduling regular updates and releases to introduce new features, enhancements, and bug fixes.
   * Change management processes are established to assess, prioritize, and implement changes effectively while minimizing disruption to the application's functionality.
2. **Modifications and Enhancements:**
   * Modifications or enhancements to the application are documented systematically, including details such as the rationale for the change, implementation details, and impact analysis.
   * Each modification or enhancement is assigned a unique identifier and tracked through the development lifecycle, from planning and implementation to testing and deployment.
   * Documentation is updated accordingly to reflect changes in the application's functionality, architecture, or design.
3. **Bug and Issue Resolution:**
   * Bugs, issues, and user feedback are addressed promptly through a systematic approach.
   * Reported bugs and issues are logged in a centralized issue tracking system, providing visibility and accountability for resolution.
   * Each bug or issue is assigned to a responsible team member, who investigates, diagnoses, and resolves the problem following established procedures.
   * Test cases are created or updated to verify bug fixes, ensuring that resolved issues do not recur in future releases.
   * Regular communication with users and stakeholders facilitates the collection of feedback, which is used to prioritize enhancements and improvements for future updates.
4. **Addressing User Feedback:**
   * User feedback is valued and considered integral to the ongoing improvement of the application.
   * Feedback channels, such as contact forms or support tickets, are provided to encourage users to report issues, suggest enhancements, or provide general feedback.
   * Feedback is systematically reviewed, categorized, and prioritized based on its significance and impact on the application.
   * Users are kept informed of the status of their feedback, including any actions taken or planned resolutions.
   * Feedback is incorporated into the development roadmap, guiding the prioritization of future updates and features to meet user needs and expectations.