1. Functional and Non-Functional Requirements Specifications:

Functional Requirements:

**User Management:**

Users should be able to register and login using their email and password.

Password recovery functionality should allow users to reset their passwords securely.

**Support Requests Management:**

Users should be able to submit support requests through a form, providing detailed descriptions of their issues.

Support requests should be assigned a unique identifier and stored in the database.

The system should track the status of support requests (e.g., open, closed)

Job Scheduling:

Administrators or designated personnel should have the ability to manually schedule jobs based on customer preferences and technician availability.

The system should provide real-time updates on job statuses

**Knowledge Base:**

The knowledge base should contain a repository of articles addressing common issues and their solutions.

Users should be able to search and access knowledge base articles based on keywords or categories.

Articles should be regularly updated.

**Non-Functional Requirements:**

**Performance**:

The system should be capable of handling a large number of concurrent user sessions without significant performance degradation.

Database queries and page loads should execute quickly to ensure a smooth user experience.

Security:

User authentication and authorization mechanisms should be robust and secure, employing industry-standard encryption techniques.

Sensitive user data, such as passwords and personal information, should be encrypted both in transit and at rest.

Regular security audits should be conducted to identify and mitigate potential vulnerabilities.

**Usability**:

The user interface should be intuitive and easy to navigate, with clear prompts and instructions for users at every step.

Accessibility features, such as alternative layouts and keyboard shortcuts, should be implemented to accommodate users with disabilities.

The architecture should be designed to scale horizontally to accommodate an increasing number of users and data volume.

Load balancing mechanisms should be in place to distribute traffic evenly across multiple servers.

**Maintainability:**

The codebase should adhere to best practices and coding standards, with clear documentation and comments for future reference.

Modular design principles should be followed to facilitate easy maintenance and updates to individual components.

Version control systems should be used to track changes and manage code repositories effectively.

Alternative Layouts: Provide options for users to customize the layout according to their preferences, such as adjusting font size, color contrast, and interface elements placement.

Screen Reader Compatibility: Ensure compatibility with screen reader software by incorporating appropriate HTML markup and ARIA attributes for screen reader navigation.

Keyboard Navigation: Implement keyboard shortcuts and navigation controls to enable users to interact with the application without relying solely on mouse input.

High Contrast Mode: Offer a high contrast mode option for users with visual impairments to improve readability and usability.

Text Resizing: Enable users to resize text dynamically to accommodate different visual preferences and needs.

5. Details of Components Reused/Refactored:

Laravel Framework: Utilize the Laravel PHP framework for backend development, leveraging its robust features and ecosystem for rapid development and scalability.

Third-Party Libraries: Integrate third-party libraries for authentication and database management (e.g., Eloquent ORM for database interactions), ensuring efficient and secure handling of user data and system resources.