Additional Documentation/Information Needed for Planning Tests

1. **User Stories/Requirements Documentation**
    - **Purpose:** To provide a clear understanding of the intended functionalities and expected outcomes.
    - **Details:** Specific user scenarios, detailed descriptions of each feature, acceptance criteria, edge cases, and any assumptions or constraints.
2. **Mockups or Design Specifications**
    - **Purpose:** To understand the visual layout and user interface interactions.
    - **Details**: Fesign mockups of each screen, specifications for UI elements (buttons, forms, etc.), and any responsive design requirements.
3. **Test Data Requirements**
    - **Purpose:** To ensure that tests cover a wide range of inputs, including valid, invalid, and edge cases.
    - **Details:** Examples of valid and invalid data for each input field, boundary values, and any special data handling requirements (e.g., character encoding).
4. **API Specifications**
    - **Purpose:** To understand the behaviour of backend services and ensure correct integration.
    - **Details:** Detailed documentation of each API endpoint, including request methods, required and optional parameters, sample request and response payloads, error codes, and expected error handling.
5. **Security Requirements**
    - **Purpose:** To ensure that the application meets security standards and protects user data.
    - **Details:** Authentication and authorization mechanisms, data encryption requirements, session management, and any compliance standards (e.g., GDPR).
6. **Performance Requirements**
    - **Purpose:** To define acceptable performance levels and ensure the application can handle expected loads.
    - **Details:** Response time thresholds, load testing criteria, stress testing scenarios, and performance monitoring requirements.
7. **Error Handling and Logging**
    - Purpose: To ensure that errors are handled gracefully and logged for troubleshooting.

- **Details:** List of possible errors, their causes, corresponding error messages, and logging requirements (e.g., log levels, formats).

8. **Deployment and Environment Details**
   - **Purpose:** To ensure consistency across different environments (development, testing, production).
   - **Details:** Configuration details, environment-specific settings, and any deployment scripts or instructions.
9. **Accessibility Requirements**
   - **Purpose:** To ensure the application is usable by people with disabilities.
   - **Details:** Accessibility standards to follow (e.g., WCAG), keyboard navigation requirements, screen reader compatibility, and color contrast guidelines.
10. **Integration Points**
    - **Purpose:** To understand dependencies on other systems or services.
    - **Details:** Descriptions of any third-party services or internal systems the application interacts with, data exchange formats, and integration testing requirements.

Components of the App That Can Be Tested and Their Hierarchy of Importance

1. **Form Validation (High Priority)**
   - **Components:** Required fields, format validation, length constraints, special characters handling.
   - **Why Important:** Ensures data integrity and prevents incorrect data entry, which is critical for maintaining database consistency and functionality.
2. **API Functionality (High Priority)**
   - **Components:** Endpoints for adding, updating, retrieving, and deleting contacts, response formats, error codes.
   - **Why Important:** Ensures the backend logic is working correctly and the application can interact with the server as expected.
3. **Error Messages (Medium Priority)**
   - **Components:** Display of validation errors, system errors, and user-friendly messages.
   - **Why Important:** Enhances user experience by providing clear feedback and guidance, helping users correct mistakes and understand system issues.
4. **UI Components (Medium Priority)**

    - **Components:** Input fields, buttons, labels, modals, navigation bars.
    - **Why Important:** Ensures usability and accessibility, making the application easy to use and navigate.
5. **Navigation (Medium Priority)**
    - **Components:** Page transitions, navigation links, breadcrumbs, back/forward button functionality.
    - **Why Important:** Ensures a seamless user experience by allowing users to move through the application efficiently.
6. **Database Interactions (Low Priority)**
    - **Components:** Data persistence, retrieval, updates, deletions, consistency checks.
    - **Why Important:** Ensures data is stored and retrieved correctly, maintaining the integrity and reliability of the application data.
7. **Security Features (Medium Priority)**
    - **Components:** Authentication, authorization, data encryption, session management.
    - **Why Important:** Protects user data and ensures that only authorized users can access or modify data.
8. **Performance Aspects (Medium Priority)**
    - Components: Response times, load handling, stress resilience.
    - Why Important: Ensures the application performs well under expected usage conditions, providing a responsive user experience.

Approach to Testing Each Area

1. **Form Validation**
    - **Manual Testing:** Enter various combinations of valid and invalid data to ensure that form validations work correctly.
    - **Automated Testing:** Use automated testing tools (e.g., Cypress) to simulate user inputs and validate form behavior under different conditions.
2. **API Functionality**
    - **Manual Testing:** Use tools like Postman to manually send requests to API endpoints and verify responses.
    - **Automated Testing:** Implement automated tests that send various requests to the API, validate responses, check data persistence, and error handling.
3. **Error Messages**

- **Manual Testing:** Trigger different types of errors by providing invalid inputs or causing system issues and observe the error messages displayed.
- **Automated Testing:** Automate tests to deliberately cause errors and verify that the correct messages are shown.

4. **UI Components**
    - **Manual Testing:** Inspect the UI for completeness, layout consistency, and responsiveness. Interact with each UI element to ensure it functions correctly.
    - **Automated Testing:** Use automated UI testing tools (e.g., Selenium, Cypress) to simulate user interactions and verify that UI components behave as expected.
5. **Navigation**
    - **Manual Testing:** Navigate through the application manually to ensure that all links and buttons work correctly and lead to the expected pages.
    - **Automated Testing:** Automate tests to simulate user navigation and verify that the correct pages are loaded.
6. **Database Interactions**
    - **Manual Testing:** Perform operations in the application and manually query the database to verify data correctness.
    - **Automated Testing:** Use automated tests to perform database operations and check the state of the database before and after these operations.
7. **Security Features**
    - **Manual Testing:** Test authentication and authorization by logging in with various user roles and checking access controls.
    - **Automated Testing:** Use security testing tools to automate the verification of security features such as session management and data encryption.
8. **Performance Aspects**
    - **Manual Testing:** Conduct basic performance checks by simulating user actions and measuring response times.
    - **Automated Testing:** Use performance testing tools (e.g., JMeter, LoadRunner) to simulate high loads and measure the application's performance under stress conditions.