1. Identify Potential Exceptions: Start by identifying potential exceptions and errors that can occur in your property management application. These can include database connection issues, data validation errors, external API failures, and custom business logic errors specific to your application.
2. Define Custom Exception Classes: In Salesforce Apex, you can define custom exception classes to encapsulate and represent different types of exceptions. This allows you to create a more structured and organized way to handle exceptions. For example, you might create custom exceptions like PropertyNotFoundException or TenantAlreadyExistsException.

apexCopy code

```apex
try {
   // Code that may throw exceptions
} catch (PropertyNotFoundException e) {
   // Handle PropertyNotFoundException
} catch (TenantAlreadyExistsException e) {
   // Handle TenantAlreadyExistsException
} catch (Exception e) {
   // Catch-all exception handler for unexpected errors
   // Log the error, notify administrators, or take appropriate action
}
```

Use Try-Catch Blocks: Implement try-catch blocks to capture and handle exceptions. When you have code that may throw an exception, place it within a try block, and then catch the exceptions that may occur in the catch block. This allows you to take specific actions when an exception occurs.

4. Logging and Notification: Implement logging to record details about the exception, such as the error message, stack trace, and context information. Use Salesforce's logging features or external logging services to store and monitor logs. Additionally, consider implementing notification mechanisms,

such as sending emails or alerts to system administrators or support teams when critical exceptions occur.

5. Graceful User Feedback: When an exception occurs, provide clear and user-friendly error messages to the end-users of your application. This helps them understand what went wrong and what steps to take next.
6. Unit Testing: Write unit tests to cover various exception scenarios. Unit tests are essential for validating that your exception handling code works as expected.
7. Security Considerations: Ensure that your exception handling does not leak sensitive information. Avoid displaying detailed error messages to end-users that could potentially reveal implementation details.
8. Continuous Improvement: Continuously monitor and analyze exceptions that occur in your application and make improvements to your exception handling strategy based on the patterns you observe.

Exception handling is a crucial aspect of maintaining the reliability and stability of your property management application. By proactively identifying and handling exceptions, you can improve the user experience and minimize the impact of unexpected issues.