System Design Specification

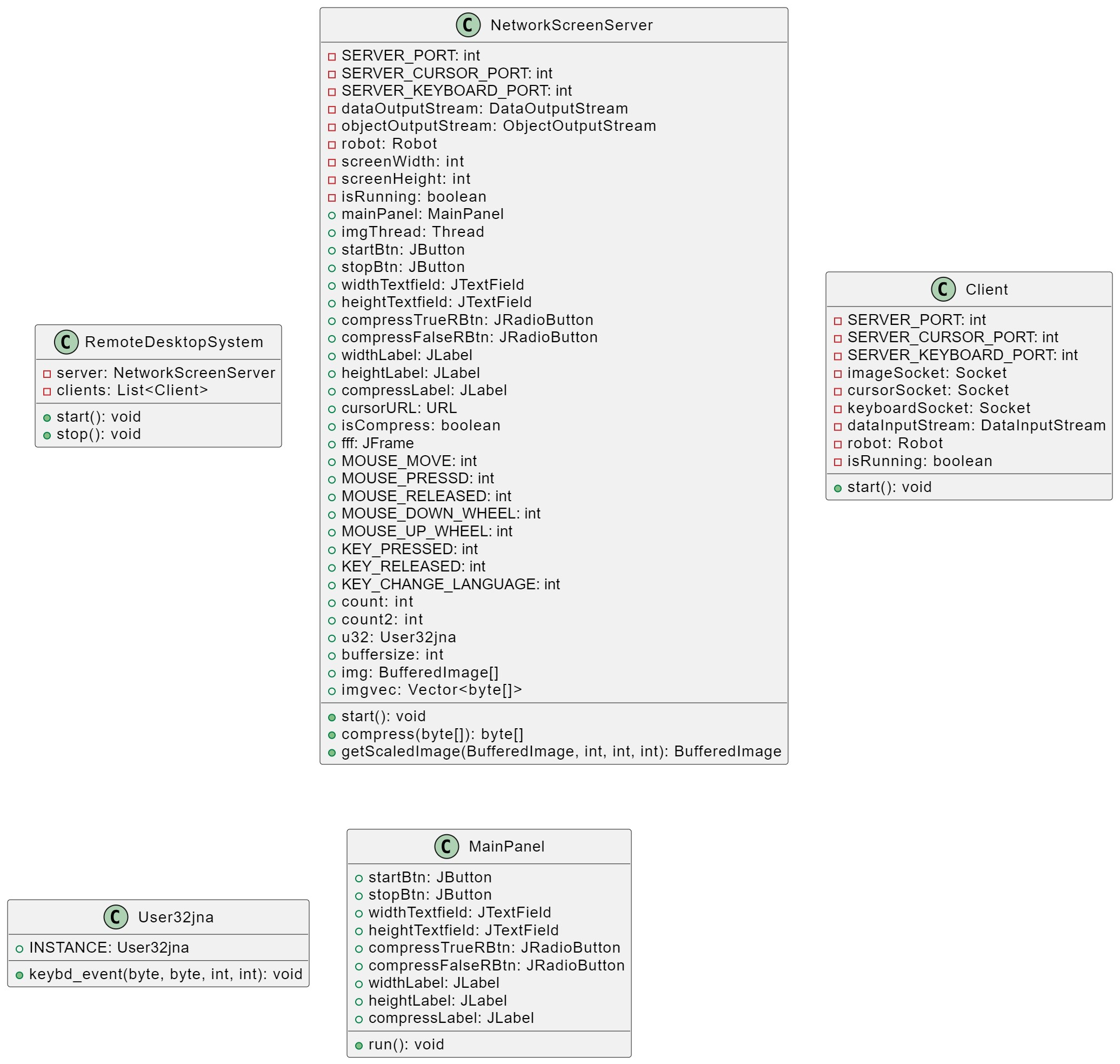
**213J15**

**219J05**

1. Overview:

- The system comprises a server (NetworkScreenServer) and clients.

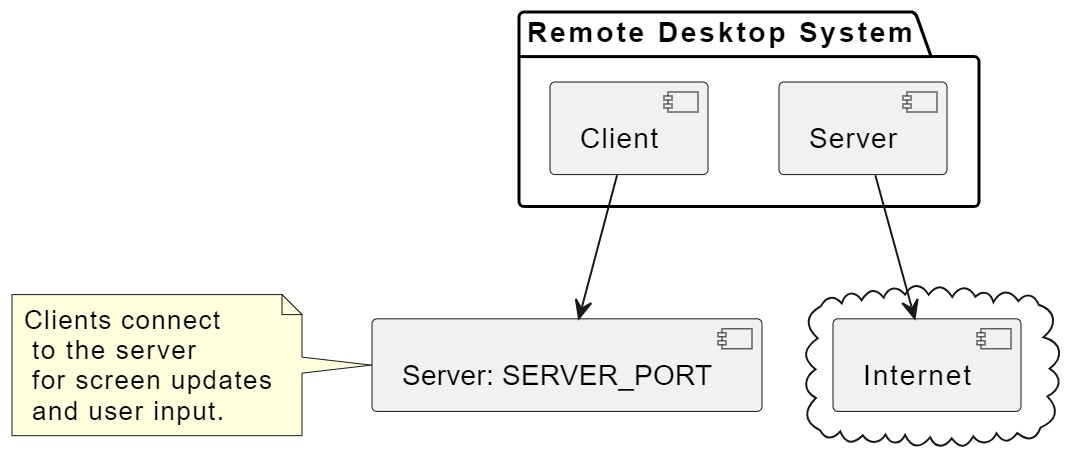
- Purpose: Remote screen sharing and control.



2. System Architecture:

- Client-Server Architecture:

- Clients connect to the server to request screen updates and send user input.



3. Components:

- Server Components:

- `NetworkScreenServer` class handling screen capturing, input events, and network communication.

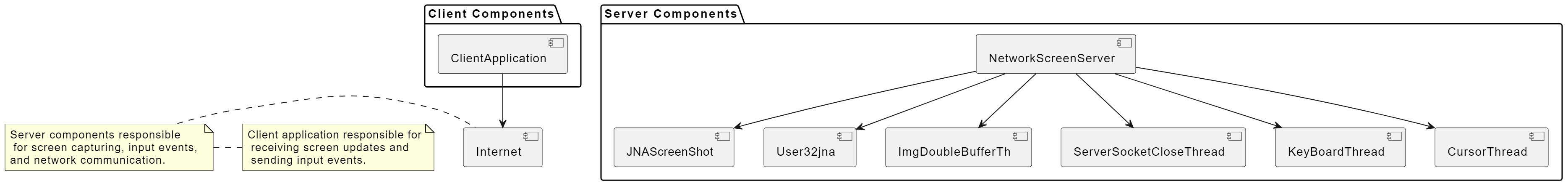
- `JNAScreenShot` class for screen capturing.

- `User32jna` interface for native keyboard events.

- `ImgDoubleBufferTh`, `ServerSocketCloseThread`, `KeyBoardThread`, `CursorThread` classes for threading and event handling.

- Client Components:

- Client application for receiving screen updates and sending input events.



4. Communication:

- Protocol:

- Use TCP/IP sockets for reliable communication.

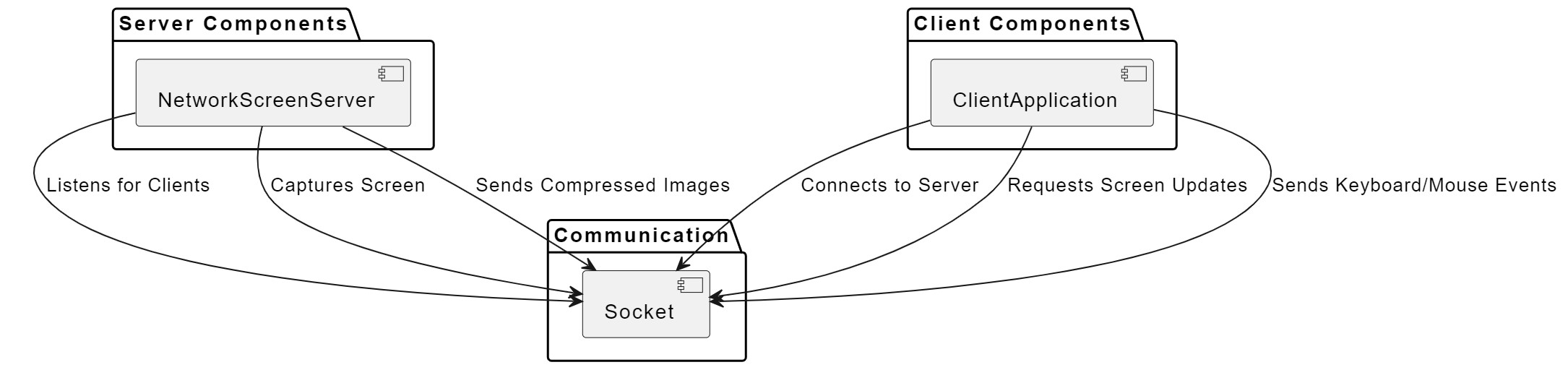
- Define a custom protocol for message exchange between server and clients (e.g., for screen updates, keyboard/mouse events).

- Data Flow:

- Clients request screen updates.

- Server captures the screen and sends compressed images to clients.

- Clients send keyboard/mouse events to the server.



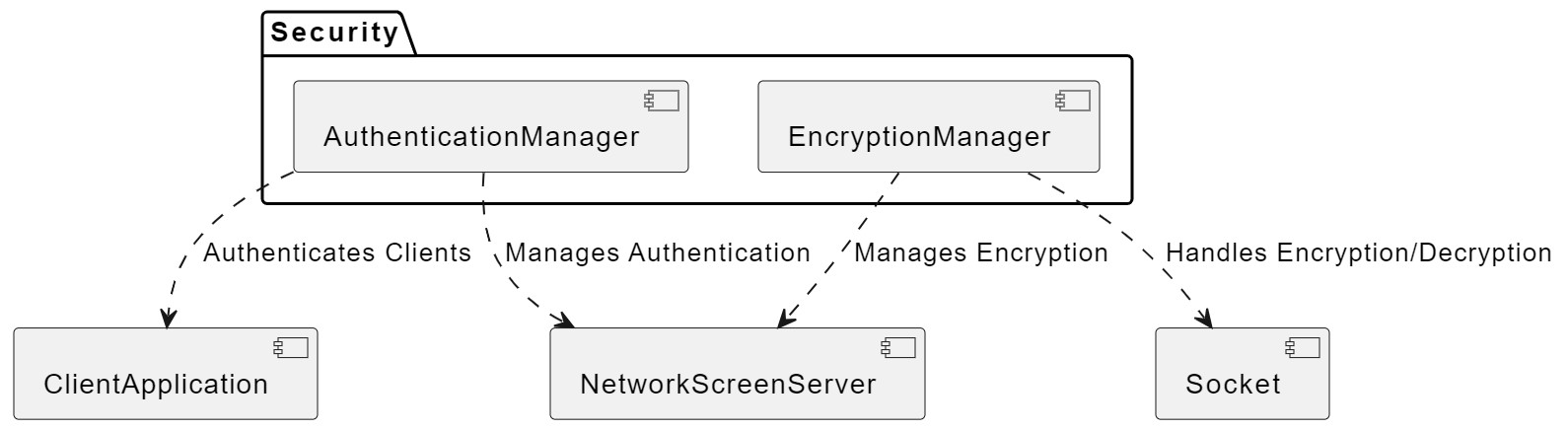
5. Security:

- Authentication:

- Implement a secure authentication mechanism for client-server communication.

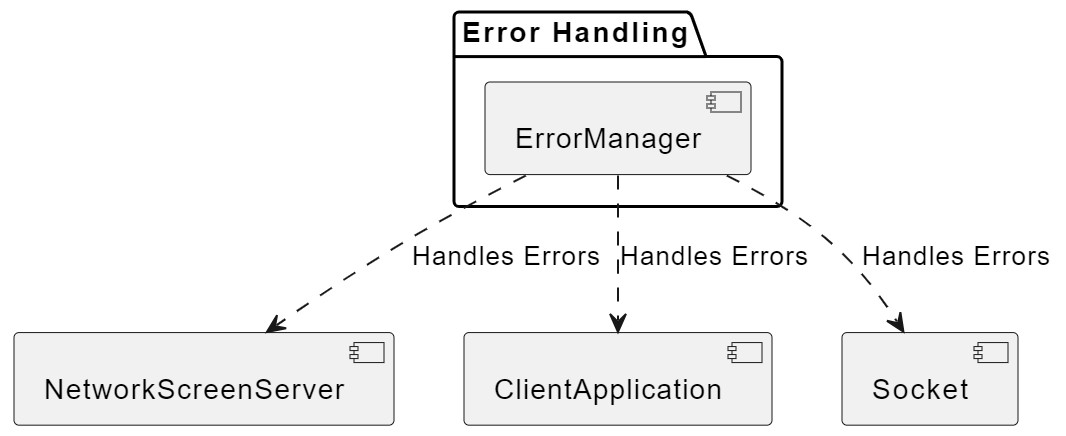
- Encryption:

- Use encryption for sensitive data transmission (e.g., SSL/TLS).



6. Error Handling:

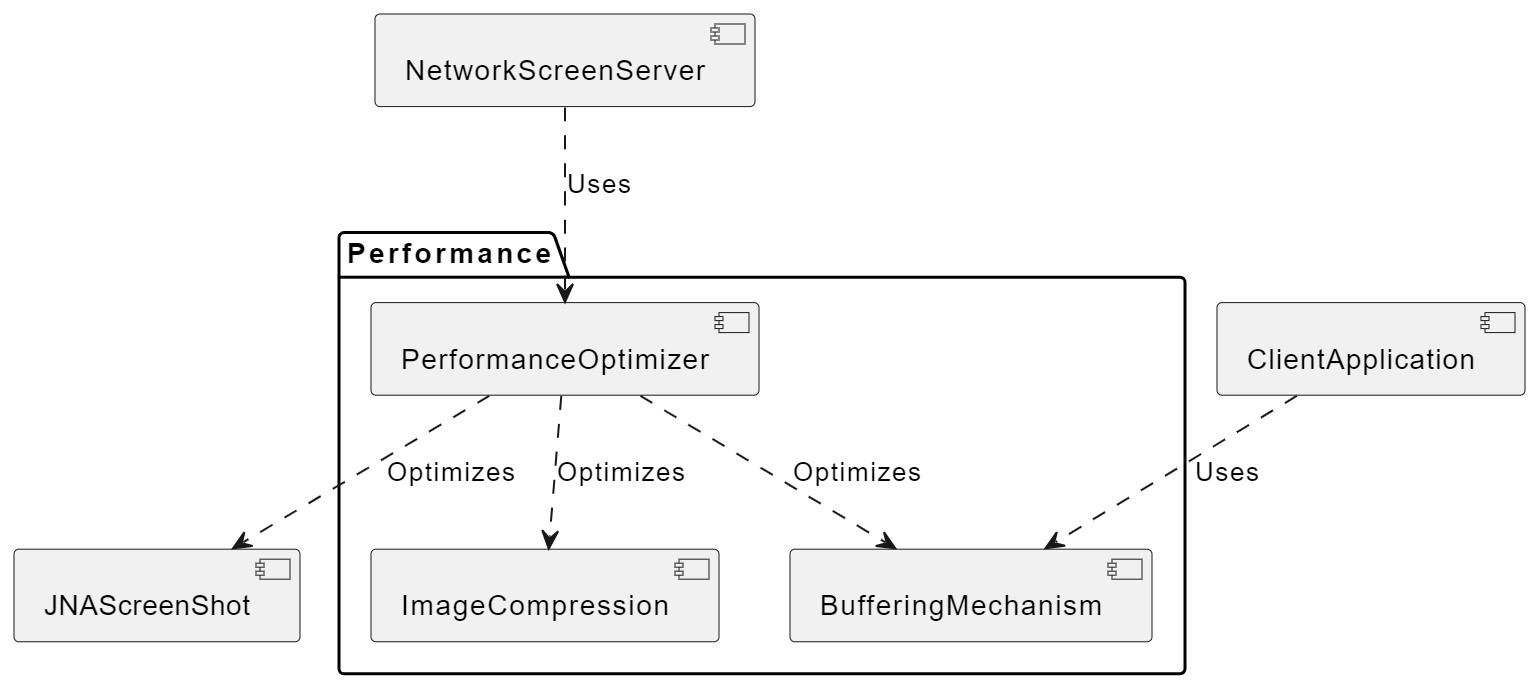
- Implement robust error handling mechanisms to manage network issues, client disconnects, and other exceptions.



7. Performance Considerations:

- Optimize screen capturing and image compression for efficient performance.

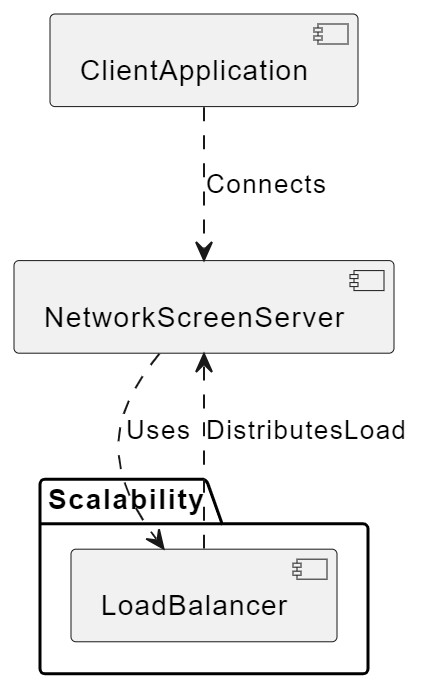
- Implement buffering mechanisms to handle data spikes.



8. Scalability:

- Design the system to handle multiple concurrent clients.

- Consider load balancing if required.



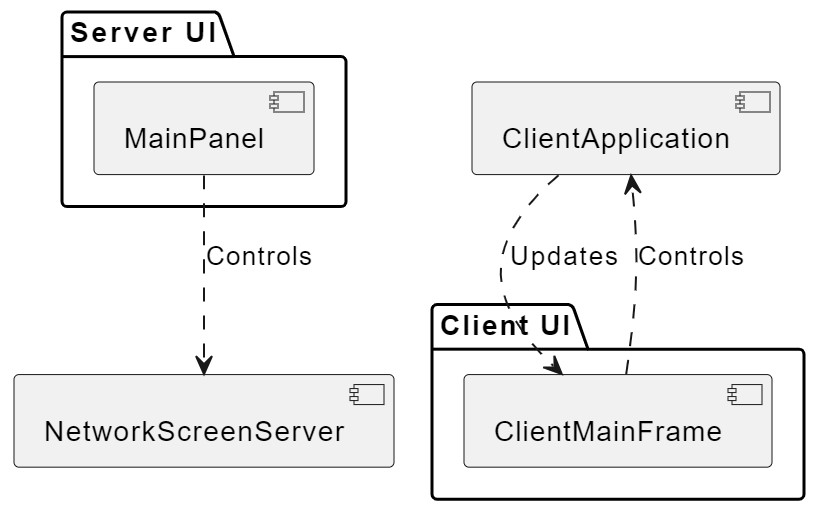
9. User Interface (UI):

- Server UI:

- `MainPanel` class provides a simple GUI for starting and stopping the server.

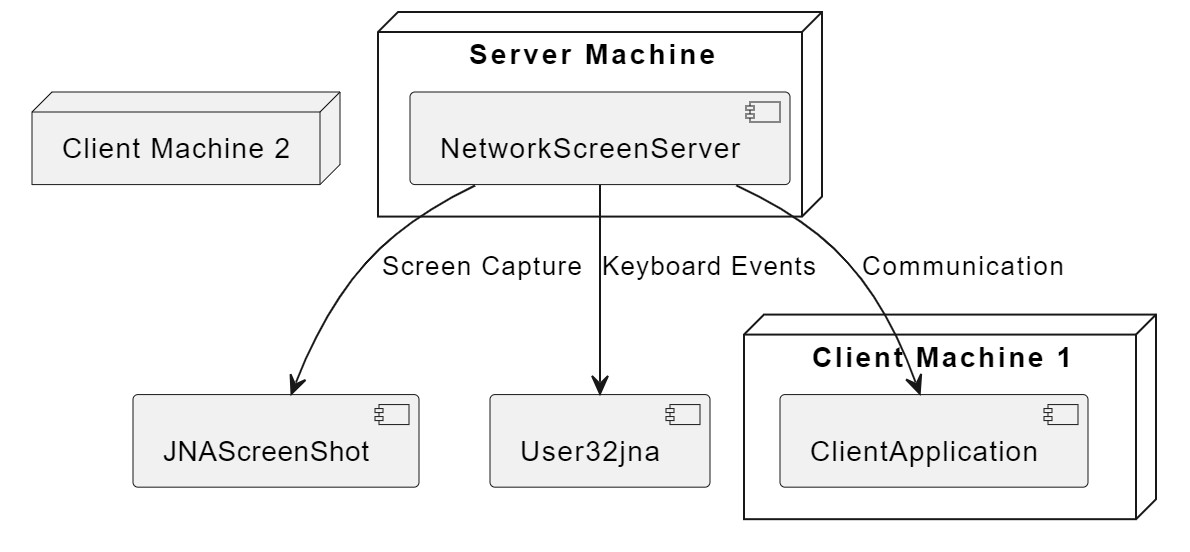
- Client UI:

- Implement a user-friendly interface for connecting to the server and interacting with the remote screen.



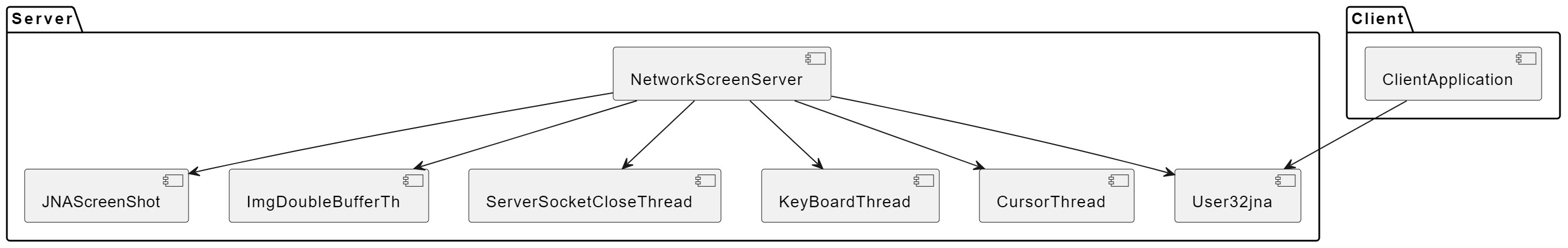
10. Deployment:

- Specify deployment details for server and client applications.



11. Dependencies:

- List external libraries and APIs used (e.g., JNA).



12. Testing:

12.1 Unit Testing:

12.1.1 Screen Capturing:

- Verify that the system can capture the screen correctly.

- Check if the captured images are of the expected dimensions.

12.1.2 Compression:

- Ensure that Snappy compression is working as expected.

- Test the compression process on sample images.

12.1.3 Input Handling:

- Test the server's ability to interpret and respond to keyboard and mouse events.

12.2 Integration Testing:

12.2.1 Server-Client Interaction:

- Confirm that clients can successfully connect to the server.

- Test the transmission of screen updates from server to client.

12.2.2 Input Synchronization:

- Verify that user input from clients is accurately reflected on the server.

12.3 Performance Testing:

12.3.1 Latency:

- Measure the time taken for screen updates to reach clients.

- Evaluate the overall system latency.

12.3.2 Load Testing:

- Test the system's performance under various loads.

- Check for any degradation in performance with an increasing number of clients.

12.4 Error Handling:

12.4.1 Network Issues:

- Simulate network issues and check if the system handles them gracefully.

- Ensure appropriate error messages are displayed.

12.4.2 Client Disconnects:

- Test the system's response when a client disconnects unexpectedly.

- Verify that the server continues to function correctly.

12.5 Usability Testing:

12.5.1 User Interface:

- Evaluate the user interfaces of the server and client applications for usability.

- Check if the interfaces are intuitive and user-friendly.

12.6 Documentation:

12.6.1 Code Documentation:

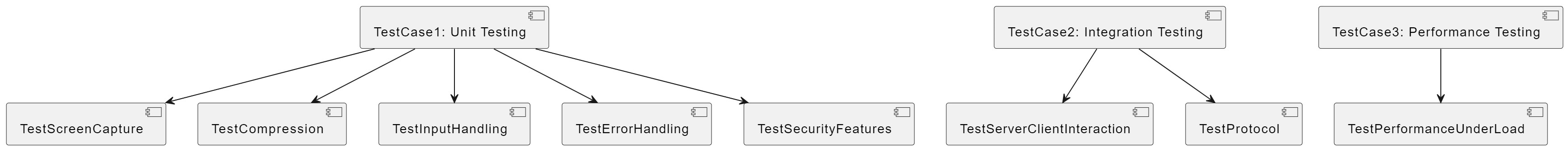
- Ensure that the code is well-documented, providing details on classes, methods, and variables.

- Check for the presence of inline comments explaining complex logic.

12.6.2 User Manuals:

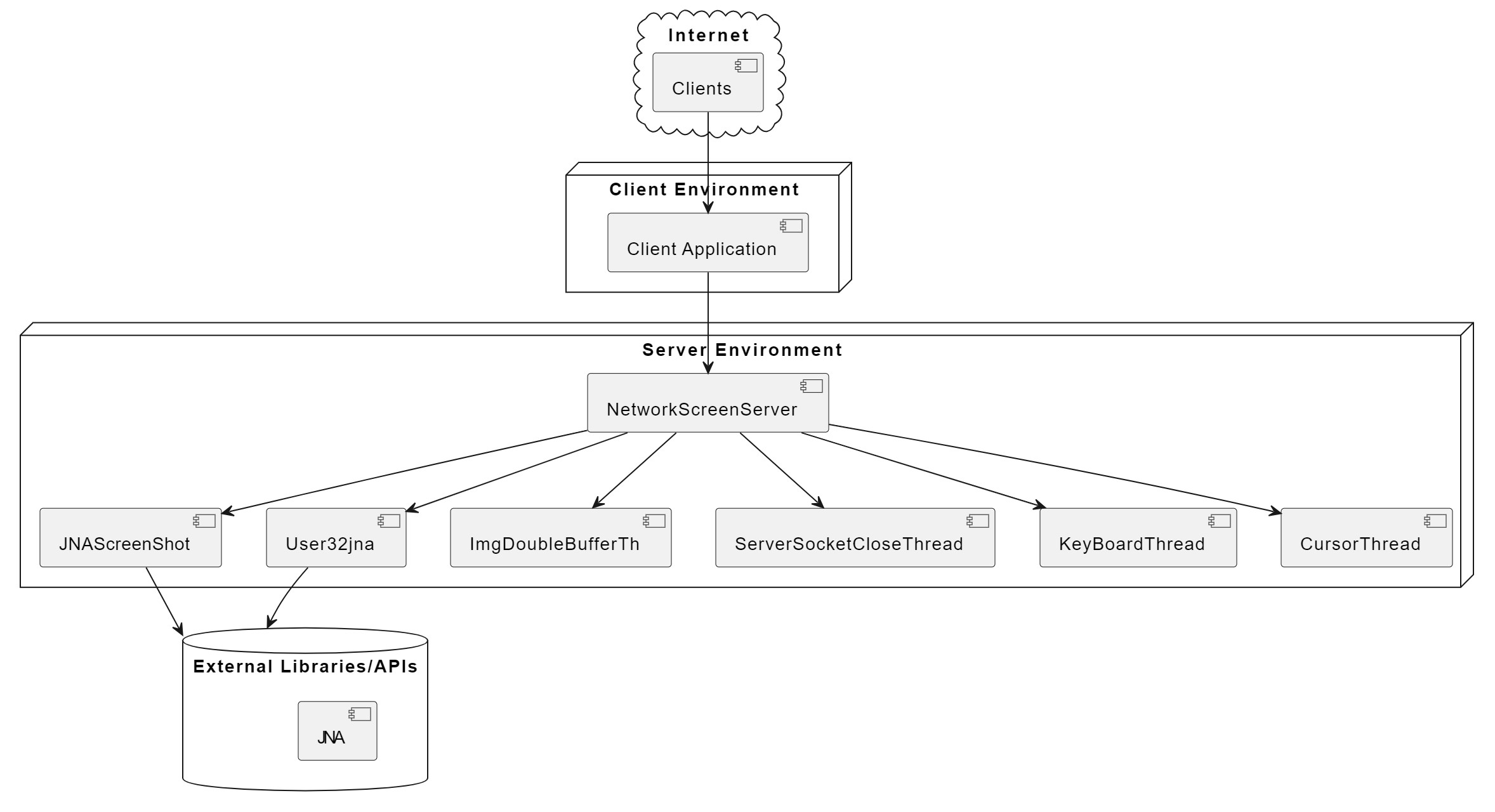
- Review user manuals for both the server and client applications.

- Confirm that the manuals provide clear instructions for setup and usage.



13. Documentation:

- Maintain comprehensive code comments and user documentation.



14. Future Enhancements:

1. Real-Time Collaboration: Enable multiple clients to collaborate on the same screen simultaneously, supporting teamwork and remote presentations.

2. File Transfer: Implement a secure file transfer mechanism between the server and clients to facilitate sharing of documents and files.

3. Cross-Platform Compatibility: Extend the system to support multiple operating systems, such as macOS and Linux, for broader usability.

4. Mobile Application: Develop a mobile application to allow users to connect and control the server from their smartphones or tablets.

5. Session Recording: Introduce a feature to record remote sessions for later review or documentation purposes.

6. Remote Access Management:Enhance security by implementing user access controls and permission levels for different clients.

7. Chat Functionality: Integrate a chat system between the server and clients to enable communication during remote sessions.

8. Improved Compression Algorithms: Research and implement advanced compression algorithms to further optimize bandwidth usage and improve performance.

9. Dynamic Screen Re sizing: Allow dynamic adjustment of screen resolution during an active session to accommodate varying client preferences.

10. Multi-Monitor Support: Extend support for clients with multiple monitors, providing a seamless remote experience for users with complex workstation setups.

