ChatRelayProject Test Plan

**Classes to be Tested**

From the project structure, these are the classes that need to be tested:

1. **Packet** - has tests in PacketTesting.java
2. **Chat** - has tests in ChatTest.java
3. **Message** – has tests in MessageTesting.java
4. **User/AbstractUser** - Needs tests \*\*
5. **DBManager**- Needs tests \*\*
6. **Server** - Needs tests \*\*
7. **Client/ClientHandler** - Needs tests \*\*
8. **ITAdmin** - Needs tests \*\*

**Operations to be Tested**

Packet Class:

* Constructor initialization
* getActionType() method
* getActionArguments() method
* getSenderId() method
* getTimeCreated() method
* Serialization/deserialization
* Various action types

Chat Class:

* Constructors
* Adding users to chat
* Removing users from chat
* Adding messages
* Privacy settings
* toString() representation

Message Class:

* Constructor initialization
* getCreatedAt() method
* getId() method
* getContent() method
* getChat() method
* getSender() method
* toString() method

User/AbstractUser Class:

* Constructor initialization
* User information getters (username, password, ID)
* Adding chats to user
* isAdmin and isDisabled status
* CreateChat, addUserToChat, sendMessage functionalities

DBManager Class:

* User loading/saving
* Chat loading/saving
* Message loading/saving
* ID-based retrieval methods
* Data persistence

Server Class:

* Connection handling
* Packet processing
* Client management
* Error handling

Client/ClientHandler Class:

* Connection to server
* Packet sending/receiving
* User interface integration
* Login/logout process
* Message sending

ITAdmin Class:

* User management (create, enable, disable)
* Chat log operations

**Specific Test Cases**

Packet Tests:

* Verify packet construction with all action types
* Test packet serialization and deserialization
* Test handling of null parameters
* Test timestamp accuracy

Chat Tests:

* Verify chat creation with appropriate ownership
* Test adding and removing users from chat
* Test privacy settings changes
* Test message addition to chat

Message Tests:

1. Test message construction with valid parameters
2. Test message ID uniqueness
3. Test message timestamp accuracy
4. Test message content retrieval
5. Test message-to-string conversion

User/AbstractUser Tests:

1. Test user creation with valid parameters
2. Test user ID uniqueness
3. Test adding chats to user
4. Test user roles (regular user vs. admin)
5. Test disabled status effects

DBManager Tests:

1. Test loading users from text files
2. Test loading chats from text files
3. Test loading messages from text files
4. Test user retrieval by ID
5. Test chat retrieval by ID
6. Test message retrieval by ID
7. Test data persistence (save & reload)
8. Test file format validation

Server Tests:

1. Test server initialization with various ports
2. Test client connection acceptance
3. Test packet reception and handling
4. Test concurrent client handling
5. Test error handling for invalid packets
6. Test server shutdown

Client/ClientHandler Tests:

1. Test client connection to server
2. Test login packet sending
3. Test logout packet sending
4. Test message sending
5. Test chat creation
6. Test response handling from server
7. Test disconnection handling

ITAdmin Tests:

1. Test user creation by admin
2. Test user disabling by admin
3. Test user enabling by admin
4. Test chat log extraction

Integration Tests:

1. End-to-end test: Login → Create Chat → Send Message → Logout
2. Admin operations test: Create User → Disable User → Enable User
3. Chat persistence test: Create Chat → Close App → Reopen → Verify Chat Exists
4. Multiple client test: Two clients sending messages to each other

**Instructions to Run the Test Suite**

Setup Instructions:

1. Ensure you have JUnit 5 in your classpath
2. Ensure the project is properly built
3. Create a test database folder structure with sample data for testing

Running the Test Suite:

1. From the command line:

*cd* /Users/(YourUserName)/Desktop/ChatRelayProject/ChatRelayProject/src

*javac* -cp .:junit-platform-console-standalone-1.8.2.jar chatRelay/\*.java unitTesting/\*.java

*java* -cp .:junit-platform-console-standalone-1.8.2.jar org.junit.platform.console.ConsoleLauncher --scan-classpath

1. From IDE (Eclipse/IntelliJ):
   * Right-click on the test package
   * Select "Run All Tests"

Expected Results:

* All tests should pass with no errors
* Console output should indicate successful test runs
* Any failures will be reported with the test name and reason

5. Test Suite Implementation

The following test classes need to be created:

1. MessageTest.java
2. UserTest.java
3. DBManagerTest.java
4. ServerTest.java
5. ClientTest.java
6. ITAdminTest.java
7. IntegrationTest.java

Each test class should follow the pattern shown in the existing PacketTest.java and ChatTest.java, and other test files with proper setup methods and JUnit 5 annotations.

Test Environment:

* Tests should run against a test database directory ([test](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html))
* Test files should be automatically created and cleaned up
* The test environment should be isolated from development/production

6. Testing Schedule

1. Unit Testing Phase:
   * Complete all individual class tests
   * Verify each class works in isolation
2. Integration Testing Phase:
   * Test component interactions
   * Verify message flow from client to server and back
3. System Testing Phase:
   * End-to-end tests with multiple clients
   * Performance testing under load

7. Reporting

Test results should be documented including:

* Number of tests passed/failed
* Code coverage metrics
* Any issues found during testing
* Performance benchmarks