COMP90015: Distributed Systems

Assignment 2 - Report

Shuangkun Fan

Student ID: 1131667

1. Introduction:

In this assignment, we were required to design and implement The Collaborative Whiteboard System by using Java that enables multiple users to concurrently interact with a shared canvas. The system allows users to draw freehand, create various shapes, input text, and communicate with each other through a chat window. The implementation utilizes Java Remote Method Invocation (RMI) to establish a networked environment where clients can collaborate in real-time.

Basic Features:

Shared Interactive white board: The system supports a single whiteboard shared among all connected clients. Users can draw, edit, and manipulate objects on the canvas in real-time.
Drawing Shapes: The whiteboard supports various shapes such as lines, circles, ovals, and rectangles. Users can create and modify these shapes using intuitive mouse interactions.
Text Inputting: Users can input text anywhere within the whiteboard, allowing for annotations, labels, or additional information.
Color Customization: Users have the freedom to choose their preferred color for drawing. The system provides a palette of at least 16 colors, enabling users to express their creativity.

Advanced Features:

Chat Window: The application incorporates a text-based chat window, enabling users to communicate with each other. Users can exchange messages, share ideas, and discuss their work, fostering collaboration and real-time interaction.

File Menu: The system includes a file menu with options such as new, open, save, saveAs, and close. These functionalities are managed by the designated "manager" user, who has control over the whiteboard's file operations.
User Management: The manager user possesses the ability to kick out specific users from the shared whiteboard session, facilitating moderation and control over the collaboration process.

2. System architecture

The shared whiteboard system is built upon three fundamental components: the server, client, and remote interfaces. The communication infrastructure relies on Java RMI, enabling seamless interaction between these components. Furthermore, a wrapper interface and wrapper class are implemented to facilitate communication between clients and the server. These wrappers encompass essential information such as Color, Shape, and String, allowing clients to synchronize their drawings with other connected peers.

The server plays a pivotal role as the central communication bridge, facilitating the exchange of information among all connected clients. Additionally, the server incorporates a clientManager module responsible for effectively managing each client's presence and interactions within the shared whiteboard session. This management functionality ensures smooth collaboration and coordination between multiple users.

The remote interfaces define the RMI methods that clients can invoke to interact with the server and other clients. These interfaces serve as the contractual agreement between the client and server, facilitating the communication protocol.
The client component comprises two main elements: the CanvasClient(clientUI) and the whiteboard(canvas. The CanvasClient(clientUI) manages all of the UI components of whiteboard and provides the user interface for clients, allowing them to interact with the whiteboard system. The canvas, on the other hand, serves as the drawing area where users can create and modify their artwork.

3. Design diagrams

1.

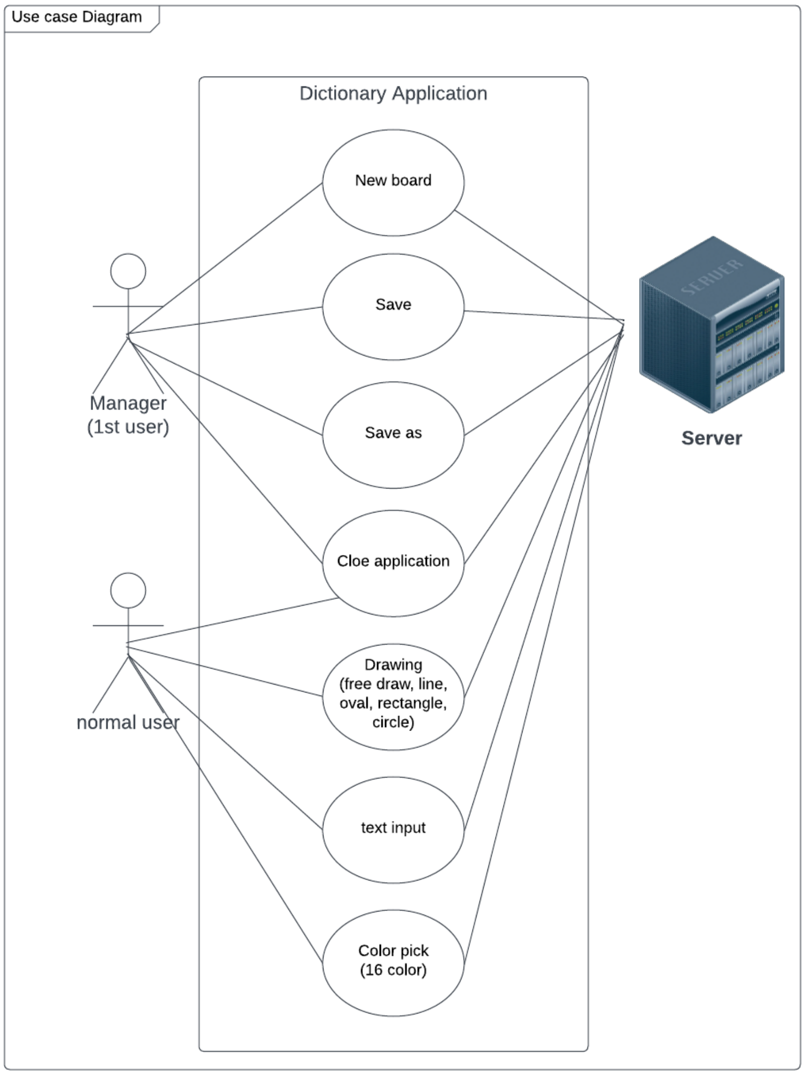

Figure 1 Use Case Diagram

2.

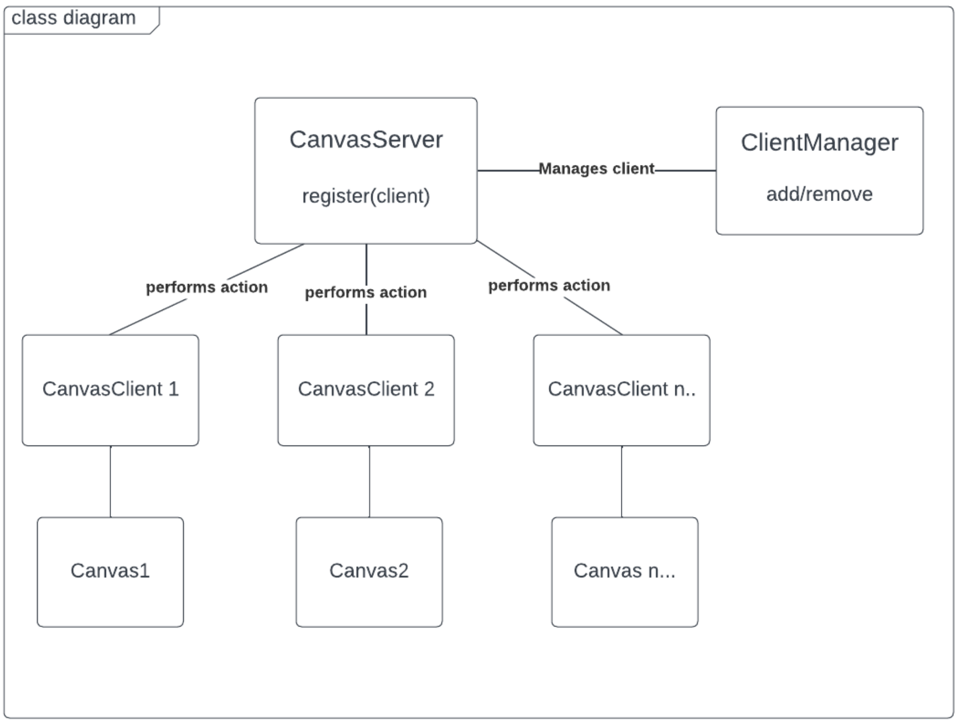

Figure 2 Class Diagram

4. Implementation details

As mentioned before, the project consists of three parts in total, so there are three packages, client, server and remote.

Server:

Server class: establishes communication by calling the RMI registry and binding the server service, and also handles the Exception in connection.

CanvasServer(Servant) class: the clients is managed through a ClientManager, enabling operations such as registration, removal, and addition. The first client to join becomes the manager, granting privileges to open, save the canvas. New users require the manager's permission to join. Additionally, CanvasServer class, encompasses all server functions and ensures the broadcast of whiteboard changes to all connected clients, promoting real-time synchronization and collaboration.

ClientManager: As its name implies, it contains all the methods for adding and removing clients

Remote:

Contains all the interfaces required by the server and client, including the CanvasClientInterface, CanvasServerInterface, and ClientMsgInterface, defining the communication protocol.

Client:

Canvas class: implementing the user interface of the canvas. It enables users to draw on the canvas and ensures real-time communication with minimal delay to all other clients through the invocation of RMI methods. The canvasClientUI also includes functionality to read images and export images from the canvas, providing flexibility and versatility in managing visual content. Through its seamless integration with the shared whiteboard system, the canvasClientUI empowers users to engage in collaborative drawing and effectively share their creations with other connected clients.

CanvasClient class: it manages all of the UI components of whiteboard and provides the user interface for clients, allowing them to interact with the whiteboard system including all basic and advanced features like different drawing and color picker. Additionally, each client is required to provide a unique name. If a name entered matches an existing active client's name, the user is prompted to enter another name until a unique and non-conflicting name is provided. The ClientUI component is responsible for displaying the names of currently active users, with the manager's name indicated by with a manager flag. The system also includes a chatbox service, allowing clients to engage in text-based communication with each other. Furthermore, clients have access to 16 color options for drawing customization. For the manager, the system implements features such as creating new canvases, saving, opening, and saving canvases under different names. The manager also has the capability to kick out clients from the shared session, providing moderation and control over user participation.

Message Wrapper Class: It defines The protocol for communication between a server and clients involves exchanging information regarding the different drawing shapes of a canvas, text input, the color of strokes, the coordinates of the mouse.