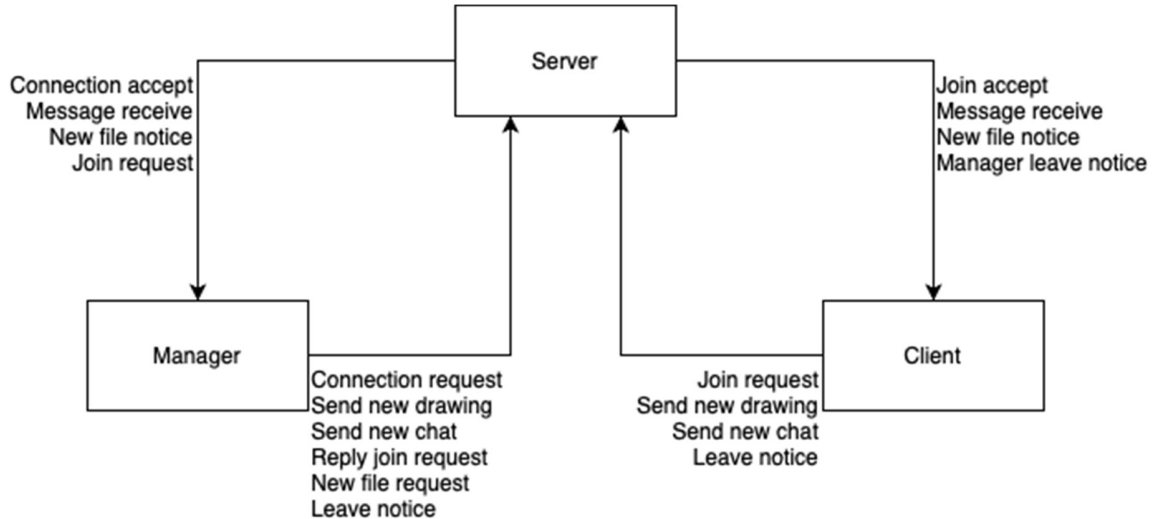# Report for COMP90015 Distributed Systems Assignment 2
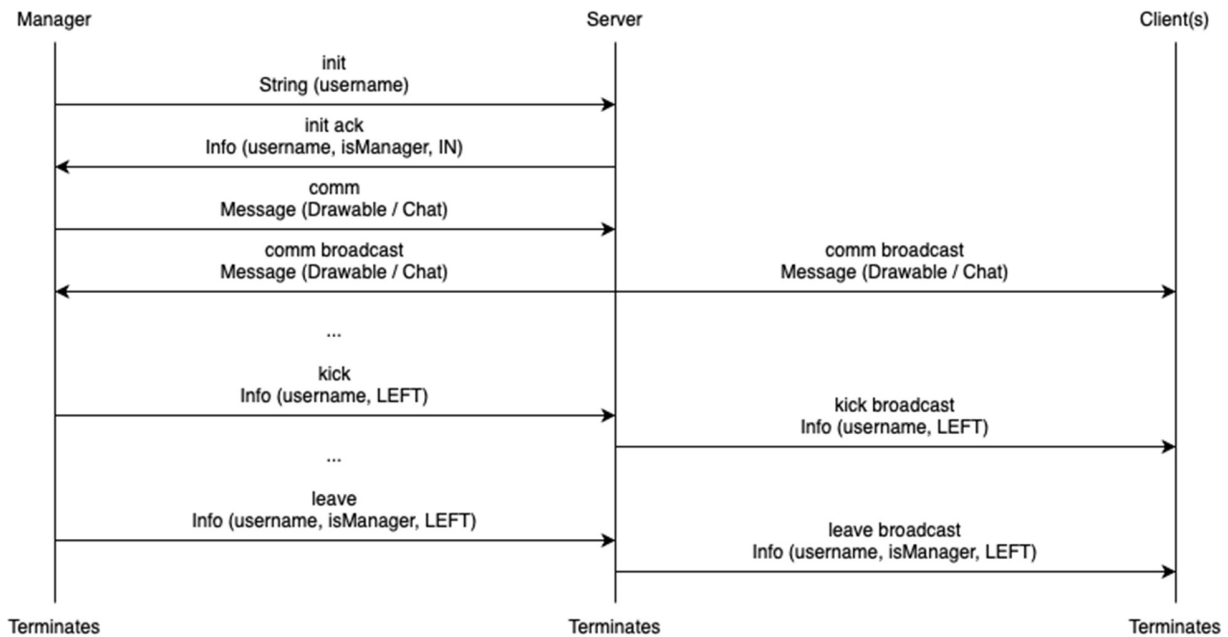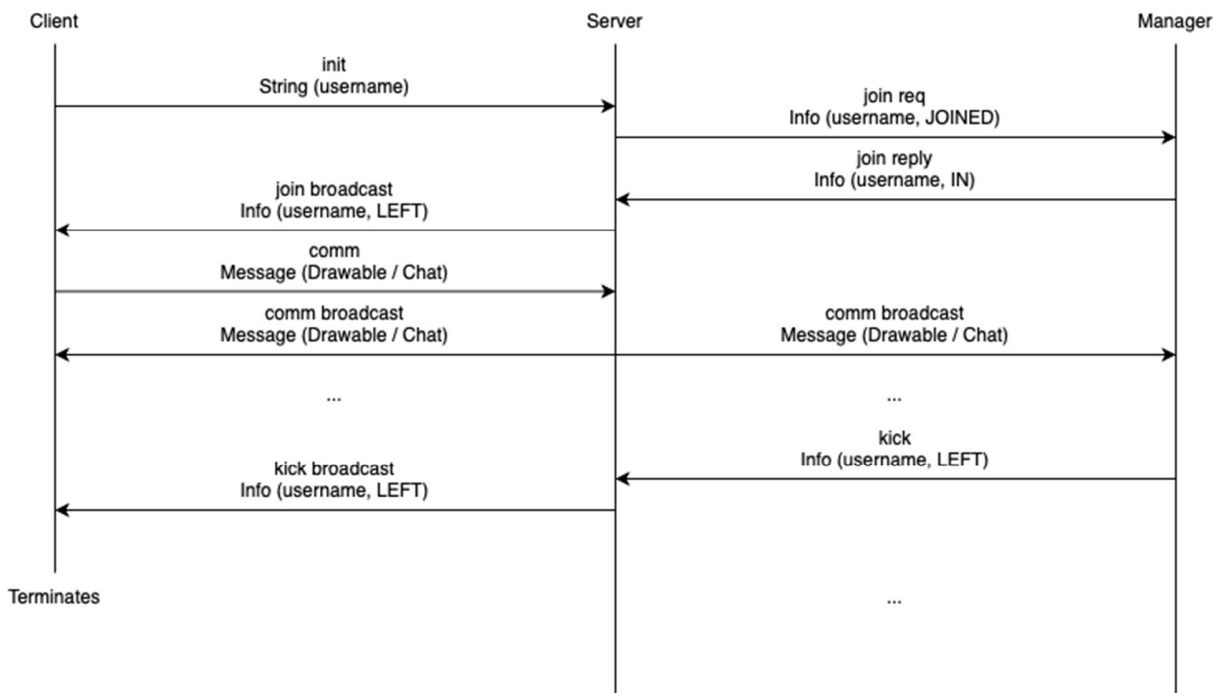
System Architecture

The system consists of a centralised server to serve between manager and clients and manage connections to the shared whiteboard. On start, the server will wait for the first client to connect, and the client becomes manager of the whiteboard. All clients connect later will require additional approval before joining the shared whiteboard, which we will discuss later in communication protocols. Apart from maintaining client connections, the server also holds all shapes drawn by all clients. This will be shared to newly joined clients once approved by the manager.

Communication Protocol

Every client connecting to the server will first send a String containing its username to server. If this is the first client, the server will reply an Info indicating the client is manager and enable manager related features on client side. If not, the server will send an Info to manager for approval. Manager will reply with modified Info that indicates if this client is approved. If yes, the server will send all previous Drawable to this client and start accepting its message, otherwise the server will inform the client and close connection. When collaborating on the shared whiteboard, every client will either send Drawable or Chat to the server, the server will then broadcast the message to all approved clients. On leave, the client will send an Info to server, the server will broadcast the message to all other clients to notify, then close the connection. If the leaving client is manager, then all other clients will close the connection as well. Additionally, the manager can kick out clients from the whiteboard by sending Info with this client's username. The server will broadcast the message to all clients and close the corresponding connection. The manager can also create a new whiteboard, the server will broadcast the message to all clients.

Diagram 1:

Manager — Server — Client(s)

- init
  String (username) (Manager → Server)
- init ack
  Info (username, isManager, IN) (Server → Manager)
- comm
  Message (Drawable / Chat) (Manager → Server)
- comm broadcast
  Message (Drawable / Chat) (Server → Manager) | comm broadcast
  Message (Drawable / Chat) (Server → Client(s))
- ...
- kick
  Info (username, LEFT) (Manager → Server) | kick broadcast
  Info (username, LEFT) (Server → Client(s))
- ...
- leave
  Info (username, isManager, LEFT) (Manager → Server) | leave broadcast
  Info (username, isManager, LEFT) (Server → Client(s))

Terminates — Terminates — Terminates

Communication flow of manager join, using whiteboard, kick out user and leave.

Diagram 2:

Client — Server — Manager

- init
  String (username) (Client → Server)
- join req
  Info (username, JOINED) (Server → Manager)
- join reply
  Info (username, IN) (Manager → Server)
- join broadcast
  Info (username, LEFT) (Server → Client)
- comm
  Message (Drawable / Chat) (Client → Server)
- comm broadcast
  Message (Drawable / Chat) (Server → Client) | comm broadcast
  Message (Drawable / Chat) (Server → Manager)
- ...
- kick
  Info (username, LEFT) (Manager → Server)
- kick broadcast
  Info (username, LEFT) (Server → Client)

Terminates — ... — ...

Communication flow of client join, using whiteboard and kicked out by manager.

Message Formats

The server and clients communicate by sending Serializable objects to each other. For each message, it contains one object of the following classes: Drawable, Info, and Chat.

Design Diagram

Paint
ⓕ ○	drawables	ArrayList<Drawable>
ⓕ ○	selfInfo	Info
ⓕ ○	userList	DefaultListModel<String>
ⓕ ○	*ELLIPSE*	int
ⓕ 🔒	txt_chat	JTextArea
ⓕ 🔒	bar	JMenuBar
ⓕ ○	*FREE*	int
ⓕ ○	socket	Socket
ⓕ ○	*LINE*	int
ⓕ ○	status	JLabel
ⓕ ○	start	Point
ⓕ 🔒	in	ObjectInputStream
ⓕ ○	out	ObjectOutputStream
ⓕ ○	currentColor	Color
ⓕ ○	pane	CanvasPane
ⓕ 🔒	username	String
ⓕ ○	*RECTANGLE*	int
ⓕ ○	current	Drawable
ⓕ ○	*TRIANGLE*	int
ⓕ ○	*TEXT*	int
ⓕ ○	selected	int

Server
ⓕ 🔒	*threads*	ArrayList<Thread>
ⓕ 🔒	*clients*	HashMap<String, ObjectOutputStream>
ⓕ 🔒	*managerIn*	ObjectInputStream
ⓕ 🔒	*drawables*	ArrayList<Drawable>
ⓕ 🔒	*managerOut*	ObjectOutputStream
ⓕ 🔒	*manager*	Socket
ⓕ 🔒	*server*	ServerSocket
ⓕ 🔒	*port*	int
ⓕ 🔒	*infos*	HashMap<String, Info>

Drawable
ⓕ ○	shape	Shape
ⓕ ○	text	String
ⓕ ○	color	Color
ⓕ ○	point	Point

Message
ⓕ ○	drawable	Drawable
ⓕ 🔒	info	Info
ⓕ 🔒	chat	Chat

Info
ⓕ 🔒	action	int
ⓕ 🔒	*LEFT*	int
ⓕ 🔒	*JOINED*	int
ⓕ 🔒	*IN*	int
ⓕ 🔒	isManager	boolean
ⓕ 🔒	username	String

Chat
ⓕ 🔒	message	String
ⓕ 🔒	recipient	String