```
const [windowSize, setWindowSize] = useState<number>(1);
	const [transmitterFrames, setTransmitterFrames] = useState<FrameT[]>([]);
	const [receiverFrames, setReceiverFrames] = useState<FrameT[]>([]);
	const [[windowStart, windowEnd], setWindow] = useState<[number, number]>([
		0, 0,
	]);

	const [logContent, setLogContent] = useState<string[]>([]);

	const resetSim = () => {
		setWindowSize(1);
		setWindow([0, 1]);
		setTransmitterFrames([]);
		setReceiverFrames([]);
		setLogContent(["Reset!"]);
	};

	const addTransmitterFrame = () => {
		let frameData = {
			ackReceived: false,
			frameType: FrameType.DAT,
			sequenceNumber: transmitterFrames.length,
		};
		setTransmitterFrames((prevFrames) => {
			return [...prevFrames, frameData];
		});
		setLogContent((prevContent) => {
			return [
				...prevContent,
				`Transmitter transmitted frame with Sequence No ${
{frameData.sequenceNumber}`,
			];
		});

		return frameData;
	};

	const slideWindowForward = () => {
		setWindow(([prevWindowStart, prevWindowEnd]) => {
			return [prevWindowStart + 1, prevWindowEnd + 1];
		});
	};

	const acknowledgeFrame = (frameNo: FrameT["sequenceNumber"]) => {
		if (frameNo < windowStart || frameNo ⩾ windowEnd) {
			return;
		}
		const allPreviousTransmittedFramesAck = transmitterFrames.every(
			(transmitFrame) => {
				const { sequenceNumber, ackReceived } = transmitFrame;
				if (sequenceNumber < windowStart) {
					return true;
				}
				return ackReceived;
			}
		);
		if (!allPreviousTransmittedFramesAck) {
			// Retransmit all previous frames
			const nonAckFrames = transmitterFrames.filter((transmitFrame) => {
				const { sequenceNumber } = transmitFrame;
				return (
					sequenceNumber < frameNo && sequenceNumber ⩾ windowStart
				);
			});
```

```
                setTransmitterFrames((prevFrames) => {
                        return [
                                ...prevFrames,
                                ...nonAckFrames.map((nonAckFrame, frameIdx) => {
                                        setLogContent((prevLog) => {
                                                return [
                                                        ...prevLog,
                                                        ...nonAckFrames.map(() => {
                                                                return `Retransmit frame ${
                                                                        nonAckFrame.sequenceNumber
                                                                } as frame ${prevFrames.length +
frameIdx}`;
                                                        }),
                                                ];
                                        });
                                        return {
                                                ...nonAckFrame,
                                                sequenceNumber: prevFrames.length + frameIdx,
                                                ackReceived: false,
                                                reTransmit: nonAckFrame.sequenceNumber,
                                        };
                                }),
                        ];
                });
                setReceiverFrames((prevFrames) => {
                        return [
                                ...prevFrames,
                                ...nonAckFrames.map((nonAckFrame, frameIdx) => {
                                        setLogContent((prevLog) => {
                                                return [
                                                        ...prevLog,
                                                        ...nonAckFrames.map(() => {
                                                                return `Re-Receive frame ${
                                                                        nonAckFrame.sequenceNumber
                                                                } as frame ${prevFrames.length +
frameIdx}`;
                                                        }),
                                                ];
                                        });
                                        return {
                                                ...nonAckFrame,
                                                sequenceNumber: prevFrames.length + frameIdx,
                                                ackReceived: false,
                                                reTransmit: nonAckFrame.sequenceNumber,
                                        };
                                }),
                        ];
                });
        }
        setReceiverFrames((prevFrames) => {
                return prevFrames.map((frame) => {
                        if (frame.sequenceNumber === frameNo) {
                                setLogContent((prevLog) => {
                                        return [
                                                ...prevLog,
                                                `Receiver sent acknowledgement for frame $
{frameNo}`,
                                        ];
                                });
                                return { ...frame, ackReceived: true };
                        }
                        return frame;
                });
```

```
                });
                setTransmitterFrames((prevFrames) => {
                        return prevFrames.map((frame) => {
                                if (frame.sequenceNumber === frameNo) {
                                        setLogContent((prevLog) => {
                                                return [
                                                        ...prevLog,
                                                        `Transmitter received acknowledgement for frame ${
frameNo}`,
                                                ];
                                        });
                                        return { ...frame, ackReceived: true };
                                }
                                return frame;
                        });
                });
                slideWindowForward();
        };

        useEffect(() => {
                setWindow(([prevWindowStart]) => {
                        return [prevWindowStart, prevWindowStart + windowSize];
                });
        }, [windowSize]);
```