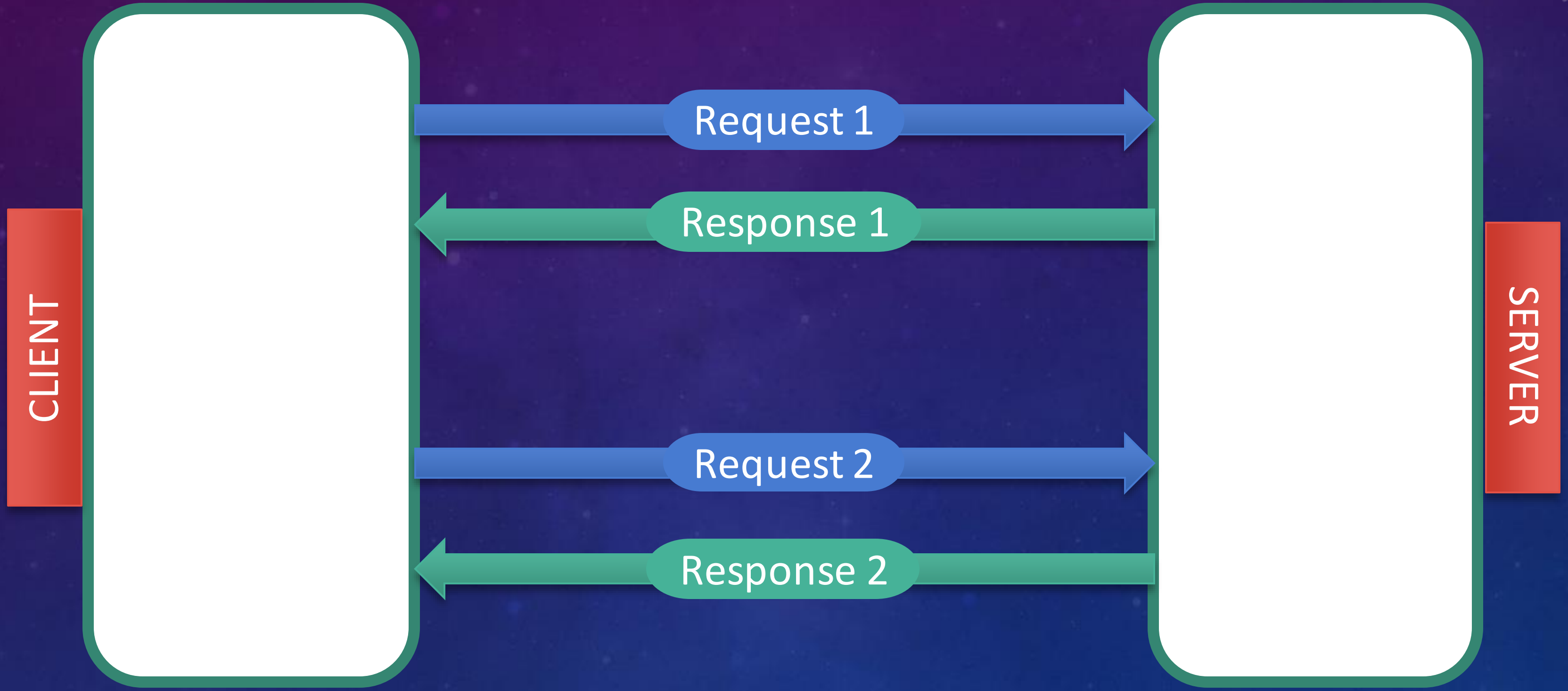
WEBSOCKET CHAT APPLICATION

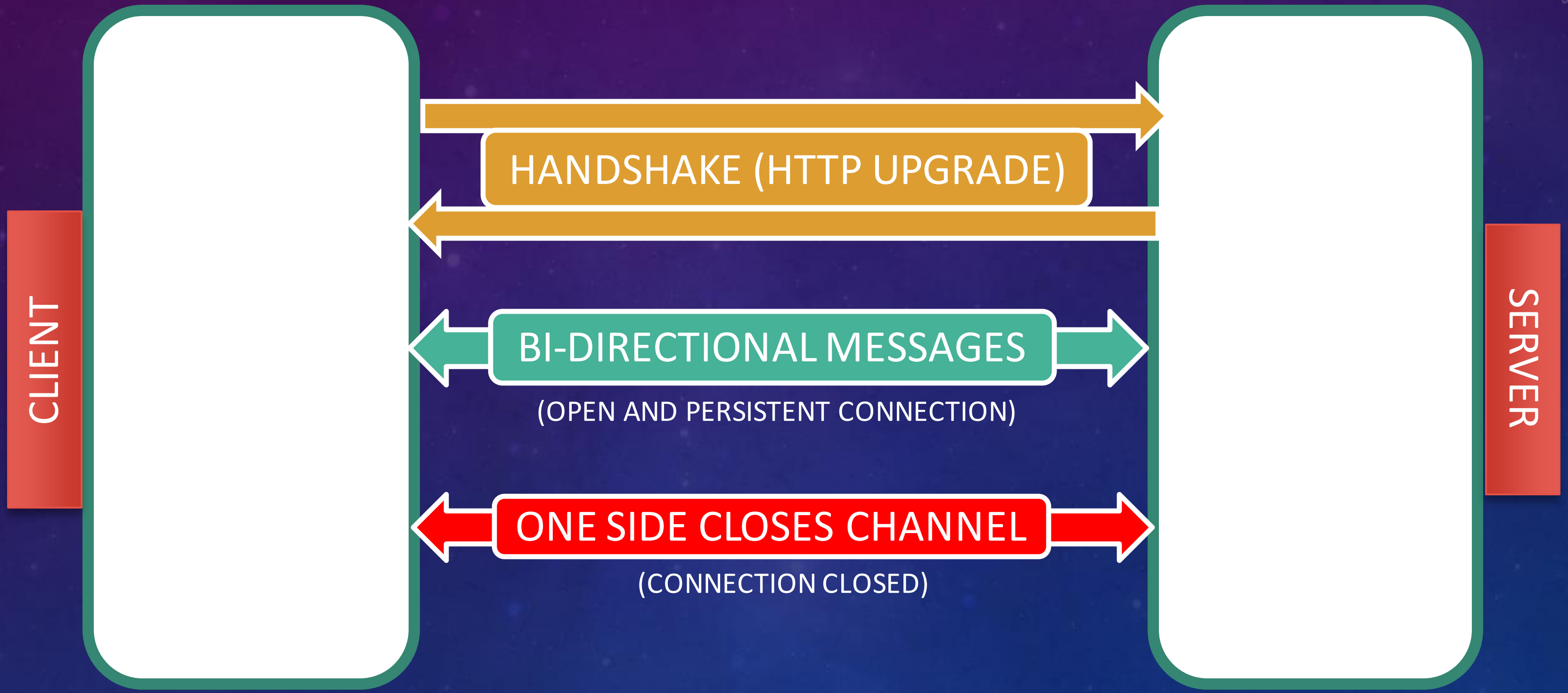
PRESENTER: JAYASHANKAR MANGINA

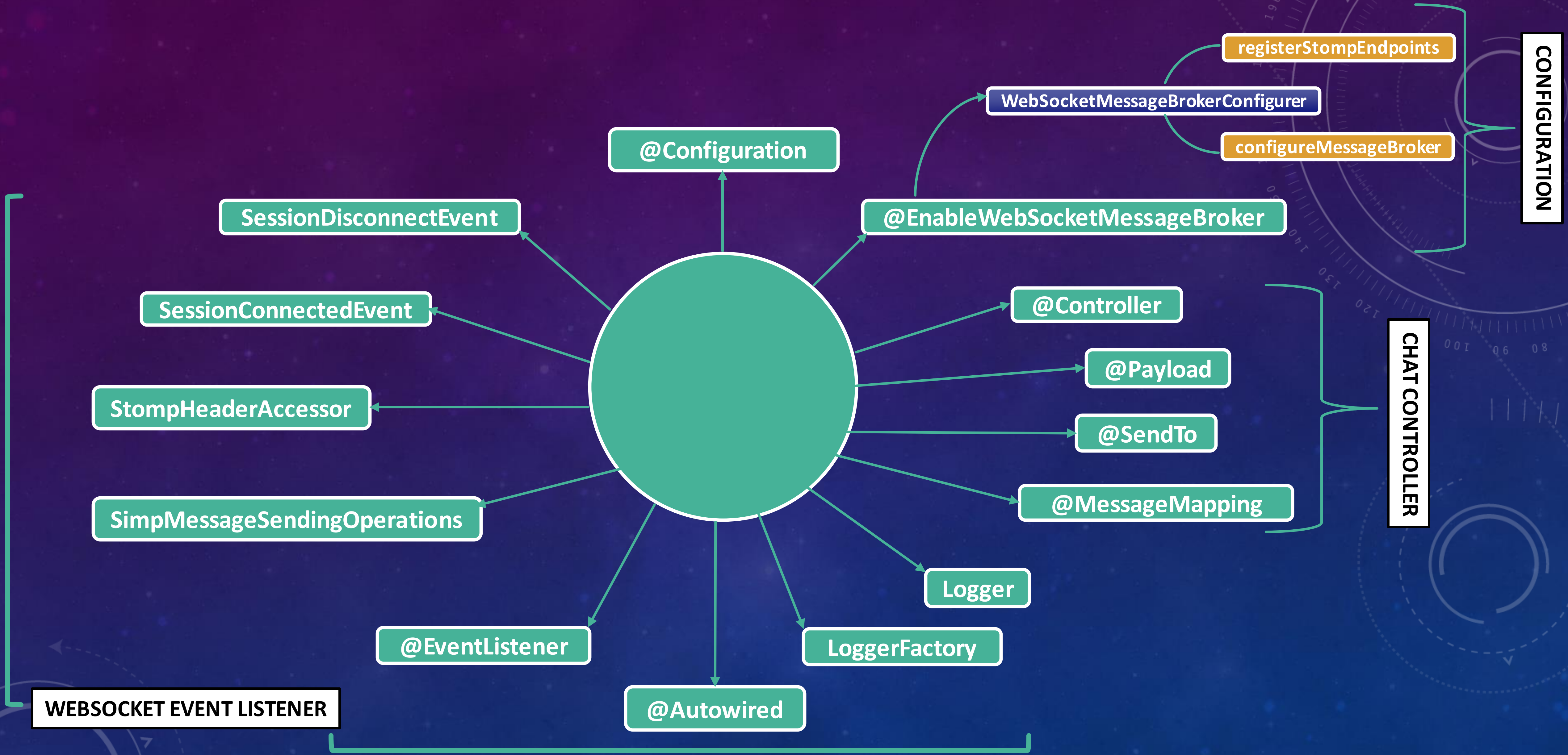
DATE: DECEMBER ,2021.

HTTP REQUEST– RESPONSE CYCLE

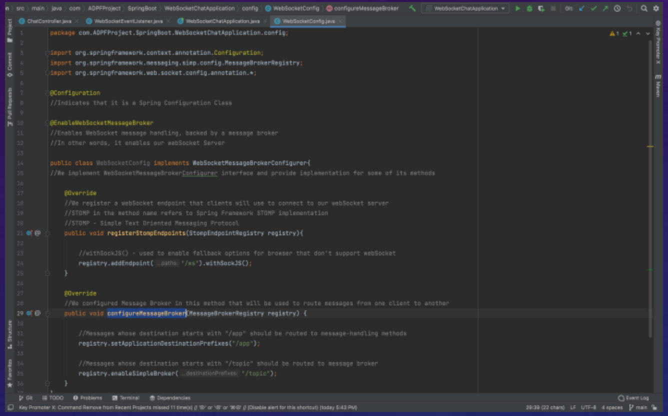


WEBSOCKET CONNECTION CYCLE

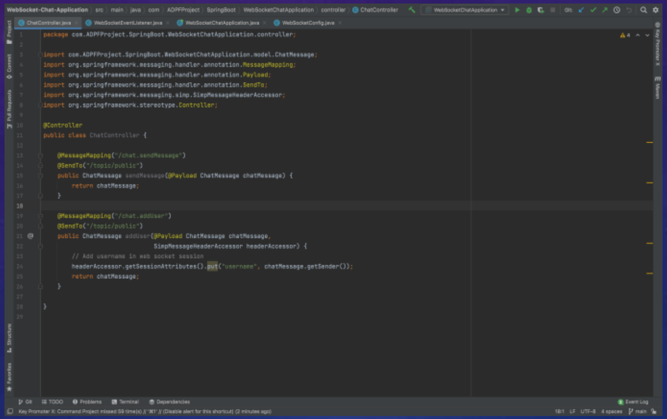



@Configuration
SessionDisconnectEvent
@EnableWebSocketMessageBroker
WebSocketMessageBrokerConfigurer
registerStompEndpoints
configureMessageBroker
CONFIGURATION
SessionConnectedEvent
@Controller
@Payload
StompHeaderAccessor
@SendTo
SimpMessageSendingOperations
@MessageMapping
CHAT CONTROLLER
Logger
@EventListener
LoggerFactory
@Autowired
WEBSOCKET EVENT LISTENER

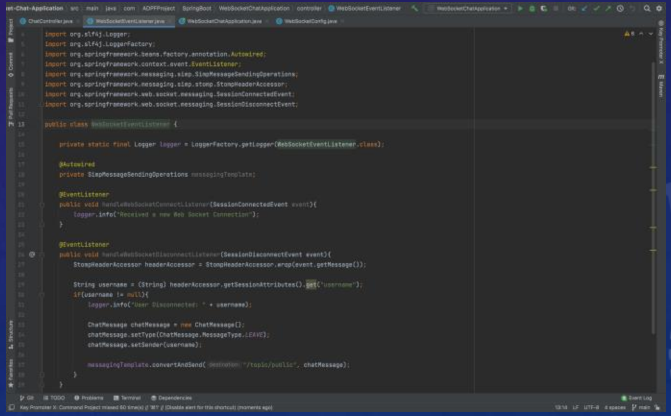
SAMPLE SCREENSHOTS



```java
package com.ADPFProject.SpringBoot.WebSocketChatApplication.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.messaging.simp.config.MessageBrokerRegistry;
import org.springframework.web.socket.config.annotation.*;

@Configuration
//Indicates that it is a Spring Configuration Class

@EnableWebSocketMessageBroker
//Enables WebSocket message handling, backed by a message broker
//In other words, it enables our webSocket Server

public class WebSocketConfig implements WebSocketMessageBrokerConfigurer{
//We implement WebSocketMessageBrokerConfigurer interface and provide implementation for some of its methods

    @Override
    //We register a webSocket endpoint that clients will use to connect to our webSocket server
    //STOMP in the method name refers to Spring Framework STOMP implementation
    //STOMP - Simple Text Oriented Messaging Protocol
    public void registerStompEndpoints(StompEndpointRegistry registry){

        //withSockJS() - used to enable fallback options for browser that don't support webSocket
        registry.addEndpoint( ...paths: "/ws").withSockJS();
    }

    @Override
    //We configured Message Broker in this method that will be used to route messages from one client to another
    public void configureMessageBroker(MessageBrokerRegistry registry) {

        //Messages whose destination starts with "/app" should be routed to message-handling methods
        registry.setApplicationDestinationPrefixes("/app");

        //Messages whose destination starts with "/topic" should be routed to message broker
        registry.enableSimpleBroker( ...destinationPrefixes: "/topic");
    }
```



```java
package com.ADPFProject.SpringBoot.WebSocketChatApplication.controller;

import com.ADPFProject.SpringBoot.WebSocketChatApplication.model.ChatMessage;
import org.springframework.messaging.handler.annotation.MessageMapping;
import org.springframework.messaging.handler.annotation.Payload;
import org.springframework.messaging.handler.annotation.SendTo;
import org.springframework.messaging.simp.SimpMessageHeaderAccessor;
import org.springframework.stereotype.Controller;

@Controller
public class ChatController {

    @MessageMapping("/chat.sendMessage")
    @SendTo("/topic/public")
    public ChatMessage sendMessage(@Payload ChatMessage chatMessage) {
        return chatMessage;
    }

    @MessageMapping("/chat.addUser")
    @SendTo("/topic/public")
    public ChatMessage addUser(@Payload ChatMessage chatMessage,
                               SimpMessageHeaderAccessor headerAccessor) {
        // Add username in web socket session
        headerAccessor.getSessionAttributes().put("username", chatMessage.getSender());
        return chatMessage;
    }

}
```



```java
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.event.EventListener;
import org.springframework.messaging.simp.SimpMessageSendingOperations;
import org.springframework.messaging.simp.stomp.StompHeaderAccessor;
import org.springframework.web.socket.messaging.SessionConnectedEvent;
import org.springframework.web.socket.messaging.SessionDisconnectEvent;

public class WebSocketEventListener {

    private static final Logger logger = LoggerFactory.getLogger(WebSocketEventListener.class);

    @Autowired
    private SimpMessageSendingOperations messagingTemplate;

    @EventListener
    public void handleWebSocketConnectListener(SessionConnectedEvent event){
        logger.info("Received a new Web Socket Connection");
    }

    @EventListener
    public void handleWebSocketDisconnectListener(SessionDisconnectEvent event){
        StompHeaderAccessor headerAccessor = StompHeaderAccessor.wrap(event.getMessage());

        String username = (String) headerAccessor.getSessionAttributes().get("username");
        if(username != null){
            logger.info("User Disconnected: " + username);

            ChatMessage chatMessage = new ChatMessage();
            chatMessage.setType(ChatMessage.MessageType.LEAVE);
            chatMessage.setSender(username);

            messagingTemplate.convertAndSend( destination: "/topic/public", chatMessage);
        }
    }
```

THE END