```rust
mod job;
mod printer;

use actix_web::{web, App, HttpRequest, HttpServer, Responder, HttpResponse};
use job::Job;
use printer::Printer;
use chrono::Utc;
use std::sync::Mutex;

struct AppState {
    printer: Mutex<Printer>,
}

#[derive(serde::Deserialize)]
struct PrintRequest {
    priority: u32,
    team_name: String,
    file_content: String,
    color: bool,
}

async fn print_job(
    data: web::Data<AppState>,
    req: web::Json<PrintRequest>,
) -> impl Responder {
    let job = Job::new(
        req.priority,
        req.team_name.clone(),
        Utc::now(),
        req.file_content.clone(),
        req.color,
    );

    let printer = data.printer.lock().unwrap();
    match printer.submit_task(job) {
        Ok(_) => HttpResponse::Ok().body("⊠ ⊠ ⊠ ⊠ ⊠ ⊠ ⊠ ⊠ "),
        Err(_) => HttpResponse::Conflict().body("⊠ ⊠ ⊠ ⊠ ⊠ ⊠ ⊠ ⊠ ⊠ ⊠ ⊠ "),
    }
}

async fn greet(req: HttpRequest) -> impl Responder {
    let name = req.match_info().get("name").unwrap_or("World");
    format!("Hello {}!", &name)
}

#[actix_web::main]
async fn main() -> std::io::Result<()> {
    let printer = Printer::new();
    let app_state = web::Data::new(AppState {
        printer: Mutex::new(printer),
    });

    HttpServer::new(move || {
        App::new()
            .app_data(app_state.clone())
            .route("/", web::get().to(greet))
            .route("/{name}", web::get().to(greet))
            .route("/print", web::post().to(print_job))
    })
    .bind("127.0.0.1:8080")?
    .run()
    .await
}
```