# Deployment Plan for Kronos Calendar Application

## 1. Overview

The Kronos Calendar Application will be deployed using Docker and Docker Compose, ensuring a consistent environment for both backend and frontend services.   
The application can be easily started using a simple shell script or batch file, designed for non-technical users to deploy the system with minimal effort.  
This document outlines the deployment plan, including the prerequisites, steps, and considerations to ensure successful deployment.

## 2. Prerequisites

Before starting the deployment process, ensure the following tools are installed:  
- Docker  
- Docker Compose  
- Git  
- Text Editor/IDE (e.g., VS Code, IntelliJ IDEA)

## 3. Architecture Overview

The Kronos Calendar Application consists of two major services:  
- Backend (Python): Manages API and logic, exposes port 5000.  
- Frontend (Vue.js): Provides UI, runs on port 8080.  
These services are managed through Docker Compose.

## 4. Deployment Steps

Step 1: Clone the Repository  
git clone https://github.com/aaronchristian99/kronos-app.git  
cd kronos-app  
  
Step 2: Configure Environment Variables  
Create a .env file in root directory:  
MYSQL\_DATABASE=YOUR-DATABASE  
MYSQL\_USER=YOUR-DATABASE-USER  
MYSQL\_PASSWORD=YOUR-DATABASE-PASSWORD  
MYSQL\_ROOT\_PASSWORD=YOUR-DATABASE-ROOT-PASSWORD  
  
ENVIRONMENT=development

Create a .env file in backend directory:

DB\_CONNECTION=YOUR-DATABASE  
DB\_NAME=YOUR-DATABASE-NAME   
DB\_USER=YOUR-DATABASE-USER (Same as your MYSQL\_USER)  
DB\_PASSWORD=YOUR-DATABASE-PASSWORD (Same as your MYSQL\_PASSWORD)  
DB\_HOST=YOUR-DATABASE-HOST (Should the database container name eg. mysql from current docker-compose.yml)  
DB\_PORT=YOUR-DATABASE-PORT  
  
Step 3: Review Docker Compose Configuration  
Ensure docker-compose.yml is correct.  
  
Step 4: Build and Run the Application  
docker-compose build –no-cache  
docker-compose up -d  
docker exec -it python bash  
alembic upgrade head  
  
Step 5: Verify the Deployment  
Visit http://localhost:8080

## 5. Script Automation for Non-Technical Users

Linux/macOS: start.sh  
#!/bin/bash  
echo "Starting the Kronos application..."  
docker-compose up --build -d  
echo "Application is running at http://localhost:8080"  
  
Windows: start.bat  
@echo off  
echo Starting the Kronos application...  
docker-compose up --build -d  
echo Application is running at http://localhost:8080  
pause

## 6. Troubleshooting and Error Handling

- Docker errors: Ensure Docker Desktop is installed and running.  
- Port conflicts: Modify docker-compose.yml ports.  
- Missing environment variables: Check .env file.

## 7. Deployment Verification and Testing

- Verify Docker Containers: docker ps  
- Test Backend API: Postman or frontend  
- Test Frontend Interface: Full UI test

## 8. Post-Deployment Considerations

- Monitor Logs: docker logs <container-name>  
- Database Setup: Run migrations if needed (e.g., alembic upgrade head)

## 9. Conclusion

Following this deployment plan ensures a successful deployment of the Kronos Calendar Application in a Dockerized environment for both technical and non-technical users.