デモ プロジェクト

概要説明

このデモ プロジェクトは、工場設備管理アプリのプロトタイプ開発をテーマとしています。工場内の設備の稼働状況やメンテナンス情報を一元管理し、効率的な運用をサポートすることを目的としています。

プロジェクトの目的

- **設備の稼働状況のリアルタイム監視**: センサーを活用して、設備の稼働状況をリアルタイムで監視します。
- **メンテナンス管理**: 設備のメンテナンススケジュールを管理し、予防保全を実現します。
- **データ分析**: 設備の稼働データを分析し、効率的な運用方法を提案します。

歴史的な経緯
工場の設備管理は、初期には人手による目視確認や定期的な巡回が主流でした。
しかし、これには限界があり、異常の早期発見や効率的な監視が難しいという課題がありました。
近年では、IoT 技術やセンサー技術の導入により、リアルタイムでのデータ収集と分析が可能となり、効率的な監視システムの導入が進んでいます。

成功例
1. **リアルタイム監視で稼働率向上**:
   - ある工場では、IoT センサーを導入し、設備の稼働状況をリアルタイムで監視するシステムを導入しました。その結果、稼働率が 38% 向上し、生産性が大幅に改善されました。
2. **予知保全の実現**:
   - 予知保全システムを導入した工場では、異常の早期発見と迅速な対応が可能となり、設備の故障率を未然に防ぐことができました（10~15% の削減）。

失敗例
1. **導入コストの問題**:
   - ある工場では、稼働監視システムの導入に高額なコストがかかり、予算を超過してしまいました。また、古い設備に対応できないシステムを導入したため、追加の設備投資が必要となり、結果的にコストが増大しました。
2. **データの信頼性の問題**:
   - 手動でのデータ記録に依存していた工場では、人為的ミスやデータの不正確さが問題となり、正確な分析ができず、効率の改善が進まなかった。

ソフトウェアでの解決方法
1. **リアルタイムデータ収集と分析**:
   - IoT センサーや AI 技術を活用し、リアルタイムでデータを収集・分析することで、異常の早期発見と迅速な対応が可能になります。
2. **自動化と予知保全**:
   - 自動化された監視システムを導入することで、人手不足やスキル不足の問題を解決し、予知保全を実現します。
3. **データの可視化と一元管理**:
   - データを一元管理し、可視化することで、設備の稼働状況をリアルタイムで把握し、効率的な監視が可能になります。

これらのソフトウェアソリューションを導入することで、
工場の設備の稼働監視における課題を効果的に解決し、生産性の向上やコスト削減を実現することができます。

ヒアリングドキュメント
システムに求められる非機能要件ですか、そうですね。まあやっぱり止まらないようにしてほしいなと思います。あの休日も正月もそうですけどね。それからユーザーの数はまあ、おそらくですけど、同時にそうですね。10 人くらい捌ければいいのかなと思うんですよ。で、後から機能を追加するかもしれないので、ええ、そこの対応ができればいいかなと思ってます。ええそうですね。データのバックアップとかもちろんした方がいいとは思うんですけど、特にええとユーザーさんの課金の情報とか、予約の情報っていうのは、私たちの方で管理するってよりもそうですね。その連携サイトの方で、もちろん持っているわけですけど、ある程度はこちらでも持っておきたいですね。でも多分なくなると困るから、それは 24 時間 365 日すぐでも戻せるようにしてほしいと思います。まあいつお客さんいらっしゃってるかわかりませんからね。予約も含めて現行のシステムはありません。新規に全部作ります。セキュリティですけどね。えっと、もちろんですけど、個人情報に近いところを管理することになると思いますから、その部分に関してはすごく厳密なものが必要だと思います。エコロジーですか。太陽光発なんかが使える環境があると、いいかなと思いますけどね。まあほとんどスマホの中のアプリケーションだから、そこまではないかなと思います。

文書生成ガイドライン
1. **作成言語の指定**
　コメントおよび`.md`ファイルに記載する文章は【すべて日本語】で記述してください。

2. **書式と明確な記述**
　- コメントは簡潔で分かりやすく記述してください。
　- `.md`ファイルでは、見出し（Markdown の`#`）などを使用して、内容を読みやすく整理してください。
　- `.md`ファイルに実施手順に必要な CLI コマンドは記載して、**プログラムのソースコード**は絶対記載しないでください。

Vue 関連タスクのガイドライン
1. **Vue プロジェクトの生成**
　- このタスクでは、以下の必須タスクを参考に Vue プロジェクトを構成し、関連するソースコードを生成してください。

2. **具体的な実行手順**
　1. `package.json` を生成します
　2. `main.js` を生成します
　3. `App.vue` を生成します
　4. 画面遷移図に従って、画面遷移できるようにソースコードを生成します
　5. `vue-cli-service` をインストールします
　　- 手動でインストールする必要がある場合のコマンド例:
　　```bash
　　npm install @vue/cli-service
　　```
　6. 開発サーバーの起動し、ローカル環境で動作確認できる状態で必要なリソースを全て作成する。

Azure 関連タスクのガイドライン
1. **Azure リソースを使用する場合**
　作成するプログラムで Azure リソース（例: 仮想マシン、ストレージアカウント、SQL データベースなど）を使用する場合は、リソース作成手順を次のポイントに従って記述してください。手順の実施は Azure の初心者です。初心者でも分かるように手順のステップを詳細に作成してください。

2. **Azure CLI を使用した詳細な作成手順の記述**
　- 必ず Azure CLI を利用してリソースを作成する方法を記載してください。
　- 以下に記述例を示します。

　例 1: 仮想マシンの作成
　```bash
　# 1. Azure ログイン
　az login

　# 2. リソースグループの作成
　az group create --name <リソースグループ名> --location <リージョン>

　# 3. 仮想マシンの作成
　az vm create ¥
　　--resource-group <リソースグループ名> ¥
　　--name <仮想マシン名> ¥
　　--image UbuntuLTS ¥
　　--admin-username <ユーザー名> ¥
　　--generate-ssh-keys
　```

　例 2: ストレージアカウントの作成
　```bash
　# ストレージアカウントの作成
　az storage account create ¥
　　--name <ストレージアカウント名> ¥
　　--resource-group <リソースグループ名> ¥
　　--location <リージョン> ¥
　　--sku Standard_LRS
　```

3. **CLI コマンドの動作確認**
　記述した手順が再現性を持つよう、実行可能であることを確認してください。