Gear Up Frontend

Modern, responsive vehicle service management system built with Next.js 15 and React 18.

🚀 Tech Stack

- **Next.js 15** - React framework with App Router
- **React 18** - UI library with modern hooks
- **TypeScript** - Type-safe development
- **Tailwind CSS** - Utility-first styling
- **shadcn/ui** - High-quality React components
- **Lucide Icons** - Beautiful icon library
- **Docker** - Containerization
- **Google Kubernetes Engine** - Cloud deployment

📋 Features

Customer Portal

- 🚗 Vehicle registration and management
- 📅 Service appointment booking
- 📊 Real-time appointment status tracking
- 💬 AI-powered chatbot assistance
- 📜 Service history and invoices

Employee Dashboard

- ✅ Task management and tracking
- ⏱️ Work time logging against projects/appointments
- 📈 Personal productivity statistics
- 🔔 Appointment notifications
- 📊 Work distribution analytics

Admin Console

- 👥 User and employee management
- 🏗️ Project creation and oversight
- 📋 Task assignment and tracking
- 📊 Time log monitoring and reporting
- 📈 Team performance analytics
- 🚗 Vehicle and customer management

🛠️ Prerequisites

- Node.js 18+ (20+ recommended)
- npm 9+ or pnpm

- Backend API running (gear-up-be)

⚙️ Environment Configuration

Create `.env.local`:

```
# API Configuration
NEXT_PUBLIC_API_BASE_URL=http://localhost:8080/api

# Optional: Analytics, etc.
# NEXT_PUBLIC_GA_ID=your-ga-id
```

🏃 Running Locally

Getting Started

First, run the development server:

```
npm run dev
# or
yarn dev
# or
pnpm dev
# or
bun dev
```

Open http://localhost:3000 with your browser to see the result.

You can start editing the page by modifying `app/page.tsx`. The page auto-updates as you edit the file.

This project uses `next/font` to automatically optimize and load Geist, a new font family for Vercel.

Styling Guide

Colors

This project includes a custom color palette defined in `src/app/globals.css`. Here are the available colors and how to use them:

CSS Variables

```
/* Primary colors */
--color-primary: #163172;    /* Dark blue */
--color-secondary: #1E56A0;  /* Medium blue */
--color-ternary: #D6E4F0;    /* Light blue */
--color-bg: #F6F6F6;         /* Light gray background */
```

```
/* Usage in CSS */
.my-element {
  color: var(--color-primary);
  background-color: var(--color-secondary);
  border: 1px solid var(--color-ternary);
}
```

Tailwind Utility Classes

For easier usage, we've created utility classes:

```
<!-- Text colors -->
<p class="text-primary">Primary text color</p>
<p class="text-secondary">Secondary text color</p>
<p class="text-ternary">Light text color</p>

<!-- Background colors -->
<div class="bg-primary">Primary background</div>
<div class="bg-secondary">Secondary background</div>
<div class="bg-ternary">Light background</div>
<div class="bg-custom">Custom background</div>

<!-- Border colors -->
<div class="border-primary">Primary border</div>
<div class="border-secondary">Secondary border</div>
<div class="border-ternary">Light border</div>
```

Fonts

The project uses Geist font family with optimized loading:

Default Font Stack

- **Sans-serif**: Geist (automatically loaded by Next.js)
- **Monospace**: Geist Mono (available as `--font-mono`)

Usage in Components

```
// Using default sans font (automatic)
export default function MyComponent() {
  return (
    <div className="font-sans">
      <h1>This uses the default Geist font</h1>
      <code className="font-mono">This uses monospace font</code>
    </div>
  );
}
```

Custom Font Loading (if needed)

```
import { Geist, Geist_Mono } from 'next/font/google';

const geistSans = Geist({
  variable: '--font-geist-sans',
  subsets: ['latin'],
});

const geistMono = Geist_Mono({
  variable: '--font-geist-mono',
  subsets: ['latin'],
});

export default function RootLayout({ children }) {
  return (
    <html lang="en" className={`${geistSans.variable} ${geistMono.variable}`}>
      <body>{children}</body>
    </html>
  );
}
```

Learn More

To learn more about Next.js, take a look at the following resources:

- Next.js Documentation - learn about Next.js features and API.
- Learn Next.js - an interactive Next.js tutorial.

You can check out the Next.js GitHub repository - your feedback and contributions are welcome!

Deploy on Vercel

The easiest way to deploy your Next.js app is to use the Vercel Platform from the creators of Next.js.

Check out our Next.js deployment documentation for more details.

#� �g�e�a�r�-�u�p�-�f�e�

�

�