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

Development Server

```
# Install dependencies
npm install

# Start development server
npm run dev
```

Open http://localhost:3000 in your browser.

Docker

```
# Build image
docker build -t gearup-frontend .

# Run container
docker run -p 3000:3000 \
  -e NEXT_PUBLIC_API_BASE_URL=http://localhost:8080/api \
  gearup-frontend
```

Docker Compose

```
# Start frontend only
docker-compose up -d

# View logs
docker-compose logs -f
```

🎨 Styling Guide

Color Palette

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

Tailwind Utilities

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

// Backgrounds
<div className="bg-primary">Dark blue background</div>
<div className="bg-secondary">Medium blue background</div>
<div className="bg-ternary">Light blue background</div>

// Borders
<div className="border-primary">Primary border</div>
```

Fonts

- **Sans-serif**: Geist (default)
- **Monospace**: Geist Mono

```
<div className="font-sans">Regular text</div>
<code className="font-mono">Code text</code>
```

📦 Building for Production

```
# Create production build
npm run build

# Test production build locally
npm run start

# Production server runs on http://localhost:3000
```

🧪 Testing

```
# Lint code
npm run lint

# Type check
npx tsc --noEmit
```

Deployment

Google Kubernetes Engine (GKE)

Automated deployment via GitHub Actions:

1. **Setup**: Follow `../DEPLOYMENT_GUIDE.md`
2. **Configure**: Add GitHub secrets for GCP
3. **Deploy**: Push to `main` or `production` branch

```
# Manual deployment
kubectl apply -f k8s/namespace.yaml
kubectl apply -f k8s/frontend-configmap.yaml
kubectl apply -f k8s/frontend-deployment.yaml
```

Vercel (Alternative)

▲ Deploy

1. Import repository
2. Set environment variable: `NEXT_PUBLIC_API_BASE_URL`
3. Deploy

Project Structure

```
src/
├── app/                    # Next.js App Router pages
│   ├── admin/             # Admin dashboard routes
│   ├── employee/          # Employee dashboard routes
│   ├── customer/          # Customer portal routes
│   └── auth/              # Authentication pages
├── components/            # React components
│   ├── admin/            # Admin-specific components
│   ├── employee/         # Employee-specific components
│   ├── customer/         # Customer-specific components
│   └── ui/               # shadcn/ui components
├── contexts/             # React Context providers
│   └── AuthContext.tsx   # Authentication state
├── lib/                  # Utilities and services
```

```
│   ├── services/        # API service functions
│   └── utils.ts         # Helper functions
└── types/               # TypeScript type definitions
```

🔑 Key Features Implementation

Authentication

```
import { useAuth } from '@/contexts/AuthContext';

function MyComponent() {
  const { user, login, logout, isAuthenticated } = useAuth();

  return (
    <div>
      {isAuthenticated ? (
        <p>Welcome {user.name}</p>
      ) : (
        <button onClick={login}>Login</button>
      )}
    </div>
  );
}
```

API Calls

```
import { projectService } from '@/lib/services/projectService';

async function fetchProjects() {
  const projects = await projectService.getAllProjects();
  return projects;
}
```

Time Logging

```
import { timeLogService } from '@/lib/services/timeLogService';

await timeLogService.createTimeLog({
  description: 'Completed oil change',
  startTime: '2025-01-15T09:00:00',
  endTime: '2025-01-15T11:00:00',
  taskId: 123,
  appointmentId: 456
});
```

Routes

Public Routes

- `/` - Landing page
- `/auth/login` - Login page
- `/auth/register` - Registration

Customer Routes

- `/customer/dashboard` - Customer dashboard
- `/customer/vehicles` - Vehicle management
- `/customer/appointments` - Appointment booking

Employee Routes

- `/employee/dashboard` - Employee dashboard
- `/employee/tasks` - Task list
- `/employee/time-logs` - Time tracking

Admin Routes

- `/admin/dashboard` - Admin overview
- `/admin/projects` - Project management
- `/admin/employees` - Employee management
- `/admin/time-logs` - Team time tracking

Security

- JWT token stored in localStorage
- Protected routes with middleware
- Role-based UI rendering
- XSS protection via React
- HTTPS in production

Browser Support

- Chrome (latest)
- Firefox (latest)
- Safari (latest)
- Edge (latest)

Contributing

1. Create feature branch from `main`
2. Follow TypeScript and ESLint rules
3. Test responsiveness (mobile, tablet, desktop)
4. Submit pull request

Resources

- Next.js Documentation - Next.js features and API
- React Documentation - React fundamentals
- Tailwind CSS - Styling utility classes
- shadcn/ui - Component library
- TypeScript Handbook - TypeScript guide

License

MIT License - see LICENSE file for details

Team

EAD Group Project 2025

For deployment instructions, see DEPLOYMENT_GUIDE.md